IBM DB2 Everyplace
Guía de desarrollo de aplicaciones

Versión 8.14

SC10-3942-01

IBM DB2 Everyplace
Guía de desarrollo de aplicaciones

Versión 8.14

SC10-3942-01

Nota:
Antes de utilizar este manual y el producto al que da soporte, lea la información general incluida en la sección "Avisos" en la página 371.

Segunda edición (octubre de 2003)

Este manual es la traducción del original inglés *IBM DB2 Everyplace Application Development Guide Version 8.1.4*, (SC18-7185-01).

Esta edición es aplicable a la versión 8.1 de DB2 Everyplace (número de producto 5724-D04) y a todos los releases y modificaciones subsiguientes hasta que se indique lo contrario en nuevas ediciones.

Esta edición sustituye a SC18-7185-00

Este documento contiene información de propiedad de IBM. Se suministra bajo un contrato de licencia y está protegido por las leyes de copyright. La información contenida en esta publicación no incluye ninguna garantía sobre el producto y ninguna de las declaraciones proporcionadas en este manual se debe interpretar como tal.

Solicite las publicaciones a través del representante de IBM o de la sucursal de IBM que dé servicio en su localidad, o bien llamando al teléfono 1-800-879-2755 en Estados Unidos o al teléfono 1-800-IBM-4YOU en Canadá.

Cuando se envía información a IBM, se otorga a IBM el derecho a utilizar o distribuir dicha información en la forma que considere adecuada, sin contraer por ello ninguna obligación con el remitente.

© Copyright International Business Machines Corporation 1999,2003. Reservados todos los derechos.

Tabla de contenido

Parte 1. Introducción 1

Capítulo 1. Visión general de producto de DB2 Everyplace 3
Qué es DB2 Everyplace 3
Componentes de la solución DB2 Everyplace . . . 3
La base de datos portátil DB2 Everyplace 4
El DB2 Everyplace Sync Server 4
El DB2 Everyplace Sync Client 5
DB2 Everyplace Mobile Application Builder . . . 5
Las aplicaciones DB2 Everyplace de ejemplo . . . 5
Escenario DB2 Everyplace de ejemplo 6

Parte 2. Desarrollo de aplicaciones DB2 Everyplace 7

Capítulo 2. Cómo desarrollar aplicaciones C/C++ de DB2 Everyplace . 9
Cómo desarrollar aplicaciones C/C++ de DB2 Everyplace 9
Herramientas de desarrollo de C/C++ soportadas 10
Sistemas operativos soportados de C/C++ 11
Preparación, compilación y enlace de un proyecto C/C++ 11
Cómo probar una aplicación C/C++ 13
Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++ 15

Capítulo 3. Cómo desarrollar aplicaciones Java de DB2 Everyplace . 17
Sistemas operativos soportados de la interfaz JDBC 17
Desarrollo de aplicaciones Java de DB2 Everyplace 17

Capítulo 4. Cómo desarrollar aplicaciones Java Sync Client. 19
Sistemas operativos soportados por la API de Java Sync. 19
Las API de IBM Java Sync 19
Visión general de los proveedores de sincronización de DB2 Everyplace 19
Sincronización nativa de DB2 Everyplace 20
Instalación de proveedores de sincronización nativa de DB2 Everyplace 20
Instalación del proveedor de sincronización nativa basado en JNI 21
Instalación del proveedor de sincronización basado en JNI 22
Instalación del proveedor de sincronización basado en JNI en dispositivos Nokia 9210/9290 Communicator utilizando Symbian V6 22
Instalación del proveedor de sincronización basado en JNI en Windows CE 23
Instalación y verificación del proveedor de sincronización nativa basada en la detección . . 24
Proveedores de sincronización de DB2 Everyplace Java 26
Sincronización de Java de DB2 Everyplace . . . 27
Sincronización J2ME MIDP de DB2 Everyplace 27
Cliente DB2 Everyplace Java Sync para Cloudscape 28

Capítulo 5. Cómo desarrollar aplicaciones de Visual Basic 31
Cómo desarrollar aplicaciones de Visual Basic de DB2 Everyplace 31
Sistemas operativos soportados de la interfaz de Visual Basic 32

Capítulo 6. Desarrollo de aplicaciones JSP 33
Soporte de JSP para los sistemas operativos . . . 33
Desarrollo de aplicaciones JSP de DB2 Everyplace 33
Visión general del soporte de JSP en DB2 Everyplace 34
Configuración del desarrollo de JSP 35
Verificación del soporte de JSP en una estación de trabajo Windows 35
Configuración del desarrollo de JSP en un dispositivo Windows CE 36
Instalación del entorno de ejecución de una JVM J9 en un dispositivo Windows CE 37
Instalación y verificación del soporte de JSP en un dispositivo Windows CE 38
Instalación y verificación del soporte de JSP en un dispositivo Symbian OS Versión 6 39
Transferencia de una aplicación JSP a un dispositivo Windows CE 40
Ejecución de una aplicación JSP 41
Configuración del mini servidor de Web HTTP 41
Ejecución de una aplicación JSP en una estación de trabajo Windows 43
Ejecución de una aplicación JSP en un dispositivo Windows CE 43
Ejecución de una aplicación JSP en un dispositivo Symbian OS Versión 6 44
Subconjuntos JSP Versión 1.1 soportados. 45
Códigos de personalización de IBM para acceder a bases de datos de aplicación de JSP 48
Resolución de problemas de las aplicaciones de JSP 51

Capítulo 7. Desarrollo de aplicaciones .NET. 53
Soporte de sincronización. 53
Ubicaciones de archivo de API de ISync.Net . . 53
Utilización de la API de ISync.NET 54
Utilización de ISyncComponent 55

© Copyright IBM Corp. 1999,2003

Aplicación de ejemplo simple que utiliza la API de ISync.NET 56
Soporte para crear aplicaciones de .NET 56
Visión general del soporte de .NET para crear aplicaciones en la base de datos cliente 57
Visión general del modo de desarrollar aplicaciones ADO.NET utilizando el DB2 Everyplace .NET Data Provider 57
Código de aplicación del proveedor de datos de .NET de DB2 Everyplace para WinCE y Win32 . 61

Capítulo 8. Cómo conectarse a una base de datos de DB2 Everyplace . . . 65
Visión general de las tablas de base de datos de DB2 Everyplace 65
Manejo de conflictos de denominación 65
Cómo conectarse a la base de datos de DB2 Everyplace 66
Serialización de conexiones 67
Bases de datos DB2 Everyplace en soportes de almacenamiento de sólo lectura. 68

Capítulo 9. Recuperación gradual de datos por medio de CLI 69

Capítulo 10. Marcadores de parámetros 71
Visión general de los marcadores de parámetro . . 71
Ejemplos de utilización de marcador de parámetro 71
Marcadores de parámetro soportados de DB2 Everyplace 76

Capítulo 11. Comportamiento del cursor en el contexto de una conexión . 79

Capítulo 12. Cifrado de los datos locales 81
Visión general del cifrado local de los datos . . . 81
Establecimiento de una conexión con la base de datos DB2 Everyplace 82
Cómo otorgar privilegios de cifrado a un usuario. . 83
Creación de una tabla cifrada 84
Gestión de los privilegios de cifrado 84
Cifrado utilizando el DB2eCLP 85

Parte 3. Aplicaciones de ejemplo 91

Capítulo 13. Aplicaciones C/C++ de ejemplo 93

Capítulo 14. Las aplicaciones Java de ejemplo 95
Visión general de las aplicaciones Java de ejemplo 95
Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS 97
Instalación de WCE Tooling para WSDD para destinos de Palm OS 97
Creación de un proyecto de WSDD para DB2eAppl.java para destinos de Palm OS 98
Cómo añadir el controlador de JDBC de DB2 Everyplace y el paquete java.sql a la vía de acceso de creación 99
Importación de DB2eAppl.java a WSDD para Palm OS 99
Ejecución de DB2eAppl.java en un emulador de Palm OS 100
Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS 102
Instalación de WCE Tooling para WSDD para destinos no Palm OS 103
Cómo crear un proyecto WSDD y añadir archivos jar a la vía de acceso de creación para DB2eAppl.java para destinos no Palm OS 103
Importación de DB2eAppl.java a WSDD para destinos no Palm OS 104
Ejecución de las aplicaciones Java de ejemplo . . 105
Ejecución de DB2eAppl.java en Win32 105
Ejecución de DB2eAppl.java en Windows CE 106
Ejecución de DB2eAppl.java en QNX Neutrino o Linux incorporado. 108
Ejecución de DB2eAppl.java en Symbian . . . 108

Capítulo 15. La aplicación de Visual Basic de ejemplo 111
Visión general de la aplicación Visual Basic de ejemplo 111
Compilación y prueba del programa Visual Basic de ejemplo 114

Capítulo 16. Las aplicaciones JSP de ejemplo 117

Capítulo 17. Aplicaciones de sincronización de ejemplo 119
La aplicación Sync Client C/C++ de ejemplo . . . 119
Las aplicaciones de sincronización nativa de Java de ejemplo 121
Las aplicaciones de sincronización MIDP de Java de ejemplo 125
Desarrollo de la aplicación isync4j para MIDP con Sun Wireless Toolkit 129
Desarrollo de la aplicación isync4j para MIDP con ANT y la línea de mandatos de Sun Wireless Toolkit. 131
Compilación y ejecución de la aplicación de sincronización de Java de ejemplo GoISyncConsole. 132

Parte 4. Consulta 135

Capítulo 18. Interfaces de programación de aplicaciones (las API) 137
Soporte de sentencia SQL de DB2 Everyplace. . . 137
Visión general del soporte de sentencia de SQL de DB2 Everyplace 137

CALL 138
CREATE INDEX 141
CREATE TABLE 143
DELETE 150
DROP 153
EXPLAIN 154
GRANT 156
INSERT 157
REORG TABLE. 160
REVOKE 161
SELECT 162
UPDATE 171
Compatibilidad entre tipos de datos para las operaciones de asignación y comparación . . . 176
Tipos de datos por omisión y simbólicos de SQL 177
Atributos de tipos de datos. 177
Listado de los SQLSTATE 180
Resumen de códigos de clase de SQLState. . . 180
Mensajes de SQLSTATE notificados por SQL 181
Mensajes de SQLSTATE notificados por CLI . . 185
Mensajes de SQLState notificados por JDBC . . 193
Funciones de CLI de DB2 soportadas 194
Resumen de las funciones de CLI de DB2 . . . 194
Clave para las descripciones de funciones de CLI de DB2 198
SQLAllocConnect—Asignar descriptor de conexión 199
SQLAllocEnv—Asignar descriptor de entorno 200
SQLAllocHandle—Asignar descriptor de contexto 200
SQLAllocStmt—Asignar un descriptor de sentencia 203
SQLBindCol—Enlazar una columna a una variable de aplicación 203
SQLBindParameter—Enlazar un marcador de parámetro a un almacenamiento intermedio . . 207
SQLConnect—Conectar con una fuente de datos 211
SQLColumns - Obtener información de columna para una tabla 216
SQLDescribeCol—Devolver un conjunto de atributos de una columna 220
SQLDisconnect—Desconectar de una fuente de datos 222
SQLEndTran—Solicitar COMMIT o ROLLBACK 223
SQLError—Recuperar información sobre errores 225
SQLExecDirect—Ejecutar una sentencia directamente 225
SQLExecute—Ejecutar una sentencia 227
SQLFetch—Recuperar la fila siguiente 229
SQLFetchScroll—Recuperar conjunto de filas y devolver datos para todas las columnas enlazadas. 231
SQLForeignKeys—Obtener la lista de columnas de clave foránea 238
SQLFreeConnect—Liberar descriptor de conexión 241
SQLFreeEnv—Liberar descriptor de entorno . . 242
SQLFreeHandle—Liberar recursos de descriptor de contexto 242
SQLFreeStmt—Liberar (o restaurar) un descriptor de sentencia 244
SQLGetConnectAttr—Obtener el valor actual de un atributo de conexión 246
SQLGetCursorName—Obtener nombre de cursor 248
SQLGetData—Obtener datos de una columna 250
SQLGetDiagRec—Obtener varios valores de campos del registro de diagnósticos 254
SQLGetInfo—Obtener información general . . 257
SQLGetStmtAttr—Obtener el valor actual de un atributo de sentencia 260
SQLNumParams - Obtener número de parámetros en una sentencia SQL 264
SQLNumResultCols—Obtener número de columnas resultantes 265
SQLPrepare—Preparar una sentencia 266
SQLPrimaryKeys—Obtener columnas de clave primaria de una tabla 268
SQLRowCount—Obtener número de filas . . . 271
SQLSetConnectAttr—Establecer opciones referentes a una conexión 272
SQLSetStmtAttr—Establecer opciones referentes a una sentencia. 276
SQLTables - Obtener información de tabla . . . 283
Conversión de datos por funciones de CLI de DB2 285
Métodos de JDBC soportados 287
Visión general del soporte de JDBC de DB2 Everyplace 287
Interfaces en el paquete de java.sql 287
Interfaces en el paquete javax.sql 305
Clases .NET soportadas 307
Miembros de DB2eCommandBuilder 307
Miembros de DB2eCommand 308
Miembros de DB2eConnection. 309
Miembros de DB2eDataAdapter 309
Miembros de DB2eDataReader 310
Miembros de DB2eError. 312
Miembros de DB2eException 312
Miembros de DB2eParameter 312
Miembros de DB2eTransaction. 313
Enumeración de DB2eType 314
API C de IBM Sync Client 315
Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2 315
Resumen de funciones de la API C de IBM Sync Client 317
Tipos de datos de la API C de IBM Sync Client 319
Descripciones de funciones de la API C de IBM Sync Client 321

Capítulo 19. Tablas base del catálogo del sistema DB2 Everyplace 355

Capítulo 20. Límites de DB2 Everyplace 357

Capítulo 21. Palabras reservadas de DB2 Everyplace 359

Capítulo 22. Soporte de idioma nacional (NLS) 361
Soporte NLS por sistema operativo de DB2 Everyplace 361
Codificación de caracteres en aplicaciones Java . . 363
Habilitadores de idioma de DB2 Everyplace . . . 363
Soporte UNICODE en DB2 Everyplace 364

Capítulo 23. El conjunto de información de DB2 Everyplace . . . 367
Archivos PDF y HTML de DB2 Everyplace . . . 367
Documentación en línea sobre DB2 Everyplace . . 368

Parte 5. Apéndices 369

Avisos 371
Marcas registradas. 374

Glosario 375

Índice. 379

Cómo ponerse en contacto con IBM 387
Información sobre productos 387

Parte 1. Introducción

Capítulo 1. Visión general de producto de DB2 Everyplace 3
Qué es DB2 Everyplace 3
Componentes de la solución DB2 Everyplace . . . 3
La base de datos portátil DB2 Everyplace 4
El DB2 Everyplace Sync Server 4
El DB2 Everyplace Sync Client 5
DB2 Everyplace Mobile Application Builder . . . 5
Las aplicaciones DB2 Everyplace de ejemplo . . . 5
Escenario DB2 Everyplace de ejemplo 6

© Copyright IBM Corp. 1999,2003

Capítulo 1. Visión general de producto de DB2 Everyplace

Este apartado proporciona una introducción a DB2 Everyplace, una descripción de los componentes que forman la solución DB2 Everyplace y un ejemplo de un caso típico de DB2 Everyplace. Este apartado contiene los temas siguientes:

- "Qué es DB2 Everyplace"
- "Componentes de la solución DB2 Everyplace"
  - "La base de datos portátil DB2 Everyplace" en la página 4
  - "El DB2 Everyplace Sync Server" en la página 4
  - "El DB2 Everyplace Sync Client" en la página 5
  - "DB2 Everyplace Mobile Application Builder" en la página 5
  - "Las aplicaciones DB2 Everyplace de ejemplo" en la página 5
- "Escenario DB2 Everyplace de ejemplo" en la página 6

Qué es DB2 Everyplace

DB2 Everyplace forma parte de la solución de IBM para procesos de informática distribuida. Mediante DB2 Everyplace, los profesionales que se desplazan con frecuencia (tales como vendedores, inspectores, auditores, técnicos de mantenimiento, médicos, agentes inmobiliarios y tasadores de seguros) pueden tener acceso a datos vitales que necesitan mientras están lejos de su despacho.

Las empresas pueden ahora transferir sus datos corporativos DB2 a dispositivos portátiles o incorporados. Con DB2 Everyplace, puede acceder a una base de datos contenida en su dispositivo portátil y realizar actualizaciones en ella. Con DB2 Everyplace Sync Server, puede sincronizar datos entre el dispositivo portátil y otras fuentes de datos ubicadas en la empresa. El Adaptador de Archivos le permite distribuir archivos y aplicaciones hacia usuarios móviles.

DB2 Everyplace es una base de datos relacional que reside en su dispositivo portátil. Para acceder a los datos del dispositivo portátil, puede escribir sus propias aplicaciones utilizando las herramientas para el desarrollo rápido de aplicaciones, el conjunto soportado de funciones de CLI (Call Level Interface) de DB2, los métodos de JDBC (Java Database Connectivity) o los métodos ADO.NET.

Componentes de la solución DB2 Everyplace

La solución DB2 Everyplace tiene los componentes y las características clave siguientes:

- La base de datos portátil DB2 Everyplace.
- El DB2 Everyplace Sync Server.
- El DB2 Everyplace Sync Client.
- El DB2 Everyplace Mobile Application Builder.
- Las aplicaciones de ejemplo de DB2 Everyplace

© Copyright IBM Corp. 1999,2003

La base de datos portátil DB2 Everyplace

Esta base de datos reside en el dispositivo portátil. La base de datos portátil se incluye con DB2 Everyplace Database Edition, DB2 Everyplace Enterprise Edition y DB2 Everyplace Software Development Kit. Otro componente asociado con la base de datos portátil es:

- La aplicación de ejemplo (lado del motor)

La base de datos portátil DB2 Everyplace está disponible para:

- Palm OS
- Symbian OS
- Windows CE/Pocket PC
- Win32 (Windows® 95, Windows® 98, Windows® NT®, Windows® 2000® y Windows® XP®)
- Dispositivos QNX Neutrino, Linux y Linux incorporado.

DB2 Everyplace también da soporte a dispositivos portátiles MIDP que utilizan la base de datos MIDP.

El DB2 Everyplace Sync Server

El DB2 Everyplace Sync Server se incluye con DB2 Everyplace Enterprise Edition. Otros componentes importantes asociados al Sync Server incluyen:

- El Centro de administración de dispositivos portátiles de DB2 Everyplace
- Las aplicaciones de ejemplo (lado del servidor)

Puede sincronizar datos y aplicaciones entre dispositivos portátiles DB2 Everyplace y fuentes de datos corporativas utilizando el DB2 Everyplace Sync Server y DB2 Everyplace Sync Client.

La sincronización de datos puede ser bidireccional o unidireccional. Los datos se pueden actualizar en la base de datos de DB2 Everyplace en el dispositivo portátil o en la base de datos corporativa. Por ejemplo, los usuarios pueden bajar un subconjunto de datos desde una base de datos DB2 para z/OS a una base de datos de DB2 Everyplace en el dispositivo portátil, visualizar los datos, efectuar cambios en los mismos y a continuación sincronizar los datos modificados de vuelta al servidor z/OS. El DB2 Everyplace Sync Server también proporciona un mecanismo para la resolución de conflictos.

El DB2 Everyplace Sync Server brinda una herramienta de administración que le será de ayuda para gestionar y proporcionar servicios de sincronización a grupos de usuarios con necesidades parecidas de sincronización de datos. En la publicación Sync Server Administration Guide dispone de más información sobre el Centro de administración de dispositivos portátiles.

DB2 Everyplace Sync Server da soporte a la sincronización de datos relacionales con cualquier fuente de datos que tenga una interfaz JDBC, como por ejemplo DB2 Universal Database.

DB2 Everyplace Sync Server da soporte a la sincronización de datos relacionales con las fuentes de datos siguientes:

- DB2 Universal Database para z/OS
- DB2 Universal Database para iSeries
- DB2 Universal Database para Linux, UNIX y Windows

- Cualquier fuente de datos con una interfaz JDBC

El DB2 Everyplace Sync Client

El DB2 Everyplace Sync Client se incluye con DB2 Everyplace Enterprise Edition.

El DB2 Everyplace Sync Client, que se ejecuta en dispositivos portátiles, consta de aplicaciones que trabajan conjuntamente con DB2 Everyplace Sync Server. Gestiona la sincronización bidireccional de los datos relacionales corporativos con la base de datos DB2 Everyplace situada en el dispositivo portátil. El dispositivo portátil también maneja operaciones referentes a suscripciones de archivos para facilitar la distribución de aplicaciones portátiles al dispositivo, y puede ejecutar procedimientos almacenados situados en una base de datos DB2.

Sync Client está disponible para los sistemas operativos siguientes:

- Palm OS
- Symbian OS
- Windows CE/Pocket PC
- Win32 (Windows® 95, Windows® 98, Windows® NT®, Windows® 2000® y Windows® XP®)
- Dispositivos QNX Neutrino, Linux y Linux incorporado

Para obtener información sobre las API (interfaces de programación de aplicaciones) que se proporcionan con el Sync Client, consulte el manual *DB2 Everyplace Guía de desarrollo de aplicaciones*.

DB2 Everyplace Mobile Application Builder

El DB2 Everyplace Mobile Application Builder se incluye con el Software Development Kit y también se puede bajar del sitio Web de IBM.

Puede utilizar DB2 Everyplace Mobile Application Builder para desarrollar aplicaciones DB2 Everyplace para plataformas Palm OS, WinCE, Symbian OS y otras plataformas que den soporte a una interfaz de usuario y una Máquina virtual Java. Mediante Mobile Application Builder, puede crear aplicaciones sin tener que escribir una sola línea de código. Para obtener información sobre cómo obtener Mobile Application Builder, visite el sitio Web de DB2 Everyplace.

Existen otras herramientas de desarrollo que incluyen WebSphere Studio Device Developer, Visual Age Micro Edition, Metrowerks CodeWarrior y el GNU Software Developer's Kit.

Las aplicaciones DB2 Everyplace de ejemplo

Las aplicaciones de ejemplo proporcionan una muestra de aplicación que utiliza DB2 Everyplace. Puede utilizar la aplicación de ejemplo Visiting Nurse para comprobar rápidamente la sincronización bidireccional entre la base de datos portátil y el Sync Server. Las aplicaciones de ejemplo constan de dos partes, una que se ejecuta en el Sync Server y otra que se ejecuta en la base de datos portátil. Esta aplicación de ejemplo para una base de datos portátil demuestra el funcionamiento del motor de base de datos en un entorno autónomo. Cuando la aplicación de ejemplo del Sync Server y la aplicación de ejemplo del motor de base de datos se ejecutan juntas, funcionan como una aplicación completa que invoca a todos los componentes de DB2 Everyplace.

El IBM Sync es asimismo una aplicación de ejemplo que muestra el modo de utilizar la API de Cliente de sincronización de DB2 Everyplace para sincronizar las tablas de las suscripciones definidas en MDAC.

El Procesador de línea de mandatos es una herramienta para el desarrollo de aplicaciones que se proporciona como aplicación de ejemplo de utilización de DB2 Everyplace en plataformas que disponen de una interfaz de línea de mandatos. El Procesador de línea de mandatos se utiliza para la base de datos DB2 Everyplace en dispositivos portátiles. El Sync Server no lo utiliza.

Las sentencias de SQL soportadas por DB2 Everyplace permiten crear y eliminar una tabla o índice, y suprimir, insertar y actualizar las filas de una tabla.

Consulte la publicación *DB2 Everyplace Guía de desarrollo de aplicaciones* para obtener más información sobre las sentencias SQL soportadas.

Escenario DB2 Everyplace de ejemplo

Los tasadores de seguros son los encargados de inspeccionar los bienes dañados de los clientes que presentan una reclamación. En la mayoría de las empresas, el tasador visita los bienes del reclamante, rellena formularios para validar o rechazar la reclamación y evalúa el importe de los daños por los que se debe indemnizar al reclamante. Más adelante, cuando el tasador regresa a su oficina, la información de los formularios se entra manualmente en el sistema informático de la empresa, lo cual es un proceso tedioso y caro.

Este proceso se puede simplificar considerablemente equipando a los tasadores con un dispositivo portátil que ejecute una aplicación DB2 Everyplace. Mediante el dispositivos portátiles, los tasadores puede acceder, dondequiera que esté, a su plan de inspecciones, a su ruta de trabajo y a la información sobre la póliza de seguro del reclamante. Los tasadores pueden también cumplimentar el formulario de tasación en el dispositivo portátil. Cuando los tasadores regresan a la oficina, pueden sincronizar los datos de sus dispositivos portátiles con el sistema informático de la empresa, transfiriendo los nuevos datos del formulario de tasación a la base de datos corporativa de la empresa. Si los tasadores necesitan información in situ, pueden sincronizar inmediatamente los datos de sus dispositivos portátiles con el sistema informático de la empresa, a través de un módem. De esta forma, el proceso de evaluar reclamaciones puede prescindir totalmente del papel como soporte de información, lo cual supone un gran ahorro de costes para la empresa de seguros. Además, las reclamaciones se liquidan con más rapidez, pues el tasador tiene acceso inmediato a las bases de datos corporativas de la empresa.

Parte 2. Desarrollo de aplicaciones DB2 Everyplace

Capítulo 2. Cómo desarrollar aplicaciones C/C++ de DB2 Everyplace 9
Cómo desarrollar aplicaciones C/C++ de DB2 Everyplace 9
Herramientas de desarrollo de C/C++ soportadas 10
Sistemas operativos soportados de C/C++ 11
Preparación, compilación y enlace de un proyecto C/C++ 11
Cómo probar una aplicación C/C++ 13
Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++ 15

Capítulo 3. Cómo desarrollar aplicaciones Java de DB2 Everyplace 17
Sistemas operativos soportados de la interfaz JDBC 17
Desarrollo de aplicaciones Java de DB2 Everyplace 17

Capítulo 4. Cómo desarrollar aplicaciones Java Sync Client 19
Sistemas operativos soportados por la API de Java Sync 19
Las API de IBM Java Sync 19
Visión general de los proveedores de sincronización de DB2 Everyplace 19
Sincronización nativa de DB2 Everyplace 20
Instalación de proveedores de sincronización nativa de DB2 Everyplace 20
Instalación del proveedor de sincronización nativa basado en JNI 21
Instalación del proveedor de sincronización basado en JNI 22
Instalación del proveedor de sincronización basado en JNI en dispositivos Nokia 9210/9290 Communicator utilizando Symbian V6 22
Instalación del proveedor de sincronización basado en JNI en Windows CE 23
Instalación y verificación del proveedor de sincronización nativa basada en la detección . . 24
Proveedores de sincronización de DB2 Everyplace Java 26
Sincronización de Java de DB2 Everyplace . . . 27
Sincronización J2ME MIDP de DB2 Everyplace 27
Cliente DB2 Everyplace Java Sync para Cloudscape 28

Capítulo 5. Cómo desarrollar aplicaciones de Visual Basic 31
Cómo desarrollar aplicaciones de Visual Basic de DB2 Everyplace 31
Sistemas operativos soportados de la interfaz de Visual Basic 32

Capítulo 6. Desarrollo de aplicaciones JSP . . . 33
Soporte de JSP para los sistemas operativos . . . 33
Desarrollo de aplicaciones JSP de DB2 Everyplace 33
Visión general del soporte de JSP en DB2 Everyplace 34
Configuración del desarrollo de JSP 35
Verificación del soporte de JSP en una estación de trabajo Windows 35
Configuración del desarrollo de JSP en un dispositivo Windows CE 36
Instalación del entorno de ejecución de una JVM J9 en un dispositivo Windows CE 37
Instalación y verificación del soporte de JSP en un dispositivo Windows CE 38
Instalación y verificación del soporte de JSP en un dispositivo Symbian OS Versión 6 39
Transferencia de una aplicación JSP a un dispositivo Windows CE 40
Ejecución de una aplicación JSP 41
Configuración del mini servidor de Web HTTP 41
Ejecución de una aplicación JSP en una estación de trabajo Windows 43
Ejecución de una aplicación JSP en un dispositivo Windows CE 43
Ejecución de una aplicación JSP en un dispositivo Symbian OS Versión 6 44
Subconjuntos JSP Versión 1.1 soportados. 45
Códigos de personalización de IBM para acceder a bases de datos de aplicación de JSP 48
Resolución de problemas de las aplicaciones de JSP 51

Capítulo 7. Desarrollo de aplicaciones .NET . . 53
Soporte de sincronización. 53
Ubicaciones de archivo de API de ISync.Net . . 53
Utilización de la API de ISync.NET 54
Utilización de ISyncComponent 55
Aplicación de ejemplo simple que utiliza la API de ISync.NET. 56
Soporte para crear aplicaciones de .NET 56
Visión general del soporte de .NET para crear aplicaciones en la base de datos cliente 57
Visión general del modo de desarrollar aplicaciones ADO.NET utilizando el DB2 Everyplace .NET Data Provider. 57
Código de aplicación del proveedor de datos de .NET de DB2 Everyplace para WinCE y Win32 . 61

Capítulo 8. Cómo conectarse a una base de datos de DB2 Everyplace 65
Visión general de las tablas de base de datos de DB2 Everyplace 65
Manejo de conflictos de denominación 65
Cómo conectarse a la base de datos de DB2 Everyplace 66
Serialización de conexiones 67
Bases de datos DB2 Everyplace en soportes de almacenamiento de sólo lectura. 68

© Copyright IBM Corp. 1999,2003

Capítulo 9. Recuperación gradual de datos por medio de CLI 69

Capítulo 10. Marcadores de parámetros 71
Visión general de los marcadores de parámetro . . 71
Ejemplos de utilización de marcador de parámetro 71
Marcadores de parámetro soportados de DB2 Everyplace 76

Capítulo 11. Comportamiento del cursor en el contexto de una conexión 79

Capítulo 12. Cifrado de los datos locales . . . 81
Visión general del cifrado local de los datos . . . 81
Establecimiento de una conexión con la base de datos DB2 Everyplace 82
Cómo otorgar privilegios de cifrado a un usuario. . 83
Creación de una tabla cifrada 84
Gestión de los privilegios de cifrado 84
Cifrado utilizando el DB2eCLP 85

Capítulo 2. Cómo desarrollar aplicaciones C/C++ de DB2 Everyplace

Este capítulo contiene información sobre el modo de desarrollar aplicaciones C/C++ para DB2 Everyplace. Este capítulo contiene los apartados siguientes:

- "Cómo desarrollar aplicaciones C/C++ de DB2 Everyplace"
- "Sistemas operativos soportados de C/C++" en la página 11
- "Herramientas de desarrollo de C/C++ soportadas" en la página 10
- "Preparación, compilación y enlace de un proyecto C/C++" en la página 11
- "Cómo probar una aplicación C/C++" en la página 13
- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Cómo desarrollar aplicaciones C/C++ de DB2 Everyplace

Para desarrollar una aplicación DB2 Everyplace utilizando C/C++, se utiliza la interfaz CLI/ODBC de DB2 Everyplace. Este tema proporciona una visión general de las tareas que debe completar para desarrollar aplicaciones de C/C++ con DB2 Everyplace.

Para desarrollar aplicaciones de C/C++ con DB2 Everyplace utilizando C/C++::

1. Instale DB2 Everyplace en la estación de trabajo de desarrollo. Consulte el manual *DB2 Everyplace Guía del usuario y de instalación* para obtener instrucciones más detalladas.
2. Defina la aplicación y sus requisitos de datos.

   Determine qué datos necesitará ver o cambiar el usuario final, y cómo se recuperarán, almacenarán y actualizarán los datos en la base de datos de DB2 Everyplace.
3. Conozca la interfaz CLI de DB2 y determine las funciones de CLI de DB2 han de utilizarse en la aplicación.
4. Escriba un programa de aplicación C/C++ utilizando las funciones de CLI de DB2 soportadas en DB2 Everyplace.
5. Prepare, compile y enlace el código de aplicación con los archivos de cabecera y la biblioteca de sistema operativo de DB2 Everyplace.
6. Pruebe la aplicación:
   a. Copie las bibliotecas DB2 Everyplace en el emulador o dispositivo de su sistema operativo.
   b. Pruebe la aplicación en un dispositivo o emulador, si es apropiado.

Conceptos afines:

- Capítulo 13, "Aplicaciones C/C++ de ejemplo", en la página 93

Consulta relacionada:

- "Herramientas de desarrollo de C/C++ soportadas" en la página 10
- "Sistemas operativos soportados de C/C++" en la página 11
- "Resumen de las funciones de CLI de DB2" en la página 194

© Copyright IBM Corp. 1999,2003

Herramientas de desarrollo de C/C++ soportadas

Puede escribir una aplicación C/C++ utilizando las funciones de CLI de DB2 soportadas por DB2 Everyplace.

Las herramientas de desarrollo estándares de C/C++ soportadas para los sistemas operativos soportados incluyen:

Palm OS

Se puede utilizar:

- DB2 Everyplace Mobile Application Builder. Para obtener información acerca de Mobile Application Builder, visite el sitio Web de DB2 Everyplace en la dirección http://www.ibm.com/software/data/db2/everyplace/
- Software Developer's Kit de GNU.
- Metrowerks CodeWarrior para Palm Computing Platform. Este entorno de desarrollo comercial le permite crear programas C/C++ para el sistema operativo Palm OS utilizando una estación de trabajo Windows.

Recomendación: Registre los ID de creador de aplicación en Palm, Inc. para evitar conflictos con otras aplicaciones Palm OS.Las tablas y aplicaciones de DB2 Everyplace tienen unos ID de creador, cuyo formato es IBDB o DB2*x*, donde *x* es una letra de la ″a″ a la ″z″. Para obtener más información sobre los ID de creador, visite el sitio Web siguiente:

http://www.palmos.com/dev/

Symbian OS Versión 6.0

Puede utilizar Microsoft Visual C++, Versión 6, junto con el SDK (Software Developer's Kit) o Symbian Versión 6.0 C++ para desarrollar sus aplicaciones.

Recomendación: Obtenga los UID de Symbian OS para incluir en el archivo de proyecto. Puede conseguir estos ID de SDK o en el sitio Web siguiente:

http://www.symbian.com/developer/techlib/papers/tn_uid/uidinfo.html

Symbian OS Versión 7.0

Puede utilizar Metrowerks Codewarrior for Symbian junto con la Symbian OS Versión 7.0 SDK, para desarrollar sus aplicaciones.

Windows CE

Puede utilizar Microsoft eMbedded Visual Tools 3.0 para desarrollar sus aplicaciones.

Sistemas operativos Windows NT y Windows 2000

Puede utilizar Microsoft Visual C++ para desarrollar sus aplicaciones.

QNX Neutrino

Puede utilizar Metrowerks Codewarrior for QNX Neutrino o QNX Neutrino Software Developer's Kit (SDK) para desarrollar sus aplicaciones.

Linux Puede utilizar las herramientas de desarrollo entre plataformas de distribución de Linux incorporadas para desarrollar las aplicaciones. El kernel de Linux incorporado necesita disponer de soporte para los binarios ELD habilitados.

Conceptos afines:

- Capítulo 13, “Aplicaciones C/C++ de ejemplo”, en la página 93

Tareas afines:
- “Cómo desarrollar aplicaciones C/C++ de DB2 Everyplace” en la página 9

Sistemas operativos soportados de C/C++

La interfaz C/C++ está totalmente soportada en los sistemas operativos siguientes:
- Palm OS
- Symbian OS
- Windows CE® para Pocket PC
- Win32 (Windows 95, Windows 98, Windows NT, Windows 2000 y Windows XP)
- QNX Neutrino
- Linux y Linux incorporado

Conceptos afines:
- Capítulo 13, “Aplicaciones C/C++ de ejemplo”, en la página 93

Tareas afines:
- “Cómo desarrollar aplicaciones C/C++ de DB2 Everyplace” en la página 9

Consulta relacionada:
- “Herramientas de desarrollo de C/C++ soportadas” en la página 10
- “Resumen de las funciones de CLI de DB2” en la página 194

Preparación, compilación y enlace de un proyecto C/C++

Esta tarea forma parte de la tarea más amplia del Desarrollo de aplicaciones de DB2 Everyplace utilizando C/C++. Al completar los pasos para Preparar, compilar y enlazar un proyecto C/C++, vuelva a “Cómo desarrollar aplicaciones C/C++ de DB2 Everyplace” en la página 9.

Procedimiento:

DB2 Everyplace incluye archivos de cabecera y archivos de biblioteca del sistema operativo para el desarrollo de aplicaciones.

Siga estos pasos para preparar un archivo de proyecto y compilar y enlazar una aplicación de DB2 Everyplace utilizando el compilador correcto:

1. Cree un archivo de proyecto. Este procedimiento varía en función de las herramientas de desarrollo y del sistema operativo para el que se esté realizando el desarrollo.
2. Incluya en el proyecto los archivos de cabecera de DB2 Everyplace siguientes. Los archivos de cabecera contienen las constantes, tipos de datos y prototipos de función C/C++ que se proporcionan con DB2 Everyplace. Los archivos de cabecera son los siguientes:

   ```
   \db2everyplace\Clients\include\sqlcli.h
   \db2everyplace\Clients\include\sqlcli1.h
   \db2everyplace\Clients\include\sqlext.h
   \db2everyplace\Clients\include\sqlsystm.h
   ```
3. Incluya los archivos de cabecera específicos para su aplicación.
4. Incluya en el proyecto la biblioteca de DB2 Everyplace apropiada.

La tabla siguiente es un resumen de las bibliotecas de DB2 Everyplace y lista información adicional para cada sistema operativo.

Tabla 1. Bibliotecas de DB2 Everyplace

Sistema operativo	Archivos de biblioteca necesarios e información adicional
Palm OS	\db2everyplace\clients\palmos\database\DB2e.libOpcional: Aumente el tamaño de la pila de ejecución de programas hasta 8 KB. El valor por omisión es 4 KB. Las aplicaciones Palm OS tienen un tamaño predefinido limitado para la pila de ejecución de programas de la aplicación. Dependiendo de la aplicación, puede tener un problema de desbordamiento de pila cuando ejecute el programa. Para evitar este problema, especifique un tamaño de pila mayor en el archivo palm-pref.r, que se incluye con DB2 Everyplace.Siga las instrucciones del archivo palm-pref.r e incluya este archivo en el archivo del proyecto. Si está desarrollando una aplicación utilizando PRC-Tools, añada stack=0x8000 al archivo .def para su aplicación. Por ejemplo: application {"MyApplicationName" APID stack=0x8000 }
Symbian OS v6	Aplicaciones de emulador: \db2everyplace\clients\symbian6\database\wins\DB2e.lib Aplicaciones de dispositivo: \db2everyplace\clients\symbian6\database\armi\DB2e.lib
Symbian OS v7	Aplicaciones de emulador: \db2everyplace\clients\Symbian7\database\wins\DB2e.lib Aplicaciones de dispositivo: \db2everyplace\clients\Symbian7\database\armi\DB2e.lib
Windows CE	Procesador ARM: • V3.00 \db2everyplace\clients\wince\database\wce300\armrel\DB2e.lib • V4.00 \db2everyplace\clients\wince\database\wce400\ARM4VRel\DB2e.lib Procesador MIPS: • V3.00 \db2everyplace\clients\wince\database\wce300\mipsrel\DB2e.lib • V4.00 \db2everyplace\clients\wince\database\wce400\MIPSIVRel\DB2e.lib Procesador SH3: • V3.00 \db2everyplace\clients\wince\database\wce300\sh3rel\DB2e.lib • V4.00 \db2everyplace\clients\wince\database\wce400\SH3Rel\DB2e.lib Emulador de Windows CE: • V3.00 \db2everyplace\clients\wince\database\wce300\x86emrel\DB2e.lib (para emulador de Pocket PC) \db2everyplace\clients\wince\database\wce300\x86rel\DB2e.lib (para emulador de Pocket PC 2002) • V4.00 \db2everyplace\clients\wince\database\wce400\emulatorRel\DB2e.lib (para emulador de WinCE.NET) Compruebe que se haya habilitado UNICODE para el proyecto. Añada UNICODE y _UNICODE a la **Definición de preprocesador** de la Configuración del proyecto. Procesador XScale: • v3.00 \db2everyplace\clients\wince\database\wce300\xscale\DB2e.lib
Win32	\db2everyplace\clients\Win32\database\x86\DB2e.lib
Neutrino	libdb2e.so El archivo de biblioteca Neutrino está ubicado dentro del archivo tar en formato gzip de Neutrino de DB2 Everyplace DB2EveryplaceNTO.tar.gz. Este archivo está ubicado en el directorio \db2everyplace\clients\neutrino\database\. libdb2e.so está en el directorio /db2e/database/x86/ (para el tipo de procesador x86).
Linux	libdb2e.so El archivo de biblioteca Linux está ubicado dentro del archivo tar en formato gzip de Linux de DB2 Everyplace, DB2EveryplaceLN.tar.gz. Este archivo está ubicado en el directorio \db2everyplace\clients\embeddedlinux\database\. libdb2e.so está en los directorios /db2e/database/x86/ (para el tipo de procesador x86) y /db2e/database/strongarm/ (para el tipo de procesador strongarm).

5. Opcional: Defina la macro UNICODE y _UNICODE en el archivo del proyecto para obtener el soporte de UNICODE.

   Consulte el apartado "Soporte UNICODE en DB2 Everyplace" en la página 364 para obtener más información sobre UNICODE

6. Compile el proyecto y enlace el código de objeto con la biblioteca de DB2 Everyplace apropiada.

   Muchas de las herramientas de desarrollo de aplicaciones proporcionan la compilación y el enlace automáticos desde dentro de un entorno de desarrollo integrado. Para obtener información adicional acerca de cómo compilar y enlazar un proyecto, consulte la documentación que se incluye con el software de desarrollo de aplicaciones.

Conceptos afines:

- Capítulo 13, "Aplicaciones C/C++ de ejemplo", en la página 93

Tareas afines:

- "Cómo probar una aplicación C/C++"

Consulta relacionada:

- "Herramientas de desarrollo de C/C++ soportadas" en la página 10
- "Sistemas operativos soportados de C/C++" en la página 11
- "Resumen de las funciones de CLI de DB2" en la página 194
- "Soporte UNICODE en DB2 Everyplace" en la página 364

Cómo probar una aplicación C/C++

Esta tarea forma parte de la tarea más amplia del Desarrollo de aplicaciones de DB2 Everyplace utilizando C/C++. Al completar los pasos para probar una aplicación C/C++, vuelva a "Cómo desarrollar aplicaciones C/C++ de DB2 Everyplace" en la página 9.

Procedimiento:

Para probar una aplicación:

1. Copie las bibliotecas DB2 Everyplace en el emulador o dispositivo de su sistema operativo. Sin estos archivos, no se podrá cargar un aplicación de DB2 Everyplace. La tabla siguiente es un resumen de los archivos de DB2 Everyplace necesarios para cada sistema operativo.

Tabla 2. Archivos de DB2 Everyplace necesarios para la comprobación

Sistema operativo	Archivos necesarios en el dispositivo o emulador
Palm OS	\db2everyplace\clients\palmos\database\DB2eCat.prc \db2everyplace\clients\palmos\database\DB2eCLI.prc \db2everyplace\clients\palmos\database\DB2eComp.prc \db2everyplace\clients\palmos\database\DB2eRunTime.prc \db2everyplace\clients\palmos\database\DB2eDMS.prc
Symbian OS Versión 6.0	Para pruebas de emuladores, copie el archivo \db2everyplace\clients\symbian6\database\wins\DB2e.dll en cada uno de los directorios de emulador siguientes: \EPOCROOT%EPOC32\Release\wins\udeb\ (para un emulador de depuración) \EPOCROOT%EPOC32\Release\wins\urel\ (para un emulador de liberación) Para una comprobación en el dispositivo, instale el archivo siguiente utilizando el software de conexión PsiWin: \db2everyplace\clients\symbian6\database\armi\DB2e.sis

Tabla 2. Archivos de DB2 Everyplace necesarios para la comprobación (continuación)

Sistema operativo	Archivos necesarios en el dispositivo o emulador

Windows CE	Instale la biblioteca apropiada para su sistema operativo.
	
	Procesador ARM:
	• V3.00
	`db2everyplace\clients\wince\database\wce300\armrel\DB2e.dll`
	
	Procesador MIPS:
	• V3.00
	`\db2everyplace\clients\wince\database\wce300\mipsrel\DB2e.dll`
	
	Procesador SH3:
	• V3.00
	`\db2everyplace\clients\wince\database\wce300\sh3rel\DB2e.dll`
	
	Emulador de Windows CE:
	• V3.00
	Para un emulador de Pocket PC:
	`db2everyplace\clients\wince\database\wce300\x86emrel\DB2e.dll`
	
	Para un emulador de Pocket PC 2002:
	`\db2everyplace\clients\wince\database\wce300\x86rel\DB2e.dll`

Win32	Copie `\db2everyplace\clients\win32\database\x86\DB2e.dll` en el directorio actual de la aplicación en la variable de entorno PATH del sistema.
Neutrino	`/db2e/database/x86/libdb2e.so (para el tipo de procesador x86)` y `/db2e/database/strongarm/libdb2e.so (para el tipo de procesador strongarm)`
Linux	`/db2e/database/x86/libdb2e.so (para el tipo de procesador x86)` y `/db2e/database/strongarm/libdb2e.so (para el tipo de procesador strongarm)`

2. **Para Linux y Neutrino**: Añada `libdb2e.so` a la vía de acceso de búsqueda de biblioteca, utilizando uno de los métodos siguientes:
   - Copie `libdb2e.so` en un directorio que esté en la vía de acceso de búsqueda de biblioteca. Es posible que esta acción requiera permisos root.
   - Copie `libdb2e.so` en otro directorio y añada dicho directorio a la vía de acceso de búsqueda de biblioteca. Añadir un directorio a la vía de acceso de búsqueda de biblioteca de modo permanente requiere una entrada en `/etc/ld.config`. Añadir temporalmente un directorio a la vía de acceso de búsqueda de biblioteca puede efectuarse estableciendo la variable de entorno LD_LIBRARY_PATH del modo apropiado.

     Por ejemplo, escriba el mandato siguiente (bash, con `libdb2e.so` en el directorio actual): `export LD_LIBRARY_PATH=`
3. Cargue los archivos de la aplicación que está probando. Por ejemplo, para probar la aplicación de ejemplo Visiting Nurse en Palm OS, cargue los archivos `NurseInit.prc` y `Nurse.prc`.
4. Pruebe la aplicación.

Conceptos afines:

• Capítulo 13, “Aplicaciones C/C++ de ejemplo”, en la página 93

Tareas afines:

• “Preparación, compilación y enlace de un proyecto C/C++” en la página 11

Consulta relacionada:

- "Herramientas de desarrollo de C/C++ soportadas" en la página 10
- "Sistemas operativos soportados de C/C++" en la página 11
- "Resumen de las funciones de CLI de DB2" en la página 194

Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++

Este tema proporciona una visión general sobre cómo desarrollar aplicaciones DB2 Everyplace Sync Client con C/C++ basándose en la API C de IBM Sync Client para la Versión 8.1. El "Resumen de funciones de la API C de IBM Sync Client" en la página 317 proporciona las especificaciones de todas las funciones d la API C

Requisitos previos:

Instale DB2 Everyplace en la estación de trabajo de desarrollo. Consulte el manual *DB2 Everyplace Guía del usuario y de instalación* para obtener más detalles.

Procedimiento:

Para desarrollar una aplicación DB2 Everyplace Sync Client utilizando C/C++:

1. Definir la aplicación de sincronización incluyendo:
   - los datos que sincronizará;
   - las operaciones permitidas;
   - los usuarios y grupos de usuarios;
   - la seguridad de los datos (por ejemplo, el cifrado de datos sobre el cifrado de datos locales y por cable)

     Consulte la publicación *DB2 Everyplace Sync Server Administration Guide* para conocer más detalles acerca de la definición de los datos que se deben sincronizar y la administración de los usuarios.
2. Incluir el archivo de cabecera de DB2 Everyplace Sync Client (`isyncore.h`) en los programas de aplicación C, y utilizar las funciones de la API C de DB2 Everyplace Sync Client siguiendo las especificaciones de las mismas.
3. Preparar, compilar y enlazar el código de la aplicación con las bibliotecas del sistema operativo de DB2 Everyplace Sync Client, `isyncconf` e `isyncore`.
4. Probar la aplicación:
   - Instale las bibliotecas DB2 Everyplace en el emulador o dispositivo de su sistema operativo.
   - Pruebe la aplicación en un emulador, si es apropiado.
   - Pruebe la aplicación en un dispositivo.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Consulta relacionada:

- "Herramientas de desarrollo de C/C++ soportadas" en la página 10

Capítulo 3. Cómo desarrollar aplicaciones Java de DB2 Everyplace

Este capítulo describe el modo de desarrollar aplicaciones Java de DB2 Everyplace. Los temas que se tratan son:

- "Sistemas operativos soportados de la interfaz JDBC"
- "Desarrollo de aplicaciones Java de DB2 Everyplace"

Sistemas operativos soportados de la interfaz JDBC

La interfaz JDBC se soporta en los sistemas operativos siguientes:

- Palm OS
- Symbian OS
- Windows CE® para Pocket PC
- Win32 (Windows 95, Windows 98, Windows NT, Windows 2000 y Windows XP)
- QNX Neutrino
- Linux y Linux incorporado

Desarrollo de aplicaciones Java de DB2 Everyplace

Para desarrollar una aplicación de DB2 Everyplace utilizando Java, puede utilizar el Software Developer's Kit de Java junto con la interfaz JDBC (Java Database Connectivity) de DB2 Everyplace para Java.

Este tema proporciona una visión general de alto nivel de las tareas necesarias para desarrollar aplicaciones Java con DB2 Everyplace.

Restricciones:

DB2 Everyplace no da soporte a la ejecución de varias tareas en Symbian. Para poder acceder a una base de datos desde un segundo paso, el objeto de Conexión del primer paso se debe cerrar para que se pueda establecer la conexión en el segundo paso. Distintos pasos no pueden compartir el mismo objeto de Conexión.

Requisitos previos:

Una aplicación Java que accede a DB2 Everyplace utiliza el controlador JDBC de DB2 Everyplace. Si todavía no lo ha hecho, instale Java y JDBC en la estación de trabajo.

Procedimiento:

Para desarrollar aplicaciones de DB2 Everyplace utilizando Java:

1. Importe el paquete `java.sql` y cualquier otra clase de Java que sea necesaria.
2. Cargue el controlador JDBC de DB2 Everyplace. El nombre de clase es `com.ibm.db2e.jdbc.DB2eDriver`.
3. Conéctese con la base de datos utilizando un URL con formato `jdbc:`*`subprotocolo:subnombre`*. El *subprotocolo* de DB2 Everyplace es `db2e`. Si la

© Copyright IBM Corp. 1999,2003

base de datos se encuentra en `c:\dir1\dir2`, utilice el URL `jdbc:db2e:c:/dir1/dir2/`. También puede utilizar una vía de acceso relativa para *`subnombre`*.

4. Crear un objeto de Sentencia.
5. Acceder a la base de datos (la lógica de la aplicación va aquí):
   - Ejecutar una sentencia de SQL utilizando el objeto de Sentencia.
   - Recuperar datos del objeto de ConjuntoResultados devuelto (si la sentencia de SQL que ha ejecutado es una consulta).
6. Liberar recursos de base de datos y JDBC cerrando los objetos de ConjuntoResultados, Sentencia y Conexión.

Conceptos afines:

- "Visión general de las aplicaciones Java de ejemplo" en la página 95

Tareas afines:

- "Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS" en la página 97
- "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102

Consulta relacionada:

- "Visión general del soporte de JDBC de DB2 Everyplace" en la página 287

Capítulo 4. Cómo desarrollar aplicaciones Java Sync Client

Este capítulo describe el modo de desarrollar aplicaciones Java de Cliente de sincronización de DB2 Everyplace. Los temas que se tratan son:

- "Sistemas operativos soportados por la API de Java Sync"
- "Las API de IBM Java Sync"

Sistemas operativos soportados por la API de Java Sync

Las API de Java Sync están disponibles en los sistemas operativos siguientes:

- Win32
- Symbian OS
- Windows CE (con procesadores MIPS y ARM)
- Palm OS
- Linux
- QNX Neutrino

Las API de IBM Java Sync

Puede crear aplicaciones Java utilizando Java Database Connectivity (JDBC) y la interfaz de Java a fin de integrar el funcionamiento de DB2 Everyplace Database y Sync Server.

Para obtener información detallada sobre las interfaces, clases y excepciones que se suministran con las API de IBM Java Sync soportadas por DB2 Everyplace, consulte la documentación de Javadoc del directorio `Clients\javadoc`.

Conceptos afines:

- "Visión general de las aplicaciones Java de ejemplo" en la página 95

Tareas afines:

- "Desarrollo de aplicaciones Java de DB2 Everyplace" en la página 17

Visión general de los proveedores de sincronización de DB2 Everyplace

En este tema se describe la API de Java Sync Client soportada por DB2 Everyplace. La API es un conjunto de bibliotecas que permiten que los programadores creen aplicaciones que sincronicen los datos de forma bidireccional entre DB2 Everyplace y bases de datos relacionales empresariales. Funciona junto con el DB2 Everyplace Sync Server para simplificar la sincronización de archivos y datos relacionales. El Sync Server proporciona resolución de conflictos y gestiona el movimiento de datos al dispositivo PDA portátil, incorporado o habilitado para MIDP 1.0.

La API de Sync Client Java consta de dos tipos de proveedores de sincronización:

- "Sincronización nativa de DB2 Everyplace" en la página 20
- "Sincronización de Java de DB2 Everyplace" en la página 27

© Copyright IBM Corp. 1999,2003

En los archivos de ejemplo se brinda información sobre cómo crear aplicaciones Java en el dispositivo cliente basándose en estos proveedores.

Tareas afines:

- "Instalación del proveedor de sincronización nativa basado en JNI" en la página 21
- "Instalación y verificación del proveedor de sincronización nativa basada en la detección" en la página 24

Conceptos afines:

- "Las aplicaciones de sincronización nativa de Java de ejemplo" en la página 121
- "Las aplicaciones de sincronización MIDP de Java de ejemplo" en la página 125

Sincronización nativa de DB2 Everyplace

:

Los proveedores de sincronización nativa proporcionan la interfaz Java que invoca a las bibliotecas del cliente de sincronización nativa.

Nota: Los proveedores de sincronización nativa no soportan la seguridad de hebras en este release, la coordinación de la sincronización de hebras es responsabilidad de la aplicación.

Hay dos tipos de proveedores de sincronización nativa de DB2 Everyplace:

- Proveedor de sincronización nativa basado en Java Native Interface (JNI)
- Proveedor de sincronización nativa basada en desvío de Palm OS

Tareas afines:

- "Instalación del proveedor de sincronización nativa basado en JNI" en la página 21
- "Instalación y verificación del proveedor de sincronización nativa basada en la detección" en la página 24

Conceptos afines:

- "Visión general de los proveedores de sincronización de DB2 Everyplace" en la página 19

Instalación de proveedores de sincronización nativa de DB2 Everyplace

Este capítulo describe el modo de instalar proveedores de sincronización nativa de DB2 Everyplace. Los temas que se tratan son:

- "Instalación del proveedor de sincronización nativa basado en JNI" en la página 21
- "Instalación y verificación del proveedor de sincronización nativa basada en la detección" en la página 24

Instalación del proveedor de sincronización nativa basado en JNI

El proveedor de sincronización JNI funciona con una Java VM que soporta la Java Native Interface. .

Se soporta este proveedor en los sistemas operativos siguientes:

- Win32
- Symbian Release 6 (para dispositivos Nokia 9210/9290 Communicator)
- Symbian Release 7 (para dispositivos Sony Ericsson P800)
- Windows CE (para dispositivos Pocket PC)
- Linux
- QNX Neutrino

Requisitos previos:

El proveedor de sincronización basado en JNI requiere los archivos siguientes:

- el archivo `isync4j.jar`
- los binarios de cliente de sincronización nativa siguientes:
  - `isyncore.dll`
  - `isyncconf.dll`
  - `imsadb2e.dll`
  - `imsafile.dll`
  - `imsaconfig.dll`
  - `wbxmllib.dll`
  - `isync4j.dll`
  - `isyncxpt.dll`

Si la aplicación está utilizando el proveedor de sincronización nativa basada en JNI, debe importar los paquetes isync4j de Java siguientes:

- `com.ibm.mobileservices.isync`
- `com.ibm.mobileservices.isync.event`
- `com.ibm.mobileservices.isync.sql`

Verifique si en el sistema está instalado el software siguiente:

- DB2 Everyplace Sync Server Versión 8
- DB2 Everyplace Sync Client Libraries Versión 8

  Consulte el manual *DB2 Everyplace Guía del usuario y de instalación* para obtener más información.
- Una Java VM que soporte la Java Native Interface

Lea los temas siguientes para obtener más información sobre el modo de instalar el proveedor de sincronización basada en JNI en cada uno de los sistemas operativos soportados:

- “Instalación del proveedor de sincronización basado en JNI” en la página 22
- “Instalación del proveedor de sincronización basado en JNI en dispositivos Nokia 9210/9290 Communicator utilizando Symbian V6” en la página 22
- “Instalación del proveedor de sincronización basado en JNI en Windows CE” en la página 23

Instalación del proveedor de sincronización basado en JNI

Para instalar el proveedor de sincronización basado en JNI de un sistema operativo Win32, deberá compilar y ejecutar el programa ISyncSample. Las implementaciones basadas en JNI para dispositivos Win32 se han comprobado en Sun MicroSystems Java™ VM y en el IBM Java™ 2 Standard Edition Developer Kit .

Procedimiento:

1. Compile el programa ISyncSample.
   a. Cambie la variable del sistema PATH de forma que incluya los directorios siguientes:

      ```
      <DB2e_InstDir>\Clients\Win32\database\x86
      <DB2e_InstDir>\Clients\Win32\sync
      ```

   b. Cambie la variable CLASSPATH de forma que incluya el archivo `isync4j.jar`:

      ```
      <DB2e_InstDir>\Clients\Win32\Sync\isync4j.jar
      ```

   c. Compile los archivos de ejemplo incluidos en el directorio `<DB2e_InstDir>\Clients\clientapisample\Java_API`. Por ejemplo:

      ```
      javac ISyncSample.java
      ```

2. Edite el archivo `isyncdb2e.properties` para especificar la contraseña, usuario y URL del servidor.
3. Ejecute el programa ISyncSample.
   a. Escriba el mandato siguiente:

      ```
      java.exe ISyncSample <archivo de propiedades>
      ```

      donde *<archivo de propiedades>* es el archivo de propiedades para la base de datos del cliente. Por ejemplo:

      ```
      java.exe -classpath .; isync4j.jar ISyncSample isyncdb2e.properties
      ```

Tareas afines:

- "Instalación del proveedor de sincronización nativa basado en JNI" en la página 21

Instalación del proveedor de sincronización basado en JNI en dispositivos Nokia 9210/9290 Communicator utilizando Symbian V6

Para instalar el proveedor de sincronización basado en JNI en dispositivos Nokia 9210/9290 Communicator utilizando Symbian V6, deberá compilar y ejecutar el programa ISyncSample. Las implementaciones basadas en JNI para dispositivos Nokia 9210/9290 utilizando Symbian V6 se han comprobado en Symbian OS 6.0 PersonalJava JVM.

Procedimiento:

1. Edite y compile el programa ISyncSample en su estación de trabajo.
   a. Edite `ISyncSample.java` para adoptar `isyncdb2e.properties` como parámetro.
   b. Compile `ISyncSample.java` con `isync4j.jar` en la vía de acceso de clase escribiendo el mandato siguiente:

      ```
      javac -classpath isync4j.jar ISyncSample.java
      ```

   c. Edite `isyncdb2e.properties` para especificar el URL de servidor, usuario y contraseña.
2. Ejecute el programa ISyncSample.

a. Asegúrese de que las bibliotecas de base de datos de DB2 Everyplace y de Cliente de sincronización están instaladas en el dispositivo.
b. Copie los archivos `ISyncSample.class` y `isyncdb2e.properties` en el directorio `C:\System\Apps\ISync` del dispositivo.
c. Utilizando el Administrador de archivos de Windows, localice y seleccione el archivo `isync4j.jar`. Pulse **Intro**. Utilice el programa Redirect, que está instalado en el dispositivo Nokia, para transferir la salida del programa Java y luego visualizar dicha salida en la consola o grabarla en un archivo.

Consulta relacionada:

- "Subconjuntos JSP Versión 1.1 soportados" en la página 45
- "Códigos de personalización de IBM para acceder a bases de datos de aplicación de JSP" en la página 48

Instalación del proveedor de sincronización basado en JNI en Windows CE

Para instalar el proveedor de sincronización basado en JNI en sistemas operativos Windows CE, deberá compilar y ejecutar el programa ISyncSample. El Sync Provider basado en JNI para dispositivos Windows CE se ha comprobado en OTI J9 JVM.

Procedimiento:

1. Compile el programa ISyncSample en su estación de trabajo.
   a. Escriba el mandato siguiente para compilar `ISyncSample.java` con `isync4j.jar` en la vía de acceso de clase:

      ```
      javac -classpath isync4j.jar ISyncSample.java
      ```

   b. Edite `isyncdb2e.properties` para especificar el URL de servidor, usuario y contraseña.
2. Ejecute el programa ISyncSample.
   a. Verifique que el entorno de ejecución de J9 Java Virtual Machine (JVM) está instalado en el dispositivo (por ejemplo `\wsdd`). Además, deben estar instaladas las bibliotecas de DB2 Everyplace y de Cliente de sincronización.
   b. Copie los archivos `ISyncSample.class` y `isyncdb2e.properties` en el dispositivo (por ejemplo, `\`).
   c. Utilice uno de los dos métodos siguientes para invocar al programa ISyncSample con `isync4j.jar` en CLASSPATH.

      Consola de Java

      Escriba el mandato siguiente:

      ```
      j9.exe -bp:\wsdd\classes.zip -cp:\wsdd;\Windows\isync4j.jar ISyncSample
        <archivo de propiedades>
      ```

      Por ejemplo:

      ```
      j9.exe -bp:\wsdd\classes.zip -cp:\wsdd;\Windows\isync4j.jar
       ISyncSample isyncdb2e.properties
      ```

      Atajo de Windows

      Cree y edite un atajo de Windows denominado `ISyncSample.lnk` en la estación de trabajo. Por ejemplo:

      ```
      255#"\wsdd\j9.exe" "-bp:\wsdd;\Windows\isync4j.jar;\wsdd\classes.zip"
      "ISyncSample" "isyncdb2e.properties"
      ```

Entre el atajo en una sola línea y encierre cada uno de los campos entre comillas dobles. El primer campo que escriba *debe* ser el nombre del ejecutable. Los archivos y directorios que especifique tienen que estar completamente calificados.

d. Ejecute el programa de ejemplo y verifique que los datos sincronizados residen en el directorio destino tal y como se ha especificado en el archivo de propiedades.

Tareas afines:

- "Instalación del proveedor de sincronización nativa basado en JNI" en la página 21

Instalación y verificación del proveedor de sincronización nativa basada en la detección

El proveedor de sincronización nativo basado en la detección sólo se utiliza con WebSphere Studio Device Developer J9 JVM en plataformas Palm OS.

En este tema se describe cómo se puede utilizar isync4j de DB2 Everyplace para Palm OS con la configuración jclMidp de la J9 (J2ME MIDP). Este proveedor de sincronización hace referencia al paquete com.ibm.oti.palmos, por lo que sólo se ejecutará en WSDD J9 JVM para PalmOS v1.5 o superiores.

La tabla siguiente describe el lugar en el que están ubicados los programas que se utilizan para instalar la API en dispositivos Palm, donde %DSYINSTDIR% representa el directorio de instalación para DB2 Everyplace.

Directorio	Descripción
`%DSYINSTDIR%/Clients/PalmOS/Sync/isync4j-palm/lib`	La carpeta que contiene isync4j para clases Java de Palm OS. Estas clases se importan durante la implementación.
`%DSYINSTDIR%/Clients/PalmOS/Sync/isync4j-palm/sample`	La carpeta que contiene el código fuente para la aplicación isync4j de ejemplo.
`%DSYINSTDIR%/Clients/PalmOS/Sync/isync4j-palm/bin/ISyncSample.prc`	La aplicación isync4j de ejemplo utilizada con la biblioteca CLDC de J9 Palm OS.

Requisitos previos:

El proveedor de sincronización nativo basado en la detección requiere las siguientes bibliotecas compartidas nativas de Cliente de sincronización y las bibliotecas de DB2 Everyplace:

- `isyncore.prc`
- `isyncconf.prc`
- `imsaconfig.prc`
- `imsafile.prc`
- `imsadb2e.prc`
- `wbxmllib.prc`
- `isyncxpt.prc`

Además, es necesario instalar los binarios de JVM de J9 Palm OS en el dispositivo.

Si la aplicación está utilizando el proveedor de sincronización nativa basada en la detección, debe importar los paquetes isync4j de Java siguientes:

- `com.ibm.mobileservices.isync`
- `com.ibm.mobileservices.isync.db2e.sti`

- com.ibm.mobileservices.isync.event
- com.ibm.mobileservices.isync.sql

Verifique si en el sistema está instalado el software siguiente:
- Palm OS Versión 3.5 o posteriores (con un mínimo de ocho MB de memoria)
- WebSphere Studio Device Developer (WSDD) Versión 4.0
- base de datos DB2 Everyplace para Palm OS Versión 7.1 o posteriores
- Bibliotecas de DB2 Everyplace Sync Client Versión 8.1 o posteriores

Después de haber instalado WSDD, debe configurar un destino de Palm OS. Para configurar un destino de Palm OS, consulte el capítulo titulado ″Getting Started with Palm OS Targets″ de la publicación WSDD Development Environment & Tools Product Documentation. La documentación de WSDD está ubicada en el CD-ROM del producto en IBM\wsdd\wsdd4.0\doc\wsddCustomer.pdf. Finalmente, verifique que WSDD está bien instalado creando y ejecutando una aplicación de ejemplo de WSDD.

Procedimiento:

Para comprobar si WSDD se ha instalado debidamente:
1. Cree un nuevo proyecto para la aplicación de ejemplo de isync4j:
   a. Abra Java Perspective en WSDD.
   b. Seleccione **File –> New–> Other**.
   c. Seleccione el asistente para J2ME para J9 y Create MIDlet Suite.
   d. Asigne un nombre al proyecto personalizado, nombre de MIDlet, y nombre de clase de MIDlet en el diálogo MIDlet Suite Creation.
   e. Pulse **Next**.
   f. Pulse de nuevo **Next** para ir a Java Settings.
   g. En Java Settings, pulse la pestaña **Libraries** y pulse **Create Folder...**. Escriba **lib** en el diálogo New Class Folder.
   h. Pulse **Finish**.
2. Importe las clases Java ISYNC4J de DB2 Everyplace y configure la vía de acceso de construcción.
   a. Pulse el proyecto en la vista **Packages** y luego el elemento de menú **File->Import...**
   b. Importe la carpeta %DSYINSTDIR%/Clients/PalmOS/Sync/isync4j-palm/lib y seleccione %DSYINSTDIR%/Clients/PalmOS/Sync/isync4j-palm/lib como directorio de origen.
   c. Expanda el directorio lib y seleccione el recuadro de selección para el directorio com bajo /lib. Bajo **Seleccione el destino para recursos importados:** teclee el nombre del proyecto seguido por /lib en el campo **Carpeta:**. Por ejemplo, si el nombre del proyecto es ISyncSample, el campo debe contener ISyncSample/lib.
   d. Pulse **Finish**.
   e. Expanda la carpeta lib; deberá ver los paquetes ISYNC4J de Java siguientes:

      com.ibm.mobileservices.isync

      com.ibm.mobileservices.isync.db2e.sti

      com.ibm.mobileservices.isync.event

      com.ibm.mobileservices.isync.sql

3. Verifique la configuración de la biblioteca isync4j creando y ejecutando la aplicación de ejemplo.
   a. Importe la aplicación de ejemplo.
      - Pulse la carpeta src para el proyecto en la vista **Packages** y después pulse **File >Import** en el menú principal.
      - Importe ISyncSample.java. Seleccione `%DSYINSTDIR%/Clients/PalmOS/Sync/isync4j-palm/samples/ISyncSample/` como directorio de origen y seleccione el recuadro de selección para ISyncSample.java. Verifique que el destino de los recursos importados es <proyecto>/src.).
   b. Cree un archivo de construcción para la aplicación de ejemplo.
      - En el editor, pulse la pestaña **in/exclusion** y luego **New**.
      - Entre **ISyncSample** como clase principal y seleccione **J9 para Palm 68k** como plataforma. Pulse **Next**.
      - Entre el Creator id, e ISyncSample como App Name. Pulse **Next** dos veces.
      - Seleccione **Prc Application on PalmOS emulator**. Pulse **Finish**.
   c. Modifique el archivo ISyncSample.jxeLinkOptions.
      - Expanda la carpeta **palm68k** para el proyecto en la vista **Packages**.
      - Realice una doble pulsación sobre `ISyncSample.jxeLinkOptions`.
      - En el editor, pulse la pestaña **in/exclusion** y **New**.
      - Entre **com.ibm.mobileservices.isync.db2e.sti.DB2eISyncProvider** como Rule pattern, y luego pulse **OK**.
      - En el editor, pulse la pestaña **source**
      - Escriba **-vmOption -ms:15** para establecer el tamaño de pila.
      - Guarde los cambios efectuados.
   d. Ejecute la aplicación de ejemplo.
      - Pulse el icono **Run** del menú y seleccione **Run –>Build –>Launch** en el archivo de construcción.
      - Seleccione el destino para la aplicación de ejemplo y pulse **Finish**.
      - Si no hay ningún error, el emulador de Palm OS debe comenzar y ejecutar la aplicación.

Ahora puede crear su propia aplicación. Cuando cree una aplicación nueva, incluya un nombre de proyecto nuevo para isync4j de DB2 Everyplace en la vía de acceso de construcción del proyecto. Una vez que cree un archivo de construcción para la aplicación, modifique el archivo `jxeLinkOptions` del mismo de forma que se ajuste a las necesidades de la aplicación.

Conceptos afines:

- "Visión general de los proveedores de sincronización de DB2 Everyplace" en la página 19

Proveedores de sincronización de DB2 Everyplace Java

Este capítulo describe los proveedores de sincronización de DB2 Everyplace Java. Los temas que se tratan son:

- "Sincronización de Java de DB2 Everyplace" en la página 27
- "Sincronización J2ME MIDP de DB2 Everyplace" en la página 27
- "Cliente DB2 Everyplace Java Sync para Cloudscape" en la página 28

Sincronización de Java de DB2 Everyplace

:

Los proveedores de sincronización de Java proporcionan la interfaz Java que invoca a las bibliotecas del Java Sync Client.

Hay dos tipos de proveedores de sincronización de Java de DB2 Everyplace:

- "Sincronización J2ME MIDP de DB2 Everyplace"
- "Cliente DB2 Everyplace Java Sync para Cloudscape" en la página 28

Tareas afines:

- "Instalación del proveedor de sincronización nativa basado en JNI" en la página 21
- "Instalación y verificación del proveedor de sincronización nativa basada en la detección" en la página 24

Conceptos afines:

- "Visión general de los proveedores de sincronización de DB2 Everyplace" en la página 19

Sincronización J2ME MIDP de DB2 Everyplace

El J2ME MIDP ISync Client permite crear aplicaciones que sincronizan suscripciones con el MIDP Record Store Management System (RMS). El J2ME MIDP ISync Client es un conjunto de bibliotecas que colaboran con el DB2 Everyplace Sync Server para simplificar la sincronización de datos relacionales entre bases de datos empresariales y dispositivos habilitados para MIDP 1.0. El Sync Server gestiona el movimiento de datos a y desde el dispositivo MIDP.

Este tema incluye la información siguiente sobre el J2ME MIDP ISync Client:

- Software de servidor Web necesario para instalar el J2ME MIDP ISync Client
- Software y hardware necesario para ejecutar el J2ME MIDP ISync Client en teléfonos Motorola iDEN
- Diseño del directorio de instalación de J2ME MIDP ISync Client

Software de servidor Web necesario para instalar el J2ME MIDP ISync Client:

Para poder instalar el J2ME MIDP Sync Client, se necesita uno de los dos productos de software siguientes:

- WebSphere Application Server, Advanced Single Server Edition Versión 4.x o posteriores. Puede bajar una versión libre de este software del sitio Web de IBM en la dirección http://www-3.ibm.com/software/webservers/appserv/advanced.html.
- Apache Tomcat Versión 4.0.x o posteriores. Puede bajar una copia libre de este software de la dirección http://jakarta.apache.org/tomcat/.

Software y hardware necesario para ejecutar el J2ME MIDP ISync Client en teléfonos Motorola iDEN:

Para instalar y ejecutar el proveedor de sincronización MIDP en teléfonos Motorola iDEN, se necesitan el hardware y el software siguientes:

- Sun Microsystems JavaTM 2 Platform Micro Edition, Wireless Toolkit

- iDEN Update y cable de datos (para cargar aplicaciones en el teléfono)
- Apache ANT
- RetroGuard Ofuscator
- Paquetes Java isync4j (incluidos con DB2 Everyplace)
  - com.ibm.mobileservices.isync
  - com.ibm.mobileservices.isync.midp
  - com.ibm.mobileservices.isync.event

Diseño del directorio de instalación de J2ME MIDP ISync Client:

El proceso de instalación de J2ME MIDP ISync Client crea cuatro directorios iniciales:

- `bin` - `%DSYINSTDIR%\Clients\Midp\bin`, contiene un script para ejecutar el emulador WTK desde la línea de mandatos.
- `lib` - `%DSYINSTDIR%\Clients\Midp\lib` contiene los jar de la API de MIDP ISync, el Servlet para MIDP, el archivo FilterServlet.jar y MIDLets de ejemplo con los archivos JAD asociados.
- `docs` - `%DSYINSTDIR%\Clients\Midp\doc` contiene el J2ME MIDP ISync Client Javadoc.
- `samples` - `%DSYINSTDIR%\Clients\Midp\samples`, contiene el código fuente para la aplicación isync4j de ejemplo, donde `%DSYINSTDIR%` es el directorio de instalación de DB2 Everyplace.

Si decide volver a compilar los ejemplos, encontrará varios directorios build*Classes que se utilizan a efectos de verificación previa y ofuscación.

Tareas afines:

- "Instalación del proveedor de sincronización nativa basado en JNI" en la página 21
- "Instalación y verificación del proveedor de sincronización nativa basada en la detección" en la página 24

Conceptos afines:

- "Visión general de los proveedores de sincronización de DB2 Everyplace" en la página 19

Cliente DB2 Everyplace Java Sync para Cloudscape

El Cliente DB2 Everyplace Java Sync para Cloudscape permite crear aplicaciones que sincronizan suscripciones con una base de datos Cloudscape. El Cliente Java Sync para Cloudscape es un conjunto de bibliotecas que colaboran con el DB2 Everyplace Java Sync Server para simplificar la sincronización de datos relacionales entre bases de datos empresariales y un cliente de Cloudscape. El Sync Server gestiona el movimiento de datos a y desde el dispositivo.

Este tema incluye la información siguiente sobre el Cliente Java Sync para Cloudscape:

- Software necesario para ejecutar el Cliente Java Sync para Cloudscape
- Cliente Java Sync para el diseño del directorio de Cloudscape
- Cómo establecer la variable de entorno CLASSPATH

Software necesario para ejecutar el Cliente Java Sync para Cloudscape:

Para ejecutar el Cliente Java Sync para Cloudscape, necesitará los siguientes productos de software:

- DB2 Everyplace versión 8.1.4 o posterior
- La base de datos de Cloudscape

Cliente Java Sync para el diseño del directorio de Cloudscape:

Los archivos de Cliente de Java Sync para archivos de Cloudscape están en los directorios siguientes:

- `%DSYINSTDIR%\Clients\javaclient` contiene el archivo jar de la API de Cloudscape ISync.
- `%DSYINSTDIR%\doc\javadoc\javaclient` contiene el Cliente de Java Sync para Cloudscape Javadoc.

Cómo establecer la variable de entorno CLASSPATH:

Para utilizar el Cliente de Java Sync para Cloudscape, establezca la variable de entorno CLASSPATH para que incluya los archivos siguientes:

- Los archivos jar de Cloudscape (`db2j.jar`, `db2jtools.jar`) de la instalación de Cloudscape.
- El archivo jar de la API de Cloudscape ISync (`db2jisync.jar`).
- Opcional: Las aplicaciones de ejemplo (ISyncSample y GoISyncConsole).

Tareas afines:

- "Compilación y ejecución de la aplicación de sincronización de Java de ejemplo GoISyncConsole" en la página 132

Capítulo 5. Cómo desarrollar aplicaciones de Visual Basic

Este capítulo proporciona información sobre el desarrollo de aplicaciones DB2 Everyplace utilizando Visual Basic. Los temas que se tratan son:

- "Cómo desarrollar aplicaciones de Visual Basic de DB2 Everyplace"
- "Sistemas operativos soportados de la interfaz de Visual Basic" en la página 32

Cómo desarrollar aplicaciones de Visual Basic de DB2 Everyplace

Para desarrollar una aplicación DB2 Everyplace en Visual Basic, se utiliza la interfaz CLI/ODBC de DB2 Everyplace. Este tema proporciona una visión general de alto nivel de las tareas que debe completar para desarrollar aplicaciones de Visual Basic con DB2 Everyplace.

Restricciones:

Cuando desarrolle aplicaciones para DB2 Everyplace utilizando Visual Basic, tenga en cuenta las restricciones y los requisitos siguientes:

- No utilice directamente la función SQLAllocHandleVer en el código de su aplicación. SQLAllocHandleVer es utilizado por DB2 Everyplace internamente. Si lo utiliza en su código de aplicación, puede provocar errores de programa.
- La depuración puede que no funcione debido a la forma en que Visual Basic carga y maneja las llamadas a funciones dentro de una DLL.
- Las funciones Visual Basic que invocan funciones DB2 Everyplace contenidas en `db2e.dll` deben tener la sentencia ″On Error Resume Next″, de lo contrario el programa no funcionará debidamente.

Procedimiento:

Los pasos básicos para desarrollar una aplicación Visual Basic de DB2 Everyplace son:

1. Cree un nuevo proyecto Visual Basic.
2. Copie el archivo `db2ecli.bas` (del directorio de proyectos de DB2 Everyplace para Visual Basic) en la carpeta de proyectos e inserte el archivo en el proyecto Visual Basic.
3. Copie `DB2e.dll` en la carpeta de proyectos. Si no desea colocar `DB2e.dll` en la carpeta de proyectos, modifique la vía de acceso de `DB2e.dll` en las declaraciones de función del archivo `db2ecli.bas`.
4. Escriba su propio código de aplicación. Puede utilizar como guía el programa de ejemplo de Visual Basic de DB2 Everyplace.
5. Cree el programa ejecutable de la aplicación; para ello seleccione la opción de menú **Archivo –> Crear** *proyecto*.

Conceptos afines:

- "Visión general de la aplicación Visual Basic de ejemplo" en la página 111

Consulta relacionada:

- "Resumen de las funciones de CLI de DB2" en la página 194
- "Sistemas operativos soportados de la interfaz de Visual Basic" en la página 32

© Copyright IBM Corp. 1999,2003

Sistemas operativos soportados de la interfaz de Visual Basic

La interfaz Visual Basic está totalmente soportada en los sistemas operativos siguientes:

- Windows CE® para Pocket PC
- Win32 (Windows 95, Windows 98, Windows NT, Windows 2000 y Windows XP)

Capítulo 6. Desarrollo de aplicaciones JSP

Este capítulo proporciona información sobre el desarrollo de aplicaciones DB2 Everyplace utilizando JavaServer Pages. Los temas que se tratan son:

- "Soporte de JSP para los sistemas operativos"
- "Desarrollo de aplicaciones JSP de DB2 Everyplace"
- "Visión general del soporte de JSP en DB2 Everyplace" en la página 34

Soporte de JSP para los sistemas operativos

Se dispone de soporte de JSP para los sistemas operativos siguientes:

- Win32 (Windows® NT® y Windows® 2000®)
- Windows CE® para Pocket PC
- Symbian OS

Tareas afines:

- "Verificación del soporte de JSP en una estación de trabajo Windows" en la página 35
- "Configuración del desarrollo de JSP en un dispositivo Windows CE" en la página 36

Conceptos afines:

- "Visión general del soporte de JSP en DB2 Everyplace" en la página 34
- Capítulo 16, "Las aplicaciones JSP de ejemplo", en la página 117

Desarrollo de aplicaciones JSP de DB2 Everyplace

DB2 Everyplace da soporte a JavaServer Pages (JSP) para permitir que se creen fácilmente aplicaciones DB2 Everyplace basadas en la Web. Las aplicaciones que se crean son independientes del sistema operativo y se pueden ejecutar en dispositivos portátiles en modalidad desconectada o mientras se está desconectado de una Red de área local.

La tecnología JSP proporciona un método rápido y simplificado para desarrollar y mantener páginas Web dinámicas. La tecnología JSP separa la interfaz de usuario de la generación de contenido, de forma que se pueden crear y actualizar diseños de páginas sin cambiar el contenido dinámico subyacente.

JSP utiliza tecnología JDBC. Las aplicaciones basadas en la Web que se crean utilizando JSP pueden acceder a bases de datos de DB2 Everyplace a través del Controlador JDBC de DB2 Everyplace.

Consulte la documentación que acompaña a `<DB2Everyplace>\SDK\JSP\doc` para obtener más información sobre el modo de generar páginas JSP que accedan a DB2 Everyplace utilizando WebSphere Studio.

Pruebe la aplicación JSP siguiendo los pasos de la sección "Ejecución de una aplicación JSP en una estación de trabajo Windows" en la página 43. Debería ejecutar siempre la aplicación JSP en la estación de trabajo antes de transferirla al

© Copyright IBM Corp. 1999,2003

dispositivo. Ejecutar la aplicación JSP en la estación de trabajo efectúa el proceso previo necesario de algunos de los archivos de aplicación.

Tareas afines:

- "Verificación del soporte de JSP en una estación de trabajo Windows" en la página 35
- "Configuración del desarrollo de JSP en un dispositivo Windows CE" en la página 36

Conceptos afines:

- "Visión general del soporte de JSP en DB2 Everyplace"
- Capítulo 16, "Las aplicaciones JSP de ejemplo", en la página 117

Consulta relacionada:

- "Subconjuntos JSP Versión 1.1 soportados" en la página 45
- "Códigos de personalización de IBM para acceder a bases de datos de aplicación de JSP" en la página 48

Visión general del soporte de JSP en DB2 Everyplace

El soporte de JSP en DB2 Everyplace consta de dos componentes:

- El mini servidor de Web HTTP
- El procesador JSP

El mini servidor de Web HTTP recibe peticiones de un navegador Web y devuelve respuestas al navegador Web (utilizando HTTP 1.1 como protocolo para las peticiones y respuestas). El procesador JSP analiza un archivo JSP, genera el código fuente Java correspondiente y compila el código fuente. El código fuente Java puede incluir JavaBeans que generen contenido dinámico cuando se solicite la página de JSP. Cuando se solicita una página de JSP, el mini servidor de Web HTTP ejecuta el código Java correspondiente y envía la salida al navegador Web como respuesta a la petición.

Cuando entre un URL como por ejemplo `http://localhost/request.jsp` en un navegador Web (donde **request.jsp** es la página de JSP que está solicitando), el navegador Web enviará la petición al mini servidor de Web HTTP. Se transmite la petición al procesador JSP si se cumple una de las condiciones siguientes:

- No existe ningún archivo .class correspondiente para la página de JSP correspondiente (por ejemplo, si la página de JSP es una página recién creada).
- Existe un archivo .class correspondiente para la página de JSP correspondiente, pero la indicación de la hora del archivo JSP es más reciente que la del archivo .class (por ejemplo, si se ha modificado la página de JSP).

  Nota: Para **request.jsp**, el archivo .class correspondiente es **_request_jsp_.class**.

Si se transmite la petición al procesador JSP y la sintaxis del archivo JSP es válida, el mini servidor de Web HTTP envía la salida al navegador Web, indicando que la página de JSP es válida. Pulse el enlace **request.jsp** en la salida para ver la página JSP.

Si se transmite la petición al procesador JSP y la sintaxis del archivo JSP no es válida, el mini servidor de Web HTTP envía información de diagnóstico al navegador Web.

Si no es necesario transmitir la petición al procesador JSP, el mini servidor de Web HTTP ejecuta el archivo .class correspondiente para la página de JSP y envía la salida al navegador Web.

El desarrollo de aplicaciones JSP se debe realizar en una estación de trabajo Windows. Es importante comprobar las páginas de JSP durante todo el desarrollo. El procesador JSP captará los posibles errores de sintaxis de una página de JSP. Una vez arreglados los errores, compruebe de nuevo la página de JSP pulsando el botón Renovar del navegador Web. Es posible que necesite suprimir los archivos de la antememoria o de la carpeta de Archivos temporales de Internet del navegador Web antes de que se reflejen los cambios en la página JSP. Después de completar la aplicación, podrá transferir la aplicación a un dispositivo y ejecutarlo en el dispositivo.

Nota: El procesador JSP se ejecuta en la estación de trabajo y no es necesario en el dispositivo.

Tareas afines:

- "Transferencia de una aplicación JSP a un dispositivo Windows CE" en la página 40
- "Verificación del soporte de JSP en una estación de trabajo Windows"
- "Configuración del desarrollo de JSP en un dispositivo Windows CE" en la página 36

Conceptos afines:

- "Desarrollo de aplicaciones JSP de DB2 Everyplace" en la página 33
- Capítulo 16, "Las aplicaciones JSP de ejemplo", en la página 117

Configuración del desarrollo de JSP

Este apartado proporciona información sobre el modo de configurar el desarrollo de aplicaciones de DB2 Everyplace utilizando Java Server Pages. Los temas que se tratan son:

- "Verificación del soporte de JSP en una estación de trabajo Windows"
- "Configuración del desarrollo de JSP en un dispositivo Windows CE" en la página 36
  - "Instalación del entorno de ejecución de una JVM J9 en un dispositivo Windows CE" en la página 37
  - "Instalación y verificación del soporte de JSP en un dispositivo Windows CE" en la página 38
- "Instalación y verificación del soporte de JSP en un dispositivo Symbian OS Versión 6" en la página 39

Verificación del soporte de JSP en una estación de trabajo Windows

Requisitos previos:

Verifique si en la estación de trabajo Windows está instalado el software siguiente:

- DB2 Everyplace Software Development Kit
- Java 2 Standard Development Kit, Standard Edition
  - Versión 1.1.8 si el destino es un dispositivo Symbian OS Versión 6

  - Versión 1.2.2 o posterior para otros destinos
- Navegador Web
  - Internet Explorer Versión 5.50 o posterior *o*
  - Netscape Navigator Versión 6.2.1 o anterior

Procedimiento:

Para verificar si la estación de trabajo Windows está configurada para utilizar el soporte de JSP de DB2 Everyplace:

- Ejecute la aplicación JSP de ejemplo Visiting Nurse:
  1. Inicie el mini servidor de Web HTTP:
     a. Abra una ventana de MS-DOS.
     b. Cambie al directorio `<DB2Everyplace>\SDK\JSP\Win32` utilizando el mandato `cd`.
     c. Escriba `runJspServer`.
  2. Abra un navegador Web en el siguiente URL:
     http://localhost/VisitingNurse/schedule.jsp

     Alternativamente, puede entrar en el URL http://localhost/ y navegar a la página VisitingNurse/schedule.jsp.

Si ha configurado satisfactoriamente la estación de trabajo para que utilice el soporte de JSP en DB2 Everyplace, en el navegador Web aparecerá una tabla de Planificación de Visiting Nurse.

Tareas afines:

- "Ejecución de una aplicación JSP en una estación de trabajo Windows" en la página 43

Consulta relacionada:

- "Subconjuntos JSP Versión 1.1 soportados" en la página 45
- "Códigos de personalización de IBM para acceder a bases de datos de aplicación de JSP" en la página 48

Configuración del desarrollo de JSP en un dispositivo Windows CE

Requisitos previos:

Verifique si en la estación de trabajo está instalado el software siguiente:

- DB2 Everyplace Software Development Kit

Verifique que el directorio `\Windows` del dispositivo contiene los archivos siguientes:

```
<DB2Everyplace>\Clients\WinCE\database\ver\proc\CryptoPlugin.dll
<DB2Everyplace>\Clients\WinCE\database\ver\proc\DB2e.dll
<DB2Everyplace>\Clients\WinCE\database\ver\proc\DB2eJDBC.dll
<DB2Everyplace>\Clients\WinCE\database\jdbc\db2ejdbc.jar
```

donde *ver* es el número de versión del sistema operativo Windows CE en el dispositivo y *proc* es el tipo de procesador.

Verifique si en el dispositivo está instalado el software siguiente:

- Navegador Web

Procedimiento:

1. Instale el entorno de ejecución J9 en el dispositivo.
2. Instale y verifique el soporte de JSP en DB2 Everyplace en el dispositivo.

Tareas afines:

- "Ejecución de una aplicación JSP en un dispositivo Windows CE" en la página 43

Consulta relacionada:

- "Subconjuntos JSP Versión 1.1 soportados" en la página 45
- "Códigos de personalización de IBM para acceder a bases de datos de aplicación de JSP" en la página 48

Instalación del entorno de ejecución de una JVM J9 en un dispositivo Windows CE

Actualmente, sólo los dispositivos StrongARM están soportados en Pocket PC. Si el dispositivo tiene un tipo de procesador diferente, puede probar otro JVM que de soporte a JNI (por ejemplo, Sun PersonalJava, Insignia Jeode, NSIcom CrEme). Si utiliza una JVM distinta de la J9 JVM, es necesario que modifique en consecuencia el `\SDK\JSP\WinCE\MiniHttpServer.lnk`.

Esta instalación de J9 le permite ejecutar las aplicaciones JSP de ejemplo. Es posible que tenga que instalar archivos J9 adicionales para sus propias aplicaciones.

Esta tarea forma parte de la tarea más importante de Configuración del desarrollo de JSP en un dispositivo Windows CE. Después de completar estos pasos, vuelva al apartado "Configuración del desarrollo de JSP en un dispositivo Windows CE" en la página 36.

Procedimiento:

Para instalar el entorno de ejecución J9 JVM:

1. Instale WebSphere Studio Device Developer v5.5 (WSDD) en la estación de trabajo. La versión de evaluación de WSDD se puede bajar desde la dirección http://www.ibm.com/software/pervasive/products/wsdd/. En los pasos que hay a continuación, `<WSDD>` hace referencia al directorio en el que instaló WSDD.
2. En WSDD, utilice el Gestor de actualización para instalar WCE Tooling for WSDD.
   a. Pulse **Ayuda —> Actualizaciones de software —> Gestor de actualización** para abrir la Perspectiva de Instalación/Actualización.
   b. En la Vista de actualizaciones de características, amplíe los nodos siguientes: **Sitios a visitar -> Entorno personalizado de WebSphere -> Entorno personalizado de WebSphere**.
   c. Pulse WCE Tooling for WSDD 5.5.0.
   d. En la Vista previa, pulse el botón **Instalar** y siga los pasos para la instalación.
   e. Instale las características adicionales siguientes siguiendo pasos análogos:
      - WCE jclMax Class Library
      - WCE Database Enabler Library

- WCE Personal Configuration Class Library

3. Cree la estructura de directorios siguiente en el dispositivo:

```
\wsdd\bin
\wsdd\lib
\wsdd\lib\jclMax
```

4. Copie los archivos siguientes en \wsdd\bin:

```
<WSDD>\wsdd5.0\ive\runtimes\pocketpc\arm\ive\bin\j9.exe
<WSDD>\wsdd5.0\ive\runtimes\pocketpc\arm\ive\bin\j9dyn20.dll
<WSDD>\wsdd5.0\ive\runtimes\pocketpc\arm\ive\bin\j9int20.dll
<WSDD>\wsdd5.0\ive\runtimes\pocketpc\arm\ive\bin\j9max20.dll
<WSDD>\wsdd5.0\ive\runtimes\pocketpc\arm\ive\bin\j9prt20.dll
<WSDD>\wsdd5.0\ive\runtimes\pocketpc\arm\ive\bin\j9thr20.dll
<WSDD>\wsdd5.0\ive\runtimes\pocketpc\arm\ive\bin\j9vm20.dll
<WSDD>\wsdd5.0\ive\runtimes\pocketpc\arm\ive\bin\j9zlib20.dll
<WSDD>\wsdd5.0\ive\runtimes\pocketpc\arm\ive\bin\iverel20.dll
<WSDD>\wsdd5.0\ive\runtimes\pocketpc\arm\ive\bin\swt-win32-2104.dll
```

5. Copie el archivo siguiente en \wsdd\lib:

```
<WSDD>\wsdd5.0\ive\runtimes\common\ive\lib\charconv.zip
```

6. Copie los archivos siguientes en \wsdd\lib\jclMax:

```
<WSDD>\wsdd5.0\ive\runtimes\common\ive\lib\jclMax\classes.zip
<WSDD>\wsdd5.0\ive\runtimes\common\ive\lib\jclMax\database_enabler.jar
<WSDD>\wsdd5.0\ive\runtimes\common\ive\lib\jclMax\locale.zip
<WSDD>\wsdd5.0\ive\runtimes\pocketpc\common\ive\lib\jclMax\prsnlwin.jar
```

Vuelva a “Configuración del desarrollo de JSP en un dispositivo Windows CE” en la página 36.

Instalación y verificación del soporte de JSP en un dispositivo Windows CE

Esta tarea forma parte de la tarea más importante de Configuración del desarrollo de JSP en un dispositivo Windows CE. Después de completar estos pasos, vuelva al apartado “Configuración del desarrollo de JSP en un dispositivo Windows CE” en la página 36.

Procedimiento:

Para instalar y verificar el soporte de JSP:

1. Instale el siguiente archivo en el dispositivo:

```
<DB2 Everyplace>\SDK\JSP\WinCE\DB2eJSP.CAB
```

2. Verifique que el dispositivo está configurado para utilizar el soporte de JSP en DB2 Everyplace ejecutando la aplicación JSP de ejemplo Visiting Nurse:
   a. Opcional en función del JVM: Modifique MiniHttpServer.lnk en el dispositivo. Tendrá que modificar el atajo si está utilizando un JVM que no sea el J9 JVM. El número con el que comienza el atajo es el número de caracteres que viene después del carácter ’#’. El número máximo de caracteres que viene después del carácter ’#’ es 256.
   b. Inicie el mini servidor de Web HTTP:
      1) Abra el Explorador de archivos.
      2) Navegue al directorio root.
      3) Pulse el atajo de MiniHttpServer.
   c. Abra un navegador Web en el siguiente URL: http://localhost/VisitingNurse/schedule.jsp

      Alternativamente, puede entrar en el URL http://localhost/ y navegar a la página VisitingNurse/schedule.jsp.

Si ha configurado satisfactoriamente el dispositivo para que utilice el soporte de JSP en DB2 Everyplace, en el navegador Web aparecerá una tabla de Planificación de Visiting Nurse.

Vuelva a "Configuración del desarrollo de JSP en un dispositivo Windows CE" en la página 36.

Instalación y verificación del soporte de JSP en un dispositivo Symbian OS Versión 6

Requisitos previos:

Verifique si en la estación de trabajo está instalado el software siguiente:

- DB2 Everyplace Software Development Kit

Verifique si en el dispositivo está instalado el software siguiente:

- Navegador Web
- Entorno de ejecución de Java `Java.sis`)

  `Java.sis` puede bajarse de Internet. Dispositivos más antiguos pueden proporcionar `Java.sis` en el CD-ROM del paquete de ventas para el dispositivo.

Verifique que el directorio `\system\libs` del dispositivo contiene los archivos siguientes:

- `<DB2Everyplace>\Clients\Symbian6\database\armi\CryptoPlugin.dll`
- `<DB2Everyplace>\Clients\Symbian6\database\armi\DB2e.dll`
- `<DB2Everyplace>\Clients\Symbian6\database\armi\db2ejdbc.dll`
- `<DB2Everyplace>\Clients\Symbian6\database\armi\ECSPKCS11.DLL`

Verifique que el directorio `\system\java\ext` del dispositivo contiene el archivo siguiente:

- `<DB2Everyplace>\Clients\Symbian6\database\armi\db2ejdbc.jar`

Procedimiento:

Para instalar y verificar el soporte de JSP en un dispositivo Symbian OS Versión 6:

1. Instale el siguiente archivo en el dispositivo:
   `<DB2Everyplace>\SDK\JSP\Symbian6\DB2eJSP.sis`.
2. Verifique que el dispositivo está configurado para utilizar el soporte de JSP en DB2 Everyplace ejecutando la aplicación JSP de ejemplo Visiting Nurse:
   a. Inicie el mini servidor de Web HTTP:
      1) Vaya a la pantalla Extras del dispositivo.
      2) Inicie la aplicación DB2eJSP.
   b. Abra un navegador Web en el siguiente URL:
      http://localhost/VisitingNurse/schedule.jsp

      Alternativamente, puede entrar en el URL http://localhost/ y navegar a la página VisitingNurse/schedule.jsp.

Si ha configurado satisfactoriamente el dispositivo para que utilice el soporte de JSP en DB2 Everyplace, en el navegador Web aparecerá una tabla de Planificación de Visiting Nurse.

Tareas afines:

- "Ejecución de una aplicación JSP en un dispositivo Symbian OS Versión 6" en la página 44

Consulta relacionada:

- "Subconjuntos JSP Versión 1.1 soportados" en la página 45
- "Códigos de personalización de IBM para acceder a bases de datos de aplicación de JSP" en la página 48

Transferencia de una aplicación JSP a un dispositivo Windows CE

Requisitos previos:

Antes de transferir la aplicación JSP a un dispositivo, complete las tareas siguientes:

- Instale el soporte de JSP en el dispositivo.
- Desarrolle y pruebe la aplicación en la estación de trabajo.

Procedimiento:

Para transferir una aplicación JSP en un dispositivo Windows CE:

1. Copie la base de datos que utiliza la aplicación en el dispositivo, si todavía no existe en el dispositivo. Asegúrese de poner la base de datos en el directorio en el que se espera que esté la aplicación. Es decir, el directorio que especifica el URL al que se conecta la aplicación.
2. Copie los archivos de la aplicación al directorio especificado para la propiedad JspPath de `MiniHttpConfig.properties` para el dispositivo. Si la aplicación está en un subdirectorio bajo JspPath en la estación de trabajo, cree la misma estructura de subdirectorio bajo JspPath en el dispositivo. Utilice las reglas siguientes al copiar los archivos de aplicación en el dispositivo:

Tabla 3.

Archivo de la estación de trabajo	Archivo que ha de copiarse en el dispositivo
`<page>.jsp`	`<page>_jsp_.class`
`<webapp>\web.xml`	`<webapp>\<webapp>_config_.class`

Notas:

- Los archivos .class a copiar en el dispositivo se generan por medio del procesador JSP al ejecutar la aplicación en la estación de trabajo. Se ubican en el mismo directorio que los archivos `.jsp` y `web.xml` correspondientes.
- Es posible que la aplicación no incluya un archivo `web.xml`. Las aplicaciones que genera WebSphere Studio Application Developer utilizan un archivo `web.xml`.
- La longitud máxima de un atajo es de 256 caracteres detrás del carácter '#' al principio del archivo. Para no sobrepasar la longitud máxima, copie los siguientes archivos de ejemplo de WebSphere Studio Application Developer en el directorio root en vez del directorio de la aplicación de ejemplo:
  - `dbbeans.jar`
  - cualquier archivo de clase ViewBean

Para ejecutar las aplicaciones que genera WebSphere Studio que utilizan estos archivos, la vía de acceso de clases de `MiniHttpServer.lnk` debe incluir `dbbeans.jar`, así como cualquier directorio que contenga archivos de la clase ViewBean.

- Los archivos de aplicación que no sean archivos `.jsp` o `web.xml` también deben copiarse en el dispositivo.

Por ejemplo, para copiar la aplicación JSP de ejemplo Visiting Nurse en el dispositivo:

a. Copie la base de datos de ejemplo Visiting Nurse en el dispositivo:
   1) Cree la estructura de directorios siguiente en el dispositivo: `\sample\data` . Este es el directorio que especifica el URL al que se conecta la aplicación.
   2) Copie el contenido del directorio `<DB2Everyplace>\SDK\JSP\sample\data` de la estación de trabajo de `\sample\data` en el dispositivo.

b. Copie los archivos de aplicación en el dispositivo:
   1) Cree la estructura de directorios siguiente en el dispositivo: `\sample\jsp` . Este es el directorio que se especifica para JspPath en el archivo `MiniHttpConfig.properties` por omisión.
   2) Cree el subdirectorio `\VisitingNurse` en `\sample\jsp`.
   3) Copie los archivos siguientes en `\sample\jsp\VisitingNurse` :

```
<DB2Everyplace>\SDK\JSP\sample\jsp\VisitingNurse\_schedule_jsp_.class
<DB2Everyplace>\SDK\JSP\sample\jsp\VisitingNurse\_contact_jsp_.class
<DB2Everyplace>\SDK\JSP\sample\jsp\VisitingNurse\_medrecord_jsp_.class
<DB2Everyplace>\SDK\JSP\sample\jsp\VisitingNurse\_person_jsp_.class
```

Tareas afines:

- “Configuración del desarrollo de JSP en un dispositivo Windows CE” en la página 36

Consulta relacionada:

- “Subconjuntos JSP Versión 1.1 soportados” en la página 45
- “Códigos de personalización de IBM para acceder a bases de datos de aplicación de JSP” en la página 48

Ejecución de una aplicación JSP

En este apartado se describe cómo ejecutar una aplicación JSP. Se tratan los temas siguientes:

- Opcional en función de la configuración de su estación de trabajo o aplicación: “Configuración del mini servidor de Web HTTP”
- “Ejecución de una aplicación JSP en una estación de trabajo Windows” en la página 43
- “Ejecución de una aplicación JSP en un dispositivo Windows CE” en la página 43
- “Ejecución de una aplicación JSP en un dispositivo Symbian OS Versión 6” en la página 44

Configuración del mini servidor de Web HTTP

Procedimiento:

El archivo siguiente es el archivo `MiniHttpConfig.properties` por omisión para estaciones de trabajo Windows. Puede utilizar este archivo por omisión o modificarlo de forma que se ajuste a los requisitos de la aplicación y del sistema. También puede modificar el archivo `MiniHttpConfig.properties` por omisión para los dispositivos.

```
# Propiedades de mini servidor de Web HTTP - Win32

# JspPath
# Especifica: vía de acceso que contiene páginas JSP (archivos .jsp y .class).
# Valor por omisión: JspPath=<directorio en el que se inició el mini servidor de Web HTTP>
#
# Nota: se usa \\ para indicar el separador de directorios
#
JspPath=sample\\jsp

# Puerto
#
# Especifica: puerto en el que está a la escucha el servidor
# Valor por omisión: Puerto=80
#
# Nota: Si no está utilizando el Puerto por omisión, inicie las peticiones de URL con:
#  http://localhost:Port/
# (en lugar de http://localhost/) donde Puerto es el Puerto especificado.
# Mime
#
# Especifica: Tipos de Mime
# Valor por omisión: Mime=text/html wml htm html,text/plain txt,image/gif gif,image/jpeg jpg
#
# Nota: Los tipos de Mime adicionales pueden añadirse utilizando el formato siguiente:
# Mime=mime_type_A ext1 ext2 ext3 ...,mime_type_B ext4 ext5 ...,...
#
Mime=application/octet-stream exe class,image/jpeg jpeg jpg

# LogFile
#
# Especifica: archivo de anotaciones cronológicas para el servidor
# Valores:
# "" - entradas de anotaciones cronológicas para la consola
# "no" - no se conservan entradas de anotaciones cronológicas
# "<nombre_archivo_anotaciones_cronológicas>" - las entradas de
# anotaciones cronológicas se graban en <nombre_archivo_anotaciones_cronológicas>
# Valor por omisión: LogFile=
#
LogFile=JspServer.log

# Índice
#
# Especifica: la página de índice (se visualiza si se solicita http://localhost)
# Valores:
# "" - se cargará el directorio JspPath
# "no" - no se cargará ninguna página
# se cargará "<archivo_índice>" - <archivo_índice>
# Valor por omisión: Índice=
```

Tareas afines:

- "Instalación y verificación del soporte de JSP en un dispositivo Symbian OS Versión 6" en la página 39

Consulta relacionada:

- "Subconjuntos JSP Versión 1.1 soportados" en la página 45
- "Códigos de personalización de IBM para acceder a bases de datos de aplicación de JSP" en la página 48
- "Soporte de JSP para los sistemas operativos" en la página 33

Ejecución de una aplicación JSP en una estación de trabajo Windows

Procedimiento:

Para ejecutar una aplicación JSP en una estación de trabajo Windows:

1. Si es necesario, configure el archivo `MiniHttpConfig.properties` en el directorio `<DB2Everyplace>\SDK\JSP\Win32`.
2. Inicie el mini servidor de Web HTTP:
   a. Abra una ventana de MS-DOS.
   b. Cambie al directorio `<DB2Everyplace>\SDK\JSP\Win32` utilizando el mandato 'cd'.
   c. Escriba `runJspServer`.
3. Abra un navegador Web y entre el URL para iniciar la aplicación JSP. Por ejemplo, si la página inicial de la aplicación es `start.jsp`, escriba `http://localhost/start.jsp`. Si ha configurado la propiedad de Puerto en el archivo `MiniHttpConfig.properties` , escriba en su lugar `http://localhost:Puerto/start.jsp` , siendo Puerto el número de puerto especificado.
4. Para detener el mini servidor de Web HTTP cuando acabe de ejecutar la aplicación JSP:
   a. Vaya a la ventana de MS-DOS en la que inició el servidor.
   b. Escriba Control+C y después escriba 's' para finalizar el trabajo por lotes.

Tareas afines:

- "Verificación del soporte de JSP en una estación de trabajo Windows" en la página 35

Consulta relacionada:

- "Subconjuntos JSP Versión 1.1 soportados" en la página 45
- "Códigos de personalización de IBM para acceder a bases de datos de aplicación de JSP" en la página 48

Ejecución de una aplicación JSP en un dispositivo Windows CE

Procedimiento:

Para ejecutar una aplicación JSP en un dispositivo Windows CE:

1. Transfiera la aplicación al dispositivo.
2. Si es necesario, modifique `MiniHttpServer.lnk` en el dispositivo. Tendrá que modificar el atajo si está utilizando un JVM que no sea el J9 JVM. El número con el que comienza el atajo es el número de caracteres que viene después del carácter '#'.
3. Si es necesario, configure el archivo `MiniHttpConfig.properties` para el dispositivo de Windows CE.
4. Inicie el mini servidor de Web HTTP:
   a. Abra el Explorador de archivos.
   b. Navegue al directorio root.
   c. Pulse el atajo de MiniHttpServer.

5. Abra un navegador Web y entre el URL para iniciar la aplicación JSP. Por ejemplo, si la página inicial de la aplicación es `start.jsp`, escriba `http://localhost/start.jsp`. Si ha configurado la propiedad de Puerto en el archivo `MiniHttpConfig.properties` , escriba en su lugar `http://localhost:Puerto/start.jsp` , siendo Puerto el número de puerto especificado.
6. Para detener el mini servidor de Web HTTP cuando acabe de ejecutar la aplicación JSP:
   a. Vaya a la ventana de consola J9.
   b. Pulse **Archivo —> Cerrar**.

Tareas afines:

- "Configuración del desarrollo de JSP en un dispositivo Windows CE" en la página 36

Consulta relacionada:

- "Subconjuntos JSP Versión 1.1 soportados" en la página 45
- "Códigos de personalización de IBM para acceder a bases de datos de aplicación de JSP" en la página 48

Ejecución de una aplicación JSP en un dispositivo Symbian OS Versión 6

Procedimiento:

1. Transfiera la aplicación al dispositivo.
2. Si es necesario, configure el archivo `MiniHttpConfig.properties` para el dispositivo Symbian.
3. Inicie el mini servidor de Web HTTP:
   a. Vaya a la pantalla Extras del dispositivo.
   b. Inicie la aplicación DB2eJSP.
4. Abra un navegador Web y entre el URL para iniciar la aplicación JSP. Por ejemplo, si la página inicial de la aplicación es `start.jsp`, escriba http://localhost/start.jsp. Si ha configurado la propiedad de Puerto en el archivo `MiniHttpConfig.properties`, escriba http://localhost:Port/start.jsp en su lugar, siendo Puerto el número de puerto especificado.
5. Para detener el mini servidor de Web HTTP cuando acabe de ejecutar la aplicación JSP, efectúe una restauración de software.

Tareas afines:

- "Instalación y verificación del soporte de JSP en un dispositivo Symbian OS Versión 6" en la página 39

Consulta relacionada:

- "Subconjuntos JSP Versión 1.1 soportados" en la página 45
- "Códigos de personalización de IBM para acceder a bases de datos de aplicación de JSP" en la página 48
- "Soporte de JSP para los sistemas operativos" en la página 33

Subconjuntos JSP Versión 1.1 soportados

En este apartado se listan y describen las instrucciones, los objetos implícitos, los elementos de escritura y las acciones estándar que se incluyen en el soporte de JSP de DB2 Everyplace.

Instrucciones:

Instrucción de página:

Descripción

La instrucción de página define atributos dependientes de la página.

Sintaxis

```
<%@ page
page_directive_attr_list %>
page_directive_attr_list ::=
{language="scriptingLanguage"}
{extends="className" }
{import="importList" }
{contentType="ctinfo" }
```

Atributos

Los cuatro atributos válidos para esta instrucción son:

- **language** - Tiene que ser 'java' si se especifica.
- **extends** - Nombre de clase de idioma de programación Java calificada al completo que da nombre a la superclase de la clase en la que se transforma esta página JSP.
- **import** - La lista de importación por omisión es `com.ibm.db2e.jsp.server.*, java.io.*, java.sql.* y java.util.*.`
- **contentType** - Puede ser cualquier valor (tal como text/html, text/xml, application/x-octet).

Ejemplo

```
<%@ page contentType="text/html" %>
```

Instrucción de inclusión:

Descripción

Utilice la instrucción de inclusión para incluir datos en una página de JSP. El archivo incluido puede tener elementos que también se procesarán.

Sintaxis

```
<%@ include file="relativeURLspec" %>
```

Atributos

El atributo para esta instrucción es **file**, que tiene que ser una vía de acceso relativa a una página. La vía de acceso no puede empezar por "/" y se interpretará como relativa a la página de JSP actual.

Ejemplo

```
<%@ include file="copyright.html" %>
```

Objetos implícitos:

Cuando se crean páginas de JSP, se tiene acceso a determinados objetos implícitos. Estos objetos se pueden utilizar dentro de scriptlets y expresiones, sin tenerlos que declarar antes. Cada objeto implícito tiene definida una clase en un paquete `com.ibm.db2e.jsp.server` de la tecnología esencial de Java, que se muestra en la Tabla 4 en la página 46.

Tabla 4. Objetos implícitos

Variable implícita	Tipo	Representación	Resumen del método
request	com.ibm.db2e.jsp.server.MiniHttpRequest	Petición para la página de JSP.	`java.lang.String` **`getParameter`**`(java.lang.String name)` `java.lang.String` **`getQueryString`**`()`
response	com.ibm.db2e.jsp.server. MiniHttpResponse	Respuesta a la petición	`java.lang.String` **`encodeURL`** `(java.lang.String url)` `void` **`setDateHeader`**`(java.lang.String name, long date)` `void` **`setHeader`**`(java.lang.String name, java.lang.String value)`
in	java.io.BufferedReader	Este objeto no está disponible actualmente.	
out	java.io.PrintStream	Objeto que graba en el navegador Web.	
config	com.ibm.db2e.jsp.server.DB2eJspConfig	El DB2eJspConfig para esta página JSP.	`java.lang.String` **`getInitParameter`**`(java.lang.String name)`
exception	java.lang.Throwable	Excepción emitida durante la ejecución de la página de JSP.	

Nota: Algunos de los tipos de objeto implícito anteriores difieren de los de JSP 1.1 debido a la implantación del soporte de JSP en DB2 Everyplace.

Elementos de escritura:

Declaraciones:

Descripción

Utilice declaraciones para declarar variables de Java y métodos utilizados en una página de JSP. Las declaraciones son variables miembro (campos y métodos) de la clase Java para la página de JSP.

Sintaxis

```
<%!declaration(s) %>
```

Ejemplo

```
<%!
String name = "Joe Smith";

public String getName() {
return name;
}
%>
```

Acciones estándar:

<jsp:useBean>:

Descripción

Una acción jsp:useBean asocia una instancia de un objeto de idioma de programación Java definido en el ámbito de ″página″ disponible con un determinado ID por medio de una variable de creación de scripts recién declarada del mismo ID.

Sintaxis

```
<jsp:useBean
id="name" scope="page|request|session|application"
typeSpec/>
typeSpec ::=
class="className"
```

Atributos

Los tres atributos válidos para este código son:

- **id** - Especifica el identificador de este bean. No reutilice este nombre en la página de JSP. Este atributo es obligatorio
- **scope** - Este atributo se ignora en el caso de que se especifique. Se utiliza la ″página″ de rango por omisión.
- **class** - Especifica la clase que representa este bean. Este atributo es obligatorio.

Ejemplo

```
<jsp:useBean id=″masterViewDBBean″
class=″Query1HTMLResultsMasterViewBean″ />
```

<jsp:setProperty>:

Descripción

La acción jsp:setProperty establece el valor de propiedades en un Bean.

Sintaxis

```
<jsp:setProperty name="beanName" prop_expr />
prop_expr ::=
property="propertyName" value="propertyValue"
propertyValue ::= string
The value propertyValue can also be a request-time
attribute value.
propertyValue ::=
expr_scriptlet
```

Atributos

Los tres atributos válidos para este código son:

- **name** - El nombre de una instancia de Bean definida por medio de un elemento <jsp:useBean> antes de que aparezca esta acción. La instancia de Bean debe contener la propiedad que se desea establecer. Este atributo es obligatorio.
- **property** - El nombre de la propiedad de Bean cuyo valor desee establecer. Este atributo es obligatorio.
- **value** - El valor que ha de asignarse a la propiedad concreta. Este atributo puede aceptar una expresión de atributo de tiempo de petición como valor. Este atributo es obligatorio.

Ejemplo

```
<jsp:setProperty name="masterViewDBBean"
property="username"
value='<%=config.getInitParameter("username")%>' />
```

Scriptlets:

Descripción

Utilice scriptlets para contener fragmentos válidos de código Java. Estos fragmentos de código se colocan en el código fuente para la página JSP y están relacionados con otros elementos de la página de JSP.

Sintaxis

```
<% scriptlet %>
```

Ejemplo

```
<%
try {
String name = Query1DBBean.getString(1);
out.println("Nombre = " + name);
}
catch (SQLException e) {

}
%>
```

Expresiones:

Descripción

Las expresiones son representaciones de serie de tipos de datos. Puede utilizar expresiones en consultas y en comentarios de HTML. La aplicación evalúa las expresiones durante la ejecución y las convierte en series.

Sintaxis

```
<%= expression %>
```

Ejemplo

```
<%= new java.util.Date() %>
```

Nota: Los nombres de variable que empiezan por *__db2ejsp__* son palabras clave y se utilizan internamente. No utilice estas variables en la página de JSP.

Conceptos afines:

- "Visión general del soporte de JSP en DB2 Everyplace" en la página 34
- "Desarrollo de aplicaciones JSP de DB2 Everyplace" en la página 33
- Capítulo 16, "Las aplicaciones JSP de ejemplo", en la página 117

Tareas afines:

- "Verificación del soporte de JSP en una estación de trabajo Windows" en la página 35
- "Configuración del desarrollo de JSP en un dispositivo Windows CE" en la página 36

Consulta relacionada:

- "Códigos de personalización de IBM para acceder a bases de datos de aplicación de JSP"

Códigos de personalización de IBM para acceder a bases de datos de aplicación de JSP

Los códigos siguientes son códigos de personalización de IBM que puede utilizar en la aplicación JSP para acceder a una base de datos DB2 Everyplace.

<tsx:dbconnect>:

Descripción

Este código establece una conexión con una base de datos DB2 Everyplace especificada utilizando el controlador JDBC de DB2 Everyplace.

Sintaxis

```
<tsx:dbconnect
id="connection_id"
driver="com.ibm.db2e.jdbc.DB2eDriver"
url="jdbc:db2e:database"
userid="db_user"
passwd="user_password">
</tsx:dbconnect>
```

Atributos

Los cinco atributos válidos para este código son:

- **id** - Especifica el identificador de esta conexión. No reutilice este nombre en la página de JSP. Este atributo es obligatorio.
- **driver** - Especifica el controlador JDBC de DB2 Everyplace. Este atributo es obligatorio.

- **url** - Especifica la base de datos DB2 Everyplace. El término *database* hace referencia a la vía de acceso de la base de datos de DB2 Everyplace. Este atributo es obligatorio.
- **userid** - Especifica un ID de usuario válido para la base de datos a la que ha de accederse. Este atributo es opcional.
- **passwd** - Especifica la contraseña de usuario para el atributo de idusuario. Este atributo es obligatorio si se especifica el idusuario.

Ejemplo

```
<tsx:dbconnect
id="conn"
driver="com.ibm.db2e.jdbc.DB2eDriver"
url="jdbc:db2e:sample/data/" >
</tsx:dbconnect>
```

<tsx:dbquery>:

Descripción

Este código somete una consulta a la base de datos utilizando la conexión especificada mediante el código <tsx:dbconnect> y produce un objeto java.sql.ResultSet en el que el cursor apunta a la primera fila del conjunto de resultados. Puede hacer referencia a este conjunto de resultados utilizando el identificador de esta consulta y la interfaz JDBC de DB2 Everyplace para java.sql.ResultSet.

Sintaxis

```
<tsx:dbquery
id="query_id" connection="connection_id" limit="value">
select_SQL_statement
</tsx:dbquery>
```

Atributos

Los atributos para este código son:

- **id** - Especifica el identificador de esta consulta. No reutilice este identificador de consulta en la página de JSP. Este atributo es obligatorio.
- **connection** - Especifica el identificador de un código <tsx:dbconnect> en este archivo JSP. Este atributo es obligatorio.
- **limit** - Especifica el número máximo de filas que la consulta puede devolver. Este atributo es opcional.

Parámetro

El parámetro válido para este código es:

- **sentencia_SQL_seleccionada** - Especifica la sentencia de SQL que se desea someter a la base de datos. Esta sentencia de consulta de SQL puede contener datos dinámicos.

Ejemplo

```
<tsx:dbquery id="Query1DBBean" connection="conn">
select <%= request.getParameter("column") %> from vnperson
</tsx:dbquery>
```

<tsx:dbmodify>:

Descripción

Este código somete un mandato para modificar datos de la base de datos utilizando la conexión especificada mediante el código <tsx:dbconnect>. No existe resultado para este código.

Sintaxis

```
<tsx:dbmodify
connection="connection_id">
modify_command
</tsx:dbmodify>
```

Atributo

El atributo para este código es:

- **connection** - Especifica el identificador de un código <tsx:dbconnect> en este archivo JSP. Este atributo es obligatorio.

Parámetro

El parámetro válido para este código es:

- **mandato_modificación** - Especifica el mandato de SQL que se desea someter a la base de datos para modificar datos. Este mandato de modificación puede contener datos dinámicos.

Ejemplo

```
<tsx:dbmodify connection="conn">
update vnperson set Name = '<%=Name%>' where ID = '<%=id%>'
</tsx:dbmodify>
```

<tsx:repeat>:

Descripción

Utilice este código para hacer un serpentear por cada una de la filas del resultado de la consulta. Los atributos start y stop controlan el proceso de serpenteo. Si no se especifican los atributos start y stop, el serpenteo termina cuando el cursor del conjunto de resultados, al que se hace referencia mediante el código <tsx:getProperty>, llega al final del conjunto de resultados. Este código se puede anidar.

Sintaxis

```
<tsx:repeat index="name" start="starting_index" stop="ending_index">
repeat_block
</tsx:repeat>
```

Atributos

Los atributos para este código son:

- **index** - Especifica el identificador para el índice de este código. Este atributo es opcional.
- **start** - Especifica el número de filas que hay que saltarse antes de procesar el bloque de repetición. El valor por omisión es 0. Este atributo es opcional.
- **stop** - Especifica el valor del índice final de este bloque de repetición. El valor por omisión es 2,147,483,647. Este atributo es opcional.

Parámetro

El parámetro válido para este código es:

- **bloque_repetición** - Especifica el bloque de codificación HTML que contiene la sintaxis del código <tsx:getProperty> y los códigos de HTML utilizados para dar formato al contenido. Si se coloca un código <tsx:getProperty> en el bloque de repetición, el avanza a la siguiente fila cada vez que se procesa el bloque de repetición.

Ejemplo

```
<TABLE border="1">
<TR>
<TH>Name</TH>
</TR>

<tsx:repeat>
<TR>
```

```
<TD>
<tsx:getProperty name="Query1DBBean" property="Name" />
</TD>
</TR>
</tsx:repeat>
</TABLE>
```

<tsx:getProperty>:

Descripción

Este código recupera el valor de un bean ResultSet que se visualizará en una página de JSP (es decir, en la página de resultado de HTML). Si se coloca este código dentro de un código de bloque <tsx:repeat>, el cursor del bean ResultSet avanza a la siguiente fila cada vez que se procesa el bloque de repetición.

Sintaxis

```
<tsx:getProperty name="bean_name" property="property_name" />
```

Atributos

Los atributos de este código son:

- **name** - Especifica el nombre del bean ResultSet que se ha declarado previamente mediante un código <tsx:dbquery> en este archivo JSP.
- **property** - Especifica la columna del bean ResultSet a la que se debe acceder.

Ejemplo

```
<tsx:getProperty name="Query1DBBean" property="FIRSTNAME" />
```

Conceptos afines:

- "Visión general del soporte de JSP en DB2 Everyplace" en la página 34
- "Desarrollo de aplicaciones JSP de DB2 Everyplace" en la página 33
- Capítulo 16, "Las aplicaciones JSP de ejemplo", en la página 117

Tareas afines:

- "Verificación del soporte de JSP en una estación de trabajo Windows" en la página 35
- "Configuración del desarrollo de JSP en un dispositivo Windows CE" en la página 36

Consulta relacionada:

- "Subconjuntos JSP Versión 1.1 soportados" en la página 45

Resolución de problemas de las aplicaciones de JSP

Resolución de problemas:

Salida de consola del mini servidor de Web HTTP:

1. Realacionado con WebSphere Studio Application Developer
   - java.lang.NoClassDefFoundError: javax/sql/DataSource

     WSAD v4.0.3+ contiene el arreglo para este error. Puede utilizar el `dbbeans.jar` in `<DB2Everyplace>\SDK\JSP\sample\jsp\VNSchedule_wsad40, which is the jar from WSAD v4.0.3.`
   - "No se puede determinar el valor de búsqueda para el nombre de tipo de columna <nombre>. Se supone que searchable = true."

     Este mensaje lo genera `dbbeans.jar` y no puede ignorarse.

Navegador Web:

- ″La página que está buscando no puede encontrarse.″

  Si Pocket Internet Explorer visualiza este mensaje cuando se trata de conectar al URL ″http://localhost/″, su dispositivo Windows CE es probablemente más antiguo que Pocket PC v3.0.11171 y ha de obtener un arreglo para el navegador Web. (En Internet, busque el modo de conectarse a un sistema principal local por medio de Pocket Internet Explorer.) Asegúrese de que el mini servidor de web HTTP se está ejecutando.

Desarrollo de la aplicación:

- Los cambios en la página JSP no se reflejan en el navegador Web

  Si se produce esto, es posible que tenga que suprimir los archivos de la antememoria o carpeta de Archivos temporales de Internet del navegador Web.

Conceptos afines:

- “Visión general del soporte de JSP en DB2 Everyplace” en la página 34
- “Desarrollo de aplicaciones JSP de DB2 Everyplace” en la página 33
- Capítulo 16, “Las aplicaciones JSP de ejemplo”, en la página 117

Tareas afines:

- “Verificación del soporte de JSP en una estación de trabajo Windows” en la página 35
- “Configuración del desarrollo de JSP en un dispositivo Windows CE” en la página 36

Consulta relacionada:

- “Códigos de personalización de IBM para acceder a bases de datos de aplicación de JSP” en la página 48

Capítulo 7. Desarrollo de aplicaciones .NET

Este capítulo proporciona información sobre el desarrollo de aplicaciones DB2 Everyplace que utilizan .NET. Los temas que se tratan son:

- "Soporte de sincronización"
- "Soporte para crear aplicaciones de .NET" en la página 56

Soporte de sincronización

Esta sección proporciona información sobre el soporte de sincronización de DB2 Everyplace .NET. Los temas que se tratan son:

- "Ubicaciones de archivo de API de ISync.Net"
- "Utilización de la API de ISync.NET" en la página 54
- "Utilización de ISyncComponent" en la página 55
- "Aplicación de ejemplo simple que utiliza la API de ISync.NET" en la página 56

Ubicaciones de archivo de API de ISync.Net

El DB2 Everyplace Sync Client proporciona dos API que permiten a los desarrolladores crear aplicaciones gestionadas para el DB2 Everyplace Sync Server. Estas API incluyen:

- ISyncComponent
- ISync.Net

ISyncComponent es más pequeño que ISync.Net, pero proporciona soporte de diseño visual para los desarrolladores que deseen utilizar esta función.

Para obtener más información sobre los proveedores de .Net para el motor de base de datos, consulte "Visión general del soporte de .NET para crear aplicaciones en la base de datos cliente" en la página 57.

Tabla 5. Espacios de nombre y ubicación de proveedor gestionados por ISync.NET

Proveedores disponibles	Espacios de nombres	Plataformas soportadas	Ubicación, `\DB2Everyplace\Clients`
No Unicódigo para .NET Framework	IBM.Data.Sync IBM.Data.Sync.DB2e	Win 32	`Win32\Sync\NMP\IBM.Data.Sync.DB2e.dll`
Unicódigo para .NET Framework	IBM.Data.Sync IBM.Data.Sync.DB2e	Unicódigo Win32	`Win32\Sync\NMP\unicode\IBM.Data.Sync.DB2e.dll`
.NET Compact Framework	IBM.Data.Sync IBM.Data.Sync.DB2e.CF	WinCE	`WinCE\Sync\NMP\IBM.Data.Sync.DB2e.CF.dll`

Aplicación ISync.NET de ejemplo

Hay dos aplicaciones de ejemplo ubicadas en \DB2Everyplace\Clients\clientapisample\NMP*idioma*:

- GoISync
- sample1

donde *idioma* es C# (cs) o Visual Basic (vb)

Especificación de API de ISync.NET

© Copyright IBM Corp. 1999,2003

Puede encontrar la especificación de API para ISync.NET en \DB2Everyplace\Clients\clientapisample\NMP\ISync.NET.chm

Conceptos afines:

- "Aplicación de ejemplo simple que utiliza la API de ISync.NET" en la página 56
- "Visión general del soporte de .NET para crear aplicaciones en la base de datos cliente" en la página 57
- "Código de aplicación del proveedor de datos de .NET de DB2 Everyplace para WinCE y Win32" en la página 61

Tareas afines:

- "Utilización de la API de ISync.NET"
- "Utilización de ISyncComponent" en la página 55
- "Visión general del modo de desarrollar aplicaciones ADO.NET utilizando el DB2 Everyplace .NET Data Provider" en la página 57

Utilización de la API de ISync.NET

Puede encontrar la especificación de API para ISync.NET en \DB2Everyplace\Clients\clientapisample\NMP\ISync.NET.chm

Requisito previo:

Requisitos de software

- DB2 Everyplace Versión 8.1.4, beta 1
- Microsoft. NET Standard Framework 1.0 (incluido con Visual Studio 2002) — necesario para desarrollar aplicaciones en Win32
- Microsoft .NET Compact Framework (incluido con Visual Studio 2003) — necesario para desarrollar aplicaciones en WinCE

Aunque el proveedor de ISync.NET es independiente de la plataforma y del idioma, seguirá dependiendo de las bibliotecas de Cliente de sincronización nativa subyacentes. Tanto las bibliotecas del Cliente de sincronización como el proveedor deben incluirse en la vía de acceso al usuario en tiempo de ejecución de la aplicación. Durante la instalación de DB2 Everyplace deben actualizarse las vías de acceso de usuario.

Para obtener más información sobre las bibliotecas de cliente, consulte el Capítulo 3 ″Instalación de DB2 Everyplace en dispositivo portátil″ de la publicación *DB2 Everyplace Guía del usuario y de instalación*.

Procedimiento:

Para utilizar Isync.NET en las aplicaciones, deben realizarse los pasos siguientes para todas las Visual Studio Frameworks y tipos de aplicación.

1. En Microsoft Visual Studio .NET, cree un proyecto nuevo en el idioma de su elección.
2. En su aplicación, importe los espacios de nombre de DB2 Everyplace. A continuación se proporciona un ejemplo para la Estructura estándar:

```
[Visual Basic]
     Imports IBM.Data.Sync
     Imports IBM.Data.Sync.DB2e
     [C#]
     using IBM.Data.Sync;
     using IBM.Data.Sync.DB2e;
```

Para obtener más información, puede ver la aplicación de sincronización de ejemplo ubicada en \DB2Everyplace\Clients\clientapisample\NMP

3. Añada una referencia:
   a. En Visual Studio, pulse el botón derecho del ratón en el nombre de proyecto y seleccione **Añadir referencia**.
   b. En la pestaña Proyectos, navegue a la ubicación de **IBM.Data.Sync.DB2e.dll**.
   c. En la línea de mandatos, escriba: **csc /t:exe /r:IBM.Data.Sync.DB2e.dll ISyncSample.cs**.

Conceptos afines:

- "Ubicaciones de archivo de API de ISync.Net" en la página 53
- "Aplicación de ejemplo simple que utiliza la API de ISync.NET" en la página 56
- "Visión general del soporte de .NET para crear aplicaciones en la base de datos cliente" en la página 57
- "Código de aplicación del proveedor de datos de .NET de DB2 Everyplace para WinCE y Win32" en la página 61

Tareas afines:

- "Utilización de ISyncComponent"
- "Visión general del modo de desarrollar aplicaciones ADO.NET utilizando el DB2 Everyplace .NET Data Provider" en la página 57

Utilización de ISyncComponent

Procedimiento:

ISyncComponent proporciona asimismo un soporte de diseño mínimo en la Estructura estándar. Este soporte básico le permite arrastrar y soltar en un formulario y modificar las propiedades ConnectionString (servidor, puerto y nombre de usuario) y TargetPath (directorio destino para los datos). Al desarrollar una aplicación Windows de Visual Studio, asegúrese de añadir el componente de DB2 Everyplace **IBM.Data.Sync.DB2e.dll** al **Toolbox**.

Nota: Las bibliotecas de Cliente de sincronización nativo deben estar previamente en la vía de acceso de usuario para que este proceso se complete de modo satisfactorio.

Para la Estructura estándar, hay una opción para utilizar una API más simple utilizando IBM.Data.Sync.DB2e.ISyncComponent.

```
ISyncComponent comp1 = new ISyncComponent();
comp1.ConnectionString = SERVER=localhost;PORT=80;UID=username;PWD=password;
comp1.TargetPath = data;
comp1.Sync();
comp1.Close();
```

Conceptos afines:

- "Ubicaciones de archivo de API de ISync.Net" en la página 53
- "Aplicación de ejemplo simple que utiliza la API de ISync.NET" en la página 56
- "Visión general del soporte de .NET para crear aplicaciones en la base de datos cliente" en la página 57
- "Código de aplicación del proveedor de datos de .NET de DB2 Everyplace para WinCE y Win32" en la página 61

Tareas afines:

- "Utilización de la API de ISync.NET" en la página 54
- "Visión general del modo de desarrollar aplicaciones ADO.NET utilizando el DB2 Everyplace .NET Data Provider" en la página 57

Aplicación de ejemplo simple que utiliza la API de ISync.NET

Este tema incluye un ejemplo que proporciona una referencia rápida del modo de utilizar la API de ISync.NET.

```
// Propiedades de sincronización
private Hashtable userProps = new Hashtable();

// Obtener una instancia de DB2eISyncProvider
ISyncProvider provider = DB2eISyncProvider.GetInstance();

// Configurar propiedades
userProps.Add("isync.user", username);
userProps.Add("isync.password", password);

// Obtener una instancia del servicio de sincronización del proveedor
ISyncService service = provider.CreateSyncService(http://localhost:80, userProps);

// Obtener una instancia del almacén de configuración
ISyncConfigStore config = service.GetConfigStore(data);

// Obtener una instancia del controlador de sincronización para realizar la sincronización
ISyncDriver syncer = config.GetSyncDriver();

// Realizar la sincronización
syncer.Sync();

// Cerrar objetos
syncer.Close();
config.Close();
service.Close();
```

Conceptos afines:

- "Ubicaciones de archivo de API de ISync.Net" en la página 53
- "Visión general del soporte de .NET para crear aplicaciones en la base de datos cliente" en la página 57
- "Código de aplicación del proveedor de datos de .NET de DB2 Everyplace para WinCE y Win32" en la página 61

Tareas afines:

- "Utilización de la API de ISync.NET" en la página 54
- "Utilización de ISyncComponent" en la página 55
- "Visión general del modo de desarrollar aplicaciones ADO.NET utilizando el DB2 Everyplace .NET Data Provider" en la página 57

Soporte para crear aplicaciones de .NET

Esta sección proporciona información sobre el soporte para crear aplicaciones de .NET. Los temas que se tratan son:

- "Visión general del soporte de .NET para crear aplicaciones en la base de datos cliente" en la página 57
- "Visión general del modo de desarrollar aplicaciones ADO.NET utilizando el DB2 Everyplace .NET Data Provider" en la página 57
- "Código de aplicación del proveedor de datos de .NET de DB2 Everyplace para WinCE y Win32" en la página 61

Visión general del soporte de .NET para crear aplicaciones en la base de datos cliente

DB2 Everyplace proporciona las herramientas para permitir que los desarrolladores creen aplicaciones que utilizan la API de ADO.NET para manipular los datos gestionados por medio de la base de datos de DB2 Everyplace. DB2 Everyplace contiene dos Proveedores de datos de .NET. Un proveedor se ejecuta en el .NET Framework 1.0 y el otro proveedor se ejecuta en .NET Compact Framework. Encontrará estos proveedores o API en:

- **Para Win32:**

  `\DB2Everyplace\Clients\Win32\database\nmp\IBM.Data.DB2.DB2e.dll`
- **Para WinCE:**

  `\DB2Everyplace\Clients\WinCE\database\nmp\IBM.Data.DB2.DB2e.CF.dll`

Las especificaciones de API están ubicadas en

`\DB2Everyplace\Clients\Win32\database\nmp\doc\readme.html`

Para obtener más información sobre los proveedores de .NET para el Cliente de sincronización, consulte la publicación *DB2 Everyplace Sync Server Administration Guide*, en el apartado denominado "Configuración de la base de datos fuente en el sistema fuente de Windows."

Para simplificar la transición para los programadores que han utilizado el Proveedor de datos de Microsoft ODBC .NET en el pasado, las nuevas interfaces de DB2 Everyplace .NET Data Provider son casi idénticas a las del Proveedor de datos de .NET de Microsoft ODBC. Por ejemplo, el DB2 Everyplace .NET Data Provider de Microsoft ODBC tiene la clase OdbcConnection, en tanto que el DB2 Everyplace .NET Data Provider tiene DB2eConnection como clase de función equivalente. De modo análogo, puede sustituir 'Odbc' por 'DB2e' en los demás nombres de clase para obtener las clases de DB2 Everyplace .NET Data Provider correspondientes.

Conceptos afines:

- "Ubicaciones de archivo de API de ISync.Net" en la página 53
- "Aplicación de ejemplo simple que utiliza la API de ISync.NET" en la página 56
- "Código de aplicación del proveedor de datos de .NET de DB2 Everyplace para WinCE y Win32" en la página 61

Tareas afines:

- "Utilización de ISyncComponent" en la página 55
- "Visión general del modo de desarrollar aplicaciones ADO.NET utilizando el DB2 Everyplace .NET Data Provider"

Visión general del modo de desarrollar aplicaciones ADO.NET utilizando el DB2 Everyplace .NET Data Provider

Los espacios de nombres para el DB2 Everyplace .NET Data Provider son los siguientes:

- **Ejecución en el .NET Compact Framework:** IBM.Data.DB2.DB2e.CF
- **Ejecución en el .NET Framework:** IBM.Data.DB2.DB2e

El DB2 Everyplace .NET Data Provider proporciona funciones para conectarse a una fuente de datos de DB2 Everyplace, la ejecución de mandatos y la recuperación de mandatos. Dichos resultados puede procesarse directamente o

ubicarse en un Archivo ADO.NET para proseguir el proceso mientras se está en estado de desconexión. Mientras están en el Archivo, los datos pueden revelarse al usuario, combinados con otros datos procedentes de varias fuentes o pasarse de modo remoto entre niveles. Los procesos que se realicen en los datos mientras estén en el Archivo podrán reconciliarse posteriormente con la fuente de datos.

El diseño de DB2 Everyplace .NET Data Provider es sencillo. Consta de una mínima capa entre DB2 Everyplace y el código que amplía las funciones sin sacrificar el rendimiento.

Las clases de DB2 Everyplace .NET Data Provider heredan o implantan miembros de otras interfaces o clases de .NET Framework. La documentación de este proveedor incluye un resumen de los miembros soportados en cada una de estas clases. Para obtener una información más detallada sobre un miembro heredado específico, consulte el tema apropiado de Microsoft® .NET Framework SDK.

Requisito previo:

Tabla 6. Requisitos previos para utilizar el DB2 Everyplace .NET Data Provider

Componente	Requisito mínimo
Microsoft.NET Framework	Microsoft.NET Framework 1.0 Debe instalarse antes que el DB2 Everyplace .NET Data Provider para el desarrollo de la aplicación
Microsoft Visual Studio.NET 2003	Microsoft Visual Studio.NET 2003 para desarrollo de aplicaciones portátiles
Microsoft.NET Compact Framework	Microsoft .NET Compact Framework 1.0 para desarrollo de portátiles Debe instalarse en el dispositivo antes de instalar el DB2 Everyplace .NET Data Provider para el desarrollo de aplicaciones portátiles.
Producto DB2 Everyplace	• `DB2e.dll` de la versión 8.1.4 o posterior • `AgentProxy.dll` de la versión 8.1.4 o posterior que requiere la llamada al procedimiento almacenado remoto. • `wbxmllib.dll` de la versión 8.1.4 o posterior que requiere la llamada al procedimiento almacenado remoto. • DB2 Everyplace Sync Server de la versión 8.1.4 o posterior que requiere la llamada al procedimiento almacenado remoto. `DB2e.dll`, `AgentProxy.dll` y `wbxmllib.dll` son bibliotecas nativas y por tanto dependen del procesador; por tanto, el sistema operativo ha de localizar estas bibliotecas nativas (estableciendo la variable de entorno PATH, por ejemplo) para que DB2 Everyplace .NET Data Provider funcione adecuadamente.

Restricciones:

Limitaciones del proveedor:

- En DB2 Everyplace no se permite actualmente la actualización de columnas clave primarias.

- La recuperación del conjunto de resultados utilizando un procedimiento almacenado remoto tiene una limitación sobre el tamaño del conjunto de resultados.
- No se da soporte a las llamadas de procedimiento almacenado locales.
- Para los métodos o propiedades que no estén soportadas, se emitirá una `System.NotSupportedException`.

Seguridad de hebra:

Los miembros públicos no de instancia de este proveedor resultan seguros para las operaciones de varias hebras. No se garantiza que los miembros de instancia tengan seguridad de hebra.

Procedimiento:

Hay cuatro objetos de núcleo que constituyen el DB2 Everyplace .NET Data Provider. La tabla siguiente describe estos objetos y su función.

Tabla 7. DB2 Everyplace .NET Data Provider, objetos de núcleo

Objeto	Descripción
DB2eConnection	Establece una conexión para una fuente de datos de DB2 Everyplace y puede comenzar por *Transacción*.
DB2eCommand	Ejecuta un mandato en un servidor de DB2 Everyplace y revela *Parámetros*.
DB2eDataAdapter	Puebla un *Archivo* y resuelve actualizaciones en la fuente de datos de DB2 Everyplace.
DB2eDataReader	Revela y lee una corriente de datos sólo de avance desde una fuente de datos de DB2 Everyplace.

Junto con las clases de núcleo listadas en la tabla precedente, el DB2 .NET Data Provider contiene asimismo las clases listadas en la tabla siguiente.

Tabla 8. DB2 Everyplace .NET Data Provider, clases adicionales

Objeto	Descripción
DB2eCommandBuilder	Objeto de ayuda que generará automáticamente propiedades de mandato del *DB2eDataAdapter* o derivará información de parámetro de un procedimiento almacenado y poblará la colección de *DB2eParameters* de un objeto de *DB2eCommand*. Nota: La utilización del *DB2eCommandBuilder* no se recomienda ya que puede generar sentencias SQL muy ineficaces y, en algunos casos, no válidas.
DB2eError	Revela la información de un aviso o error devuelto por una fuente de datos de DB2 Everyplace.
DB2eException	Se devuelve cuando se encuentra un error en la fuente de datos de DB2 Everyplace. Para los errores encontrados en el cliente, los proveedores de datos de .NET emiten una excepción de .NET Framework.
DB2eParameter	Define parámetros de valor de retorno, salida y entrada para mandatos y procedimientos almacenados.
DB2eTransaction	Le permite reclutar mandatos en las transacciones de la fuente de datos de DB2 Everyplace.

Para utilizar el DB2 Everyplace .NET Data Provider, deberá añadir una sentencia de importación o utilización para el IBM.Data.DB2.DB2e o espacio de nombre al .DLL de la aplicación, tal y como ilustra el código siguiente:

[Visual Basic]

```
Imports IBM.Data.DB2.DB2e
```

[C#]

```
using IBM.Data.DB2.DB2e;
```

También debe incluir una referencia al .DLL al compilar el código. Por ejemplo, si está compilando un programa Microsoft® Visual C#™, la línea de mandatos debería incluir:

```
csc /r:IBM.Data.DB2.DB2e.dll
```

Para el .NET Compact Framework, el espacio de nombres es IBM.Data.DB2.DB2e.CF y la aplicación ha de hacer referencia al conjunto IBM.Data.DB2.DB2e.CF.dll.

Para obtener información sobre el mejor modo de utilizar este espacio de nombre, consulte la documentación sobre las siguientes clases de DB2 Everyplace.NET Data Provider:

- DB2eDataAdapter
- DB2eCommand
- DB2eConnection
- DB2eDataReader

Para obtener más información sobre el modo en el que el DB2 Everyplace .NET Data Provider funciona en el .NET Framework, consulte IBM.Data.DB2.DB2e Hierarchy.

Tabla 9. Clases

Objeto	Descripción
DB2eCommand	Representa el procedimiento almacenado o la sentencia SQL que ha de ejecutarse frente a la fuente de datos. Esta clase no puede heredarse.
DB2eCommandBuilder	Genera automáticamente mandatos de una sola tabla utilizada para reconciliar los cambios efectuados en un *Archivo* con la fuente de datos asociada. Esta clase no puede heredarse.
DB2eConnection	Representa una conexión abierta con una fuente de datos.
DB2eDataAdapter	Representa un conjunto de mandatos de datos y una conexión con una fuente de datos que se utilizan para rellenar el *Archivo* y actualizar la fuente de datos. Esta clase no puede heredarse.
DB2eDataReader	Proporciona un modo de leer una corriente de filas de datos sólo de avance a partir de una fuente de datos. Esta clase no puede heredarse.
DB2eError	Colecciona información relevante para un aviso o error devueltos por la fuente de datos. Esta clase no puede heredarse.
DB2eException	La excepción que se genera cuando la fuente de datos de DB2 Everyplace devuelve un aviso o error. Esta clase no puede heredarse.

Tabla 9. Clases (continuación)

Objeto	Descripción
DB2eParameter	Representa un parámetro para *DB2eCommand* y, opcionalmente, su correlación con una *DataColumn*. Esta clase no puede heredarse.
DB2eTransaction	Representa una transacción de SQL que ha de efectuarse en una fuente de datos. Esta clase no puede heredarse.

Tabla 10. Delegados

Delegado	Descripción
DB2eInfoMessageEventHandler	Representa el método que manejará el suceso de InfoMessage de una DB2eConnection.
DB2eRowUpdatedEventHandler	Representa el método que manejará el suceso de RowUpdated de un DB2eDataAdapter.
DB2eRowUpdatingEventHandler	Representa el método que manejará el suceso de RowUpdating de un DB2eDataAdapter.

Tabla 11. Enumeraciones

Enumeración	Descripción
DB2eType	Especifica el tipo de datos de un campo, propiedad o DB2eParameter.

Tabla 12. Palabras clave de serie de conexión de DB2 Everyplace .NET Provider

Palabra clave	Descripción
Database	Ubicación de la base de datos. Por ejemplo: `C:\data1\`
UID	ID de usuario
PWD	Contraseña

C# Example:

```
string connString = @"Database=C:\data1\; UID=user; PWD=userpwd";
```

Conceptos afines:

- "Ubicaciones de archivo de API de ISync.Net" en la página 53
- "Aplicación de ejemplo simple que utiliza la API de ISync.NET" en la página 56
- "Visión general del soporte de .NET para crear aplicaciones en la base de datos cliente" en la página 57
- "Código de aplicación del proveedor de datos de .NET de DB2 Everyplace para WinCE y Win32"

Tareas afines:

- "Utilización de la API de ISync.NET" en la página 54
- "Utilización de ISyncComponent" en la página 55

Código de aplicación del proveedor de datos de .NET de DB2 Everyplace para WinCE y Win32

Hay dos aplicaciones de ejemplo que ilustran el modo de desarrollar aplicaciones para WinCE y Win32 utilizando el DB2 Everyplace .NET Data Provider:

- DB2eSample1.cs
- DB2eSample2.sc

Ambos archivos están ubicados en:

- **Para Win32 y WinCE**

  `\DB2Everyplace\Clients\Win32\database\nmp\samples`

A continuación se proporciona un ejemplo de una de las aplicaciones de ejemplo:

```
using System;
using System.Text;
using System.Data;

using IBM.Data.DB2.DB2e;

/*
 * Sample1
 *
 * The following example creates a table, insert some rows to it, fetches
 * all the rows from the table, and finally drops the table.
 *
 */
namespace IBM.Data.DB2.DB2e.Samples
{
        class DB2eSample1
        {
                /// <resumen>



      /// El principal punto de entrada para la aplicación.
     /// </resumen>
     [STAThread]
     static void Main(string[] args)
       {
         DB2eConnection conn   = null;
         DB2eCommand cmd       = null;
         DB2eDataReader reader = null;
          String connString     = @"database=.\; uid=user1; pwd=user1";
          int rowsAffected      = 0;

           try
                {
                 conn = new DB2eConnection(connString);
                 conn.Open();

                   Console.WriteLine("creating table t1...");
                   cmd = new DB2eCommand("create table t1 (c1 int primary key not null,
        c2 smallint, c3 char(10), c4 varchar(10), c5 decimal(8,2), c6 date,
        c7 time, c8 timestamp )", conn);
                      rowsAffected = cmd.ExecuteNonQuery();
                      Console.WriteLine("inserting a row into table t1...");
                      cmd.CommandText = "insert into t1 values (1, 10, 'John',
         'Yip', null, current date, current time, current timestamp)";
                  rowsAffected = cmd.ExecuteNonQuery();
                  Console.WriteLine("inserting a row into table t1...");
                  cmd.CommandText = "insert into t1 values (2, 20, 'Mary', 'Jann',
         2.2, current date, current time, current timestamp)";
                  rowsAffected = cmd.ExecuteNonQuery();
                  cmd.CommandText = "select * from t1";
```

```
Console.WriteLine("fetching resultset from table t1...");
reader = cmd.ExecuteReader();
    while (reader.Read())
      {
        if (!reader.IsDBNull(0))
           Console.Write(reader.GetInt32(0) + "\t");
        else
            Console.Write("NULL " + "\t");
        if (!reader.IsDBNull(1))
            Console.Write(reader.GetInt16(1) + "\t");
        else
            Console.Write("NULL " + "\t");
        if (!reader.IsDBNull(2))
            Console.Write(reader.GetString(2) + "\t");
        else
            Console.Write("NULL " + "\t");
        if (!reader.IsDBNull(3))
            Console.Write(reader.GetString(3) + "\t");
        else
            Console.Write("NULL " + "\t");
        if (!reader.IsDBNull(4))
            Console.Write(reader.GetDecimal(4) + "\t");
        else
            Console.Write("NULL " + "\t");
        if (!reader.IsDBNull(5))
            Console.Write(reader.GetDate(5) + "\t");
        else
            Console.Write("NULL " + "\t");
        if (!reader.IsDBNull(6))
            Console.Write(reader.GetTime(6) + "\t");
        else
            Console.Write("NULL " + "\t");
        if (!reader.IsDBNull(7))
            Console.Write(reader.GetDateTime(7) + "\t");
        else
            Console.Write("NULL " + "\t");
            Console.WriteLine();
        }
   reader.Close();
   reader = null;
   Console.WriteLine("dropping table t1...");
   cmd.CommandText = "drop table t1";
   cmd.ExecuteNonQuery();
 }
   catch (DB2eException e1)
  {
       int cnt = e1.Errors.Count;
       for (int i=0; i < cnt; i++)
         {
           Console.WriteLine("Error #" + i + "\n" +
              "Message: " + e1.Errors[i].Message + "\n" +
              "Native: " + e1.Errors[i].NativeError.ToString() + "\n" +
              "SQL: " + e1.Errors[i].SQLState + "\n");
         }
      }
   catch (Exception ex)
     {
           Console.WriteLine(ex.Message);
     }
   finally
```

```
                        {
                             if (reader != null)
                                {
                                   reader.Close();
                                   reader = null;
                                }
                             if (conn != null)
                                {
                                   conn.Close();
                                   conn = null;
                                }
                        }
                }  // final del Principal

        }  // final de clase

} // final del espacio de nombre
```

Conceptos afines:

- "Ubicaciones de archivo de API de ISync.Net" en la página 53
- "Aplicación de ejemplo simple que utiliza la API de ISync.NET" en la página 56
- "Visión general del soporte de .NET para crear aplicaciones en la base de datos cliente" en la página 57

Tareas afines:

- "Utilización de la API de ISync.NET" en la página 54
- "Utilización de ISyncComponent" en la página 55
- "Visión general del modo de desarrollar aplicaciones ADO.NET utilizando el DB2 Everyplace .NET Data Provider" en la página 57

Capítulo 8. Cómo conectarse a una base de datos de DB2 Everyplace

En este capítulo se explica cómo conectar con una base de datos DB2 Everyplace. Se tratan los temas siguientes:

- "Visión general de las tablas de base de datos de DB2 Everyplace"
- "Manejo de conflictos de denominación"
- "Cómo conectarse a la base de datos de DB2 Everyplace" en la página 66
- "Serialización de conexiones" en la página 67
- "Bases de datos DB2 Everyplace en soportes de almacenamiento de sólo lectura" en la página 68

Visión general de las tablas de base de datos de DB2 Everyplace

Una base de datos DB2 Everyplace se compone de varias tablas de catálogos del sistema y diversas tablas definidas por el usuario. Cada una de las tablas se almacena en dos archivos, uno para los datos propiamente dichos y el otro para los índices. Todos los índices se conservarán en el mismo archivo de índices. A diferencia de DB2 Universal Database, las bases de datos DB2 Everyplace no tienen nombre y no se pueden catalogar ni descatalogar. Es decir, que se ignora el nombre de la base de datos. En DB2 Everyplace, una base de datos no es más que un grupo de archivos que se pueden copiar o mover a otra ubicación. Las tablas de catálogos del sistema contienen metadatos sobre tabla definidas por el usuario. Por ejemplo, si elimina archivos de una tabla definida por el usuario sin suprimir la entrada correspondiente en las tablas de catálogos, esta acción ocasionará una incoherencia. Una base de datos DB2 Everyplace debe contener las tablas de catálogos del sistema siguientes:

- DB2eSYSTABLES
- DB2eSYSCOLUMNS
- DB2eSYSRELS
- DB2eSYSUSERS (se crea esta tabla si se utiliza cifrado local de los datos)

Para acceder a las tablas de catálogos de una consulta, deberá utilizar los identificadores delimitados. Por ejemplo, se devuelve la consulta siguiente en el caso de que exista una tabla T:

```
SELECT 1 FROM "DB2eSYSTABLES" WHERE TNAME = 'T'
```

Consulta relacionada:

- Capítulo 19, "Tablas base del catálogo del sistema DB2 Everyplace", en la página 355

Manejo de conflictos de denominación

Procedimiento:

© Copyright IBM Corp. 1999,2003

En este tema se muestran algunos ejemplos de las maneras en que se pueden manejar los conflictos de denominación de archivos para las tablas definidas por el usuario. Suponga que una aplicación ejecuta la sentencia CREATE TABLE siguiente:

```
CREATE TABLE T (PK INT NOT NULL PRIMARY KEY, A INT)
```

Una vez ejecutada esta sentencia, DB2 Everyplace creará los dos archivos siguientes para la tabla *T*:

- DSY_T (datos)
- DSY_iT (índices)

Si crea otra tabla y utiliza el nombre *iT*, DB2 Everyplace creará dos archivos adicionales: DSY_iT (datos) y DSY_iiT (índices). El archivo de índices de la tabla *T* y el archivo de datos de la tabla *iT* tienen un conflicto porque tienen el mismo nombre. Ambos archivos se denominan DSY_iT. Para evitar este problema, DB2 Everyplace da soporte a la correlación de nombres de archivo. Es decir, que DB2 Everyplace creará y gestionará por completo los nombres de archivo. Para utilizar esta características, las aplicaciones deben establecer el atributo de conexión y éste se tiene que ejecutar antes de que se cree la primera tabla. Por ejemplo, en la CLI:

```
SQLSetConnectAttr(hdbc, SQL_ATTR_FILENAME_FORMAT,
                (SQLPOINTER)SQL_FILENAME_FORMAT_83, 0)
```

O en el Procesador de línea de mandatos:

```
DISABLE LONG FILENAME
```

Una vez ejecutado este mandato y creada la primera tabla, los archivos resultantes serán para la tabla *T*:

- 0001.DBd
- 0001.DBi

Conceptos afines:

- “Visión general de las tablas de base de datos de DB2 Everyplace” en la página 65
- “Serialización de conexiones” en la página 67

Tareas afines:

- “Cómo conectarse a la base de datos de DB2 Everyplace”

Cómo conectarse a la base de datos de DB2 Everyplace

Muchas veces, las aplicaciones requieren la creación de tablas en una ubicación específica, y el acceso a las mismas, como por ejemplo la carpeta `C:\TEMP` en las plataformas Win32. Para conectar con la base de datos especificando la vía de acceso de la misma, puede utilizar la llamada siguiente a la CLI:

```
rc = SQLConnect(hdbc, "C:\\TEMP\\bd", SQL_NTS, uid, SQL_NTS, pwd, SQL_NTS);
```

donde *bd* es el nombre de la base de datos (que DB2 Everyplace ignora).

Además, puede utilizar esta llamada a la CLI:

```
rc = SQLConnect(hdbc, "C:\\TEMP\\", SQL_NTS, uid, SQL_NTS, pwd, SQL_NTS);
```

La conexión con memoria ampliada en dispositivos o almacenes de objetos de WinCE requiere una especificación especial de la vía de acceso.

- Para Sony Memory Stick en PalmOS

```
rc = SQLConnect(hdbc, "#0:\\", SQL_NTS, uid, SQL_NTS, pwd, SQL_NTS);
```

- Para WinCE Object Store

```
rc = SQLConnect(hdbc, "@:\\", SQL_NTS, uid, SQL_NTS, pwd, SQL_NTS);
```

Mediante el Procesador de línea de mandatos, los usuarios pueden conectar con una ubicación específica utilizando el mandato ″CONNECT TO″. Por ejemplo, los mandatos siguientes conectarán con la base de datos que se conserva en la carpeta C:\TEMP\ de una plataforma Win32:

```
CONNECT TO C:\TEMP\
```

Conceptos afines:

- “Visión general de las tablas de base de datos de DB2 Everyplace” en la página 65

Tareas afines:

- “Manejo de conflictos de denominación” en la página 65

Serialización de conexiones

Sólo se puede mantener una conexión con una base de datos DB2 Everyplace cada vez. En todas las plataformas *excepto* Palm, DB2 Everyplace da soporte a la serialización de conexiones. Con la serialización de conexiones, las peticiones de conexión con una fuente de datos se ponen en serie (serializan). Sólo se puede realizar una conexión activa con una fuente de datos determinada cada vez. Las otras peticiones de conexión se ponen en una cola. El período de tiempo de espera se puede establecer mediante el atributo SQL_ATTR_LOGIN_TIMEOUT de la función SQLSetConnectAttr(). Los escenarios típicos de la serialización incluyen:

- Varios procesos que intentan abrir conexiones con una sola fuente de datos se serializan; una fuente de datos determinada sólo puede estar abierta para un proceso cada vez.
- Varios pasos de un solo proceso que intentan abrir conexiones con una sola fuente de datos se serializan.
- Varios procesos diferentes pueden tener abierta, cada uno de ellos, una conexión con varias fuentes de datos distintas simultáneamente.

Como ejemplo, las llamadas a CLI /JDBC siguientes establecerán el tiempo de espera de conexión en 10. Es decir, que la aplicación recibirá un error si existe otra conexión con la misma base de datos.

Para la **CLI:**

```
int i = 10; // tiempo de espera de 10 segundos
rc = SQLSetConnectAttr(hdbc, SQL_ATTR_LOGIN_TIMEOUT, (SQLPOINTER) i, 0);
```

Para **JDBC:**

```
int waitTime = 10;
String url = "jdbc:db2e:mysample";

Properties prop = new Properties();
prop.setProperty("LOGIN_TIMEOUT", Integer.toString(waitTime));

Connection con = driver.connect(url,prop);
```

Notas:

1. El tiempo de espera por omisión es de 0 segundos.
2. Una aplicación de varios pasos puede conectar con una base de datos utilizando un paso y desconectar de la base de datos utilizando otro paso.

3. Cuando una aplicación conecte satisfactoriamente con una base de datos, se creará un archivo llamado 'DSYLOCK'. Si el proceso de la aplicación termina de forma anómala, el archivo DSYLOCK se eliminará automáticamente.
4. Limitación: es posible que la serialización de conexiones no funcione para bases de datos que residan en la unidad de la red.
5. En un programa JDBC, el valor del tiempo de espera se ignorará y se establecerá en cero si se pasa en una propiedad al método DriverManager.getConnection().

Tareas afines:

• "Cómo conectarse a la base de datos de DB2 Everyplace" en la página 66

Bases de datos DB2 Everyplace en soportes de almacenamiento de sólo lectura

La base de datos DB2 Everyplace y una aplicación se pueden ejecutar directamente desde un soporte de almacenamiento de sólo lectura, como por ejemplo los CD-ROM o chips ROM de dispositivos incorporados. Por ejemplo, una aplicación de ejemplo de un catálogo trimestral de productos se puede distribuir en un CD-ROM a los representantes comerciales. Cada trimestre, los representantes comerciales reciben un CD-ROM que contiene el catálogo de productos completo de la empresa y una aplicación DB2 Everyplace para examinar, visualizar y consultar información sobre los productos que se ajustan a unas necesidades específicas de los clientes. La aplicación DB2 Everyplace se ejecutará directamente desde el CD-ROM sin necesidad de instalarla antes en una estación de trabajo.

Cuando DB2 Everyplace detecta que se está ejecutando una aplicación en un soporte de almacenamiento de sólo lectura (o que los archivos están protegidos contra grabación), se establece en modalidad de sólo lectura. En esta modalidad, las sentencias de actualización, inserción, supresión, creación y eliminación están inhabilitadas y devolverán un error. Observe que, para algunas consultas de selección, DB2 Everyplace crea archivos y tablas temporales. Estos archivos y tablas se crean en el directorio temporal por omisión. En plataformas Win32, designa este directorio la variable de entorno *TEMP*. Si la variable de entorno *TEMP* no existe, también se puede designar *TMP*. En Linux, se utiliza el directorio `/tmp/`.

Sólo se da soporte a esta característica en plataformas Win32 y Linux.

Capítulo 9. Recuperación gradual de datos por medio de CLI

Hay dos formas en las que un usuario recupera datos procedentes de una tabla de DB2 Everyplace por medio de CLI:

- La aplicación puede optar por asignar la memoria máxima que puede ocupar el valor de columna (basándose en su conocimiento de una columna en el conjunto de resultados por medio de *SQLDescribeCol()* o un conocimiento previo) y vincularlo por medio de *SQLBindCol()*.
- La aplicación puede llamar a *SQLGetData()* para obtener datos de columna en el almacenamiento intermedio asignado por la aplicación.

En el caso de los datos binarios (BLOB) o datos de carácter (CHAR o VARCHAR), la columna puede ser muy larga. Es posible que el desarrollador de aplicaciones no desee asignar un almacenamiento intermedio que sea lo suficientemente grande como para que contenga toda la columna o que no pueda permitirse asignar un almacenamiento intermedio tan grande. Adicionalmente, en algunos casos la aplicación sólo requiere algunos fragmentos de la columna. En estos escenarios, se necesita la recuperación gradual de datos.

Procedimiento:

Una característica de *SQLGetData()* permite a la aplicación utilizar llamadas repetidas para obtener, en secuencia, el valor de una única columna en fragmentos más manejables. Esencialmente, una llamada a *SQLGetData()* devuelve SQL_SUCCESS_WITH_INFO (con SQLSTATE 01004) para indicar que hay más datos para esta columna. *SQLGetData()* se llama repetidamente para obtener los restantes fragmentos de datos hasta que devuelva SQL_SUCCESS, lo que significa que se han recuperado todos los datos para esta columna.

Sintaxis:

```
SQLRETURN   SQLGetData       (
                SQLHSTMT          StatementHandle,   /* hstmt */
                SQLUSMALLINT      ColumnNumber,      /* icol */
                SQLSMALLINT       TargetType,        /* fCType */
                SQLPOINTER        TargetValuePtr,    /* rgbValue */
                SQLINTEGER        BufferLength,      /* cbValueMax */
                SQLINTEGER   *FAR StrLen_or_IndPtr); /* pcbValue */
```

Esta acción recuperará *BufferLength* bytes a la vez y *StrLen_or_IndPtr* indica el número de bytes *restantes*. El valor de retorno de la función es SQL_SUCCESS_WITH_INFO (con SQLSTATE 01004) si quedan bytes. En caso contrario, si el valor de retorno es SQL_SUCCESS, *StrLen_or_IndPtr* indica el número de bytes que DB2 Everyplace CLI tiene disponibles para volver al almacenamiento intermedio de *TargetValuePtr*. *SQLGetData()* puede utilizarse de este modo para recuperar columnas largas en el caso de que el tipo de datos C (*TargetType*) sea SQL_C_CHAR, SQL_C_BINARY o en el caso de que *TargetType* sea SQL_C_DEFAULT y el tipo de columna denote un binario o serie de caracteres.

Para utilizar esta característica de *SQLGetData()*, en primer lugar debe establecer un atributo de sentencia SQL_ATTR_GETDATA_MODE para SQL_PIECEMEAL_DATA. El valor por omisión de este atributo es SQL_CHUNK_DATA. La diferencia entre estas dos modalidades es que, en la

© Copyright IBM Corp. 1999,2003

modalidad SQL_CHUNK_DATA (que es la modalidad por omisión) y el momento en que se produce el truncamiento, el valor de retorno de SQLGetData() *StrLen_or_IndPtr* indica el número *total* de bytes de esta columna y la segunda llamada continúa recuperando datos del principio de la columna.

Ejemplo de fragmento de código:

```
sqlrc = SQLSetStmtAttr(hstmt, SQL_ATTR_GETDATA_MODE, (SQLPOINTER) SQL_PIECEMEAL_DATA, 0);
    SQLCHAR * stmt = (SQLCHAR *) "SELECT blobColumn FROM t1 where c1 = ?";
    sqlrc = SQLPrepare( hstmt, stmt, SQL_NTS ) ;
    sqlrc = SQLBindParameter(hstmt, 1, SQL_PARAM_INPUT, SQL_C_CHAR, ...);
    sqlrc = SQLExecute( hstmt ) ;
    sqlrc = SQLFetch( hstmt );
    /* get BUFSIZ bytes at a time, bufInd indicates number of Bytes LEFT */
    sqlrc = SQLGetData (hstmt, 1, SQL_C_BINARY, (SQLPOINTER) buffer, BUFSIZ, &bufInd);
    while( sqlrc == SQL_SUCCESS_WITH_INFO ) {
        // manejar BUFSIZ bytes de datos blob en almacenamiento intermedio
            :
        sqlrc = SQLGetData (hstmt, 1, SQL_C_BINARY, (SQLPOINTER) buffer, BUFSIZ, &bufInd);
    }
    if (sqlrc == SQL_SUCCESS) {  /* partial buffer on last GetData */
        // manejar bufInd bytes de datos blob en almacenamiento intermedio
            :
    }
```

Capítulo 10. Marcadores de parámetros

Este capítulo proporciona información sobre la utilización de marcadores de parámetro en consultas DB2 Everyplace. Los temas que se tratan son:

- "Visión general de los marcadores de parámetro"
- "Ejemplos de utilización de marcador de parámetro"
- "Marcadores de parámetro soportados de DB2 Everyplace" en la página 76

Visión general de los marcadores de parámetro

Para las sentencias SQL que han de ejecutarse varias veces, a menudo resulta ventajoso preparar una vez la sentencia SQL y volver a utilizar el plan de consulta utilizando marcadores de parámetro para sustituir los valores de entrada durante el tiempo de ejecución.

En DB2 Everyplace, un marcador de parámetro se representa mediante un carácter ″?″ e indica el lugar en el va a sustituirse una variable de aplicación en una sentencia SQL. Se hace referencia a los marcadores de parámetro por un número y están numerados secuencialmente de izquierda a derecha, comenzando en el 1. La aplicación debe asociar un área de almacenamiento variable a cada marcador de parámetro especificado en la sentencia de SQL antes de ejecutarse ésta. Además de eso, las variables asociadas deben ser un área de almacenamiento válida y deben contener valores de datos de entrada cuando se ejecuta la sentencia preparada en la base de datos.

El ejemplo siguiente ilustra una sentencia SQL que contiene dos marcadores de parámetro.

```
SELECT * FROM customers WHERE custid = ? AND lastname = ?
```

Conceptos afines:

- "Ejemplos de utilización de marcador de parámetro"

Ejemplos de utilización de marcador de parámetro

DB2 Everyplace proporciona un rico conjunto de interfaces estándar incluyendo CLI/ODBC, JDBC y ADO.NET para acceder a los datos de modo eficaz. Los retazos de código de ejemplo siguientes muestran la utilización de una sentencia preparada con marcadores de parámetro para cada API de acceso a los datos.

Tome en consideración el esquema de tabla siguiente para la tabla t1, en el que la columna c1 es la clave primaria para la tabla t1.

Tabla 13. Esquema de tabla de ejemplo

Nombre de la columna	Tipo de datos de Everyplace	Anulable
c1	INTEGER	falso
c2	SMALLINT	verdadero
c3	CHAR(20)	verdadero
c4	VARCHAR(20)	verdadero

© Copyright IBM Corp. 1999,2003

Tabla 13. Esquema de tabla de ejemplo (continuación)

Nombre de la columna	Tipo de datos de Everyplace	Anulable
c5	DECIMAL(8,2)	verdadero
c6	DATE	verdadero
c7	TIME	verdadero
c8	TIMESTAMP	verdadero
c9	BLOB(30)	verdadero

Los ejemplos siguientes ilustran el modo de insertar una fila en la tabla t1 utilizando una sentencia preparada.

Ejemplo de CLI

```
void parameterExample1(void)
{
   SQLHENV henv;
   SQLHDBC hdbc;
   SQLHSTMT hstmt;
   SQLRETURN      rc;
   TCHAR server[] = _T("C:\\mysample\\");
   TCHAR uid[] = _T("db2e");
   TCHAR pwd[] = _T("db2e");
   long p1 = 10;
   short p2 = 100;
   TCHAR p3[100];
   TCHAR p4[100];
   TCHAR p5[100];
   TCHAR p6[100];
   TCHAR p7[100];
   TCHAR p8[100];
   char  p9[100];
   long len = 0;

   _tcscpy(p3, _T("data1"));
   _tcscpy(p4, _T("data2"));
   _tcscpy(p5, _T("10.12"));
   _tcscpy(p6, _T("2003-06-30"));
   _tcscpy(p7, _T("12:12:12"));
   _tcscpy(p8, _T("2003-06-30-17.54.27.710000"));

   memset(p9, 0, sizeof(p9));
   p9[0] = 'X';
   p9[1] = 'Y';
   p9[2] = 'Z';

   rc = SQLAllocEnv(&henv);
   // comprobar código de retorno ...

   rc = SQLAllocConnect(henv, &hdbc);
   // comprobar código de retorno ...

   rc = SQLConnect(hdbc, (SQLTCHAR*)server, SQL_NTS,
    (SQLTCHAR*)uid, SQL_NTS, (SQLTCHAR*)pwd, SQL_NTS);
   // comprobar código de retorno ...

   rc = SQLAllocStmt(hdbc, &hstmt);
```

```
// comprobar código de retorno ...

// preparar la sentencia
rc = SQLPrepare(hstmt, _T("INSERT INTO t1 VALUES (?,?,?,?,?,?,?,?,?)"), SQL_NTS);
// comprobar código de retorno ...

// asociar parámetros de entrada
rc = SQLBindParameter(hstmt, (unsigned short)1, SQL_PARAM_INPUT,
    SQL_C_LONG, SQL_INTEGER, 4, 0, &p1, sizeof(p1), &len);
// comprobar código de retorno ...

rc = SQLBindParameter(hstmt, (unsigned short)2, SQL_PARAM_INPUT, SQL_C_LONG,
    SQL_SMALLINT, 2, 0, &p2, sizeof(p2), &len);
// comprobar código de retorno ...

len = SQL_NTS;
rc = SQLBindParameter(hstmt, (unsigned short)3, SQL_PARAM_INPUT, SQL_C_TCHAR,
    SQL_CHAR, 0, 0, &p3[0], 100, &len);
// comprobar código de retorno ...

rc = SQLBindParameter(hstmt, (unsigned short)4, SQL_PARAM_INPUT, SQL_C_TCHAR,
    SQL_VARCHAR, 0, 0, &p4[0], 100, &len);
// comprobar código de retorno ...

rc = SQLBindParameter(hstmt, (unsigned short)5, SQL_PARAM_INPUT, SQL_C_TCHAR,
    SQL_DECIMAL, 8, 2, &p5[0], 100, &len);
// comprobar código de retorno ...

rc = SQLBindParameter(hstmt, (unsigned short)6, SQL_PARAM_INPUT, SQL_C_TCHAR,
    SQL_TYPE_DATE, 0, 0, &p6[0], 100, &len);
// comprobar código de retorno ...

rc = SQLBindParameter(hstmt, (unsigned short)7, SQL_PARAM_INPUT, SQL_C_TCHAR,
    SQL_TYPE_TIME, 0, 0, &p7[0], 100, &len);
// comprobar código de retorno ...

rc = SQLBindParameter(hstmt, (unsigned short)8, SQL_PARAM_INPUT, SQL_C_TCHAR,
    SQL_TYPE_TIMESTAMP, 0, 0, &p8[0], 100, &len);
// comprobar código de retorno ...

len = 3;
rc = SQLBindParameter(hstmt, (unsigned short)9, SQL_PARAM_INPUT, SQL_C_BINARY,
    SQL_BINARY, 0, 0, &p9[0], 100, &len);
// comprobar código de retorno ...

// ejecutar la sentencia preparada
rc = SQLExecute(hstmt);
// comprobar código de retorno ...

rc = SQLFreeStmt(hstmt, SQL_DROP);
// comprobar código de retorno ...

rc = SQLDisconnect(hdbc);
// comprobar código de retorno ...

rc = SQLFreeConnect(hdbc);
// comprobar código de retorno ...

rc = SQLFreeEnv(henv);
```

```
   // comprobar código de retorno ...

}
```

Ejemplo de JDBC

```
public static void parameterExample1() {

   String driver = "com.ibm.db2e.jdbc.DB2eDriver";
   String url    = "jdbc:db2e:mysample";
   Connection conn = null;
   PreparedStatement pstmt = null;

 try
   {
      Class.forName(driver);

      conn = DriverManager.getConnection(url);

      // preparar la sentencia
      pstmt = conn.prepareStatement("INSERT INTO t1 VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)");

      // asociar los parámetros de entrada
      pstmt.setInt(1, 1);
      pstmt.setShort(2, (short)2);
      pstmt.setString(3, "data1");
      pstmt.setString(4, "data2");
      pstmt.setBigDecimal(5, new java.math.BigDecimal("12.34"));
      pstmt.setDate(6, new java.sql.Date(System.currentTimeMillis() ) );
      pstmt.setTime(7,  new java.sql.Time(System.currentTimeMillis() ) );
      pstmt.setTimestamp (8,  new java.sql.Timestamp(System.currentTimeMillis() ) );
      pstmt.setBytes(9, new byte[] { (byte)'X', (byte)'Y', (byte)'Z' } );

      // ejecutar la sentencia
      pstmt.execute();

      pstmt.close();

      conn.close();
   }
   catch (SQLException sqlEx)
   {
      while(sqlEx != null)
      {
          System.out.println("SQLERROR: \n" + sqlEx.getErrorCode() +
             ", SQLState: " + sqlEx.getSQLState() +
             ", Message: " + sqlEx.getMessage() +
             ", Vendor: " + sqlEx.getErrorCode() );
         sqlEx = sqlEx.getNextException();
      }
   }
   catch (Exception ex)
   {
      ex.printStackTrace();
   }
}
```

Ejemplo de ADO.NET

[C#]

```
public static void ParameterExample1()
{
   DB2eConnection conn = null;
   DB2eCommand cmd  = null;
   String connString   = @"database=.\; uid=db2e; pwd=db2e";
   int i = 1;

 try
   {
      conn = new DB2eConnection(connString);

      conn.Open();

      cmd = new DB2eCommand("INSERT INTO t1 VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)", conn);

      // preparar el mandato
      cmd.Prepare();

      // asociar los parámetros de entrada
      DB2eParameter p1 = new DB2eParameter("@p1", DB2eType.Integer);
      p1.Value = ++i;
      cmd.Parameters.Add(p1);

      DB2eParameter p2 = new DB2eParameter("@p2", DB2eType.SmallInt);
      p2.Value = 100;
      cmd.Parameters.Add(p2);

      DB2eParameter p3 = new DB2eParameter("@p3", DB2eType.Char);
      p3.Value = "data1";
      cmd.Parameters.Add(p3);

      DB2eParameter p4 = new DB2eParameter("@p4", DB2eType.VarChar);
      p4.Value = "data2";
      cmd.Parameters.Add(p4);

      DB2eParameter p5 = new DB2eParameter("@p5", DB2eType.Decimal);
      p5.Value = 20.25;
      cmd.Parameters.Add(p5);

      DB2eParameter p6 = new DB2eParameter("@p6", DB2eType.Date);
      p6.Value = DateTime.Now;
      cmd.Parameters.Add(p6);

      DB2eParameter p7 = new DB2eParameter("@p7", DB2eType.Time);
      p7.Value = new TimeSpan(23, 23, 23);
      cmd.Parameters.Add(p7);

      DB2eParameter p8 = new DB2eParameter("@p8", DB2eType.Timestamp);
      p8.Value = DateTime.Now;
      cmd.Parameters.Add(p8);

      byte []barr = new byte[3];
      barr[0] = (byte)'X';
      barr[1] = (byte)'Y';
      barr[2] = (byte)'Z';

      DB2eParameter p9 = new DB2eParameter("@p9", DB2eType.Blob);
      p9.Value = barr;
      cmd.Parameters.Add(p9);
```

```
      // ejecutar el mandato preparado
      cmd.ExecuteNonQuery();
   }
   catch (DB2eException e1)
   {
      for (int i=0; i < e1.Errors.Count; i++)
      {
         Console.WriteLine("Error #" + i + "\n" +
            "Message: " + e1.Errors[i].Message + "\n" +
            "Native: " + e1.Errors[i].NativeError.ToString() + "\n" +
            "SQL: " + e1.Errors[i].SQLState + "\n");
      }
   }
   catch (Exception e2)
   {
       Console.WriteLine(e2.Message);
   }
     finally
   {
      if (conn != null && conn.State != ConnectionState.Closed)
      {
            conn.Close();
            conn = null;
      }
   }
}
```

Conceptos afines:

- "Visión general de los marcadores de parámetro" en la página 71

Marcadores de parámetro soportados de DB2 Everyplace

Un marcador de parámetro, que se representa mediante un símbolo de final de interrogación (?), es un espacio reservado en una sentencia de SQL cuyo valor se obtiene durante la ejecución de la sentencia. Una aplicación utiliza la función `SQLBindParameter()` para asociar marcadores de parámetros de enlace con variables de aplicación. Durante la ejecución de las funciones de CLI de DB2 `SQLExecute()` y `SQLExecDirect()`, los valores de estas variables sustituyen a cada uno de los marcadores de parámetro respectivos. Durante el proceso puede tener lugar una conversión de datos. Vea la Tabla 95 en la página 286 para obtener más información sobre las conversiones soportadas de tipos de datos.

DB2 Everyplace sólo da soporte a los marcadores de parámetros no tipificados, que pueden utilizarse en determinadas posiciones de una sentencia de SQL. La Tabla 14 lista las restricciones sobre la utilización de los marcadores de parámetros.

Tabla 14. Restricciones sobre la utilización de los marcadores de parámetros

Ubicación de los marcadores de parámetros no tipificados	Tipo de datos
Expresión: Solo en una lista de selección	Error
Expresión: Ambos operandos de un operador aritmético	Error
Predicado: Operando de la parte izquierda de un predicado IN	Error

Tabla 14. Restricciones sobre la utilización de los marcadores de parámetros (continuación)

Ubicación de los marcadores de parámetros no tipificados	Tipo de datos
Predicado: Ambos operandos de un operador relacional	Error
Función: Operando de una función de agregación	Error

Consulta relacionada:

- "Visión general del soporte de sentencia de SQL de DB2 Everyplace" en la página 137
- "Compatibilidad entre tipos de datos para las operaciones de asignación y comparación" en la página 176
- "Listado de los SQLSTATE" en la página 180
- "Resumen de códigos de clase de SQLState" en la página 180

Marcadores de parámetros

Capítulo 11. Comportamiento del cursor en el contexto de una conexión

Cursor de lectura general en conflictos de grabación procedentes de otro descriptor de sentencia:

Una aplicación puede tener varios descriptores de sentencia que efectúen operaciones de lectura y grabación en la misma tabla al mismo tiempo. Los conflictos se producen cuando un descriptor está efectuando una operación de grabación en la tabla (por ejemplo, UPDATE, DELETE o INSERT) al tiempo que otro descriptor está en medio de una operación de lectura o grabación. En DB2 Everyplace el cursor de lectura es estable y está siempre leyendo los datos más actuales. Sobrevive a los conflictos de grabación, sin tener en cuenta si está utilizando o no un índice. Por ejemplo, suponga que una aplicación tiene dos descriptores de sentencia. El descriptor número 1 se utiliza para buscar filas de una tabla T, en tanto que el descriptor número 2 se utiliza para suprimir algunas filas de la misma tabla. Es probable que cada descriptor se haya creado por medio de hebras diferentes (por ejemplo, en un entorno de hebra de Java).

A continuación se muestra un escenario posible:

```
// Buscar 2 filas en la tabla T
   Statement handle 1: execute "SELECT A FROM T WHERE primary_key < 10"
   Statement handle 1: fetch one row; fetch another row
    // Suprimir algunas filas de la tabla T
   Statement handle 2: prepare "DELETE FROM T WHERE primary_key = ?"
   Statement handle 2: execute
    // Seguir buscando una fila más en T
   Statement handle 1: fetch one row
```

En este punto de la ejecución, el descriptor de sentencia número 1 podrá seguir buscando la fila siguiente (si existe), sin tener en cuenta si se utiliza o no un índice. En el escenario anterior, *se* utiliza un índice debido a que hay una clave primaria. La idea es que DB2 Everyplace intentará volver a situar la posición del cursor del descriptor número 1, utilizando su posición actual, antes de avanzar. Si la posición actual ya no existe (por ejemplo, otro descriptor de sentencia suprimió la fila), el cursor simplemente avanza hacia la posición siguiente efectuando una búsqueda. Del mismo modo, si otro descriptor de sentencia ha suprimido la posición siguiente, el cursor podrá saltar sobre el ″espacio″ hacia la posición siguiente.

Cursor desplazable en conflictos de grabación procedentes de otro descriptor de sentencia:

Piense en un ejemplo parecido al del apartado anterior, pero en el que el cursor de lectura es un cursor desplazable. Si es un cursor desplazable ″insensible″, esto no será un problema debido a que por definición el conjunto de resultados no cambia. Si el cursor no es ″insensible″, su comportamiento coincidirá con el cursor de lectura regular descrito anteriormente. En esencia, el comportamiento del cursor de lectura después del conflicto es que el conjunto de resultados vuelva a calcularse con arreglo a los datos de tabla más actuales y que se mantenga el comienzo del conjunto de filas actual. El cursor se avanza hacia la fila siguiente en el caso de que se suprima la fila actual.

El ejemplo siguiente ilustra el caso con un cursor desplazable utilizando CLP. Suponga que la tabla T tiene seis filas:

© Copyright IBM Corp. 1999,2003

```
create table T (a int, b int)
     create index idx1 on T(a)
     insert into T values (1, 1)
     insert into T values (2, 2)
     insert into T values (3, 1)
     insert into T values (3, 2)
     insert into T values (3, 3)
     insert into T values (4, 4)
```

Abusando de su generosidad, piense en un ejemplo en el que la aplicación tuviera dos descriptores de sentencia, uno para la lectura y el otro para la supresión.

```
Statement handle 1: enable scrollable cursor;
Statement handle 1: execute "SELECT A FROM T WHERE a < 10"
Statement handle 2: prepare "DELETE FROM T WHERE a = ?"
Statement handle 1: fetchscroll with SQL_FETCH_FIRST
-- get (1, 1)
Statement handle 1: fetchscroll with SQL_FETCH_NEXT
-- get (2, 2)
Statement handle 1: fetchscroll with SQL_FETCH_NEXT
-- get (3, 1)
Statement handle 2: execute
--- suppose delete row (2, 2)
Statement handle 1: fetchscroll with SQL_FETCH_NEXT
-- re-compute previous rows, and return (3, 2)
Statement handle 1: fetchscroll with SQL_FETCH_PRIOR
-- get (3, 1)
Statement handle 1: fetchscroll with SQL_FETCH_PRIOR
-- get (1, 1)  note that (2, 2) is gone
Statement handle 1: fetchscroll with SQL_FETCH_ABSOLUTE, offset 2
-- get (3, 1)  note that (2, 2) is gone
Statement handle 1: fetchscroll with SQL_FETCH_ABSOLUTE, offset 5
-- get (4, 4)
```

Cursor en confirmación y retrotracción, incluyendo la modalidad de confirmación automática:

Sin tener en cuenta la modalidad de transacción o confirmación automática, un cursor abierto permanece abierto en la confirmación y un cursor abierto se cierra en la retrotracción.

Dependencia del objeto:

Preparar una sentencia SQL por medio de un descriptor de sentencia H puede que produzca cierta dependencia de determinados objetos. Por ejemplo, seleccionar filas de una tabla T por medio de un Idx de índice requiere la existencia de la tabla T y del Idx de índice. Si otro descriptor de sentencia ha suprimido estos objetos (por ejemplo, si se ha descartado el Idx de índice), volver a ejecutar la sentencia por medio de H forzará una recompilación de la sentencia SQL. Como consecuencia, puede que el plan de consulta sea diferente o que se devuelva un error.

Capítulo 12. Cifrado de los datos locales

En este capítulo se explica cómo cifrar datos locales en una base de datos de DB2 Everyplace. Se tratan los temas siguientes:

- "Visión general del cifrado local de los datos"
- "Establecimiento de una conexión con la base de datos DB2 Everyplace" en la página 82
- "Cómo otorgar privilegios de cifrado a un usuario" en la página 83
- "Creación de una tabla cifrada" en la página 84
- "Gestión de los privilegios de cifrado" en la página 84
- "Cifrado utilizando el DB2eCLP" en la página 85

Visión general del cifrado local de los datos

El cifrado en DB2 Everyplace está diseñado para proteger los datos de un dispositivo portátil o incorporado. En este tema se proporciona una rápida visión general del cifrado local de los datos como ayuda a la iniciación. Se tratan los temas siguientes:

- ¿Porqué se debe utilizar el cifrado local de datos?
- Objetivos de un cifrado local de los datos
- Creación de la primera tabla cifrada
- Accesos posteriores a tablas cifradas
- Gestión de los privilegios de usuario

¿Porqué se debe utilizar el cifrado local de los datos?:

Piense en una aplicación corporativa de ventas que contiene datos de contacto con los clientes. Un viajante de comercio puede llevar estos datos en su PDA cuando visite a un cliente. A menos que la aplicación o PDA brinden un sistema de almacenamiento seguro, se puede acceder fácilmente a los datos a través de la aplicación o investigando el sistema de archivos nativo del dispositivo portátil. Los datos sensibles al cifrado se convierten en un aspecto crucial en la protección de la información corporativa.

Objetivos de un cifrado local de los datos:

DB2 Everyplace proporciona una solución que permite que una aplicación implante una política de seguridad corporativa. El primer objetivo consiste en cifrar la información secreta o delicada almacenada en tablas de DB2 Everyplace. Los datos se cifran utilizando métodos de cifrado estándares tales como DES, que implanta claves de cifrado. El segundo objetivo consiste en proporcionar un marco seguro para poder gestionar las claves utilizadas para cifrar los datos. Se requiere al usuario para que suministre un ID de usuario y una contraseña en el momento de conectar con la base de datos. Para obtener más información, consulte el apartado "Gestión de los privilegios de cifrado" en la página 84.

Para obtener más información sobre cómo utilizar el cifrado de datos, consulte el apartado "Cifrado utilizando el DB2eCLP" en la página 85.

Requisitos previos:

© Copyright IBM Corp. 1999,2003

En este apartado se describe cómo se habilita el cifrado para cada plataforma y se relacionan las bibliotecas necesarias, además de las que necesita la base de datos DB2 Everyplace.

Para Win32:

- biblioteca de plug-in: CryptoPlugin.dll (proporcionada por DB2 Everyplace)
- biblioteca de cifrado: Crypt32.dll (paquete de 128 bits Cypher Strength Encryption, se suministra con IE5.5 o superiores). Vaya a http://www.microsoft.com/windows/ie/downloads/critical /q313675/download.asp.

Para Windows CE/Pocket PC

- biblioteca de plug-in: CryptoPlugin.dll (proporcionada por DB2 Everyplace)
- biblioteca de cifrado: Microsoft High Encryption Pack para Pocket PC V1.0. Vaya a http://www.microsoft.com/mobile/pocketpc/downloads/ssl128.asp.

Para Palm OS

- biblioteca de plug-in : CryptoPlugin.PRC (proporcionada por DB2 Everyplace)
- biblioteca de cifrado: PBSPKcs11.prc (proporcionada por DB2 Everyplace)

Para Linux/Neutrino

- biblioteca de plug-in: libcryptoplugin.so (proporcionada por DB2 Everyplace)
- biblioteca de cifrado: libpvcpkcs11.so (proporcionada por DB2 Everyplace)

Para Symbian

- biblioteca de plug-in: CRYPTOPLUGIN.DLL (proporcionada por DB2 Everyplace)
- biblioteca de cifrado: ECSPKCS11.DLL (proporcionada por DB2 Everyplace)

Procedimiento:

Para utilizar el cifrado de datos:

1. Establecimiento de una conexión con la base de datos DB2 Everyplace.
2. Otorgar privilegios de cifrado a un usuario
3. Cree la primera tabla cifrada.

   Acceso posterior a las tablas cifradas: Si una base de datos contiene la tabla DB2eSYSUSERS, cualquier conexión de base de datos posterior pasará por la autentificación de usuario con el ID de usuario y la contraseña suministrados. Si falla la autentificación, la aplicación sólo podrá acceder a tablas no cifradas. La aplicación no podrá crear nuevas tablas cifradas, eliminar tablas cifradas existentes ni acceder o actualizar datos cifrados.
4. Gestionar privilegios de cifrado.

Establecimiento de una conexión con la base de datos DB2 Everyplace

Esta tarea forma parte de la tarea más importante de Cifrado de datos locales. Después de completar estos pasos, vuelva al apartado “Visión general del cifrado local de los datos” en la página 81.

Procedimiento:

Cualquier interacción con la base de datos DB2 Everyplace requiere que se establezca una conexión. Además, y para que un usuario pueda acceder a tablas cifradas o crearlas, la aplicación debe conectar con DB2 Everyplace con un ID de usuario y una contraseña que no estén vacíos, utilizando la función siguiente de la CLI:

```
rc = SQLConnect(hdbc, "C:\temp\", SQL_NTS, "usuario1", SQL_NTS, "contras1", SQL_NTS)
```

donde *"C:\temp\"* es el directorio de la base de datos con la que se conecta la aplicación, utilizando el ID de usuario *"usuario1"* y la contraseña *"contras1"*.

Para una interfaz JDBC, se puede establecer una conexión de base de datos de forma parecida.

Conceptos afines:

- "Serialización de conexiones" en la página 67

Cómo otorgar privilegios de cifrado a un usuario

Esta tarea forma parte de la tarea más importante de Cifrado de datos locales. Después de completar estos pasos, vuelva al apartado "Visión general del cifrado local de los datos" en la página 81.

Procedimiento:

Antes de crear la primera tabla cifrada, la aplicación debe otorgar privilegios de cifrado a un usuario. Por ejemplo, la aplicación puede emitir la sentencia siguiente de SQL:

```
rc = SQLExecDirect(..., "GRANT ENCRYPT ON DATABSE TO \" usuario1\"" +
                         " using \"pwd1\" new \"pwd1\"", SQL_NTS)
```

Al ejecutar esta sentencia de SQL, DB2 Everyplace creará una tabla de catálogos del sistema llamada DB2eSYSUSERS, y se insertará una fila en dicha tabla. Esto significa que el usuario *"usuario1* ya estará registrado con la contraseña correspondiente y ahora tendrá todos los privilegios de cifrado, como por ejemplo para crear tablas cifradas y acceder a ellas.

Esta tabla está fuertemente vinculada a la base de datos y a los datos cifrados, por lo que no basta con moverla a otra base de datos DB2 Everyplace para acceder a datos cifrados. Esto es debido a que las distintas bases de datos tendrán claves diferentes para el cifrado o descifrado. Como consecuencia, si una persona tiene permitido acceder a tablas cifradas de una base de datos, dicha persona no puede acceder a otra base de datos utilizando un ID de usuario y una contraseña iguales. Al igual que con otras tablas de catálogos del sistema, una aplicación puede recuperar filas utilizando la sentencia de selección de SQL, pero no puede modificar los datos de esta tabla mediante las sentencias INSERT, DELETE, UPDATE, CREATE y DROP.

Conceptos afines:

- "Cifrado utilizando el DB2eCLP" en la página 85

Tareas afines:

- "Cifrado utilizando el DB2eCLP" en la página 85
- "Creación de una tabla cifrada" en la página 84
- "Gestión de los privilegios de cifrado" en la página 84

Creación de una tabla cifrada

Esta tarea forma parte de la tarea más importante de Cifrado de datos locales. Después de completar estos pasos, vuelva al apartado "Visión general del cifrado local de los datos" en la página 81.

Procedimiento:

Una vez que se ha establecido una conexión con la base de datos DB2 Everyplace y se han otorgado privilegios de cifrado a un usuario, la aplicación puede crear tablas cifradas mediante una sentencia CREATE TABLE ampliada. Por ejemplo, se puede crear la tabla de empleados siguiente:

```
SQLExecDirect(..., "CREATE TABLE EMPLOYEES (EMPNO INT PRIMARY KEY, NAME VARCHAR(30),
SALARY DECIMAL(10,2)) WITH ENCRYPTION", SQL_NTS)
```

Conceptos afines:

- "Cifrado utilizando el DB2eCLP" en la página 85

Tareas afines:

- "Cifrado utilizando el DB2eCLP" en la página 85
- "Cómo otorgar privilegios de cifrado a un usuario" en la página 83
- "Gestión de los privilegios de cifrado"

Gestión de los privilegios de cifrado

Esta tarea forma parte de la tarea más importante de Cifrado de datos locales. Después de completar estos pasos, vuelva al apartado "Visión general del cifrado local de los datos" en la página 81.

Procedimiento:

Una vez que una aplicación conecta con una base de datos con el ID de usuario y la contraseña autentificados, la aplicación puede crear nuevos usuarios, cambiar contraseñas o eliminar del sistema un usuario registrado. La sintaxis para crear un nuevo usuario o cambiar una contraseña es:

```
GRANT ENCRYPT ON DATABASE TO "nuevousuario"
      USING "contrasotorgante" NEW "nuevacontras"
```

La sintaxis para eliminar un usuario registrado es:

```
REVOKE ENCRYPT ON DATABASE FROM "usuario"
```

Nota: Si se eliminan todos los usuarios registrados de la tabla DB2eSYSUSERS (mediante la sentencia REVOKE), no se puede realizar ninguna otra operación de cifrado, incluido el acceso a una tabla cifrada existente. No existe ningún mecanismo de recuperación.

Conceptos afines:

- "Cifrado utilizando el DB2eCLP" en la página 85

Tareas afines:

- "Cifrado utilizando el DB2eCLP" en la página 85
- "Creación de una tabla cifrada"
- "Cómo otorgar privilegios de cifrado a un usuario" en la página 83

Cifrado utilizando el DB2eCLP

Este apartado contiene un ejemplo de sesión interactiva diseñada para mostrar cómo utilizar el cifrado de datos en las aplicaciones. Se han añadido comentarios para explicar cada operación.

```
-- Cifrado utilizando DB2eCLP
--
-- Ésta es una sesión de cifrado de ejemplo utilizando el programa de
-- interfaz de línea de mandatos de ejemplo suministrado, DB2eCLP.
--
-- Sólo mostramos el código de retorno de una sentencia si ésta ha
-- fallado; si ha finalizado satisfactoriamente, sólo mostramos los
-- resultados de las selecciones.
-- Los mandatos que se pueden escribir en DB2 Everyplace tiene por
-- prefijo la serie "CLP:> ".
--
-- -- (CLI:SQLConnect, SQL:CREATE TABLE, SQL:GRANT, SQL:REVOKE)
--
-- Cuando se inicia DB2eCLP, automáticamente se conecta con
-- la base de datos por omisión (en el directorio actual).
-- Esto es equivalente a:
--
CLP:> CONNECT TO anything;

-- Puesto que no se indica ninguna vía de acceso específica, sólo
-- el nombre "anything", conecta con el directorio actual.
--
-- Ahora crearemos una tabla no cifrada que contenga una correlación
-- de algunos números en palabras de contaje en sueco.

CLP:> CREATE TABLE swedish(nummer INT, ord VARCHAR(32));
CLP:> INSERT INTO swedish VALUES(1, 'ett');
CLP:> INSERT INTO swedish VALUES(3, 'tre');
CLP:> INSERT INTO swedish VALUES(4, 'fyra');
CLP:> INSERT INTO swedish VALUES(5, 'fem');
CLP:> INSERT INTO swedish VALUES(7, 'sju');
CLP:> INSERT INTO swedish VALUES(99, 'nittionio');

-- Eche un vistazo a los datos
CLP:> SELECT * FROM swedish;

NUMMER      ORD
----------- --------------------------------
          1 ett
          3 tre
          4 fyra
          5 fem
          7 sju
         99 nittionio
6 row(s) returned.

-- Ahora intentaremos crear la tabla correspondiente para inglés,
-- pero utilizando cifrado.
--
CLP:> CREATE TABLE english(number INT, word VARCHAR(32)) WITH ENCRYPTION;
Statement failed [sqlstate = 42501].

-- Esta sentencia falla porque todavía no estamos autorizados, tal como ha
-- indicado el código de error. Por tanto, tenemos que volver a conectar:
--
CLP:> CONNECT TO something USER jsk USING hemligt;

-- Este mandato conecta con la misma base de datos (directorio por
-- omisión/actual) pero con una identidad de usuario específica "jsk"
-- y utilizando la contraseña "hemligt".
-- El mandato CONNECT TO no es una sentencia de SQL, por lo que lo
-- interpreta la aplicación DB2eCLP. Ésta desconectará y conectará
-- de nuevo con la base de datos DB2 Everyplace utilizando:
--    SQLConnect(hdbc, "something", SQL_NTS, "jsk", SQL_NTS, "hemligt", SQL_NTS);
--
-- Ahora, tenemos que crear el primer usuario autorizado. Cuando se crea
-- el primer usuario, éste tiene que tener el mismo nombre y la misma
-- contraseña que el usuario que ha iniciado la sesión:
--
CLP:> GRANT ENCRYPT ON DATABASE TO "jsk" USING "hemligt" NEW "hemligt";

-- Observe que para GRANT es necesario que el nombre y las contraseñas
-- estén entre comillas dobles. Esto es debido a que son sensibles a las
-- mayúsculas y minúsculas y la sentencia se pasa directamente a DB2 Everyplace.
--
-- Ahora que tenemos un usuario de cifrado autorizado, podemos crear
```

```
-- la tabla cifrada:
--

CLP:> CREATE TABLE english(number INT, word VARCHAR(32)) WITH ENCRYPTION;
CLP:> INSERT INTO english VALUES(1, 'one');
CLP:> INSERT INTO english VALUES(3, 'three');
CLP:> INSERT INTO english VALUES(4, 'four');
CLP:> INSERT INTO english VALUES(5, 'five');
CLP:> INSERT INTO english VALUES(7, 'seven');
CLP:> INSERT INTO english VALUES(99, 'ninety nine');

-- Eche un vistazo a los datos.
CLP:> SELECT * FROM english;

NUMBER      WORD
----------- --------------------------------
          1 one
          3 three
          4 four
          5 five
          7 seven
         99 ninety nine
6 row(s) returned.

-- Seleccione un número aleatorio elevado en sueco:
--
CLP:> SELECT * FROM swedish WHERE nummer > 42;

NUMMER      ORD
----------- --------------------------------

         99 nittionio
1 row(s) returned.

-- Seleccione un número aleatorio elevado en inglés:
--
CLP:> SELECT * FROM english WHERE number > 42;

NUMBER      WORD
----------- --------------------------------
         99 ninety nine
1 row(s) returned.

-- Traduzca 'fyra' al inglés:
--
CLP:> SELECT word FROM swedish, english WHERE number = nummer AND ord = 'fyra';

WORD
--------------------------------
four
1 row(s) returned.

-- Obtenga una tabla de traducción:
--
CLP:> SELECT number, ord, word FROM swedish, english WHERE number = nummer;

NUMBER      ORD                              WORD
----------- -------------------------------- --------------------------------
          1 ett                              one
          3 tre                              three
          4 fyra                             four
          5 fem                              five
          7 sju                              seven
         99 nittionio                        ninety nine
6 row(s) returned.

-- Intente autorizar a otra usuaria para que acceda a los datos cifrados
-- con su propia contraseña:
--
CLP:> GRANT ENCRYPT ON DATABASE TO "xin" USING "notKnown" NEW "notKnown";
Statement failed [sqlstate = 42506].

-- Esta sentencia ha fallado porque el usuario que ha iniciado la sesión
-- se tiene que autovalidar para poder añadir un nuevo usuario; esto se
-- hace proporcionando su contraseña detrás de la cláusula USING.
--
CLP:> GRANT ENCRYPT ON DATABASE TO "xin" USING "hemligt" NEW "notKnown";

-- Volvamos a conectar con el nuevo usuario:
--
CLP:> CONNECT TO samething USER xin USING notknown;
Statement failed [sqlstate = 42505].

-- Esta vez falla porque la contraseña no es igual, por lo que no se
```

```
-- generará la misma clave y se deniega el acceso.
--
CLP:> CONNECT TO samething USER ksin USING notKnown;

-- Esta vez no fallará porque el usuario ksin no existe y, por consiguiente,
-- no intentamos autentificar el usuario.
-- Sin embargo, si utilizamos SQLGetInfo podremos distinguir este caso del
-- caso en que un usuario se ha autentificado satisfactoriamente.
--
CLP:> SELECT * FROM swedish;

NUMMER      ORD
----------- --------------------------------
          1 ett
          3 tre
          4 fyra
          5 fem
          7 sju
         99 nittionio
6 row(s) returned.

-- La selección de datos no cifrados funciona bien, sin embargo los datos
-- no cifrados no se pueden leer/actualizar a no ser que haya un usuario
-- autorizado conectado:
--
CLP:> SELECT * FROM english;
Statement failed [sqlstate = 42501].

-- Conectar como el nuevo usuario, finalmente con nombre de usuario y
-- contraseña correctos.
--
CLP:> CONNECT TO samething USER xin USING notKnown;

-- Verificar que estamos autentificados y podemos acceder a los datos.
--
CLP:> SELECT * FROM english;

NUMBER      WORD
----------- --------------------------------
          1 one
          3 three
          4 four
          5 five
          7 seven
         99 ninety nine
6 row(s) returned.

-- Añadir otro usuario:
--
CLP:> GRANT ENCRYPT ON DATABASE TO "thf" USING "notKnown" NEW "heimlich";

-- Listar los usuarios existentes actualmente:
--
CLP:> SELECT username, grantorname FROM "DB2eSYSUSERS";

USERNAME            GRANTORNAME
------------------- -------------------
jsk                 jsk
xin                 jsk
thf                 xin
3 row(s) returned.

-- Volver a conectar como "jsk":
--
CLP:> CONNECT TO itagain USER jsk USING hemligt;
Statement completed successfully.

-- Intento de cambiar la contraseña por "secret":
--
CLP:> GRANT ENCRYPT ON DATABASE TO "jsk" USING "secret" NEW "secret";
Statement failed [sqlstate = 42506].

-- Ah, hemos fallado porque es necesario que suministremos primero
-- nuestra contraseña antigua, y luego la nueva:
--
CLP:> GRANT ENCRYPT ON DATABASE TO "jsk" USING "hemligt" NEW "secret";

-- Intentarlo con la nueva contraseña:
--
CLP:> CONNECT TO itagain USER jsk USING secret;

-- Asegurarnos de que podemos acceder a datos cifrados:
--
CLP:> SELECT * FROM english;
```

```
NUMBER      WORD
----------- --------------------------------
          1 one
          3 three
          4 four
          5 five
          7 seven
         99 ninety nine
6 row(s) returned.

-- Vamos a eliminar el privilegio de cifrado de "xin":
--
CLP:> REVOKE ENCRYPT ON DATABASE FROM "xin";

-- Listar los usuarios
--
CLP:> SELECT username, grantorname FROM "DB2eSYSUSERS";

USERNAME            GRANTORNAME
------------------- -------------------
jsk                 jsk
thf                 xin
2 row(s) returned.

-- Volver a conectar con el usuario que ahora no existe, sin errores.
--
CLP:> CONNECT TO thedatabase USER xin USING idontknow;

-- Intentar realizar operaciones de cifrado sin autorización:
--
CLP:> SELECT * FROM english;
Statement failed [sqlstate = 42501].

CLP:> DROP TABLE english;
Statement failed [sqlstate = 42501].

CLP:> REVOKE ENCRYPT FROM "jsk";
Statement failed [sqlstate = 42601].

CLP:> GRANT ENCRYPT ON DATABASE TO "xin" USING "idontknow" NEW "idontknow";
Statement failed [sqlstate = 42502].

-- Conectar como "thf":
--
CLP:> CONNECT TO thedatabase USER thf USING heimlich;

-- Comprobar que podemos leer datos cifrados:
--
CLP:> SELECT * FROM english;

NUMBER      WORD
----------- --------------------------------
          1 one
          3 three
          4 four
          5 five
          7 seven
         99 ninety nine
6 row(s) returned.

-- Vamos a eliminar el privilegio del usuario conectado:
--
CLP:> REVOKE ENCRYPT ON DATABASE FROM "thf";

-- Asegurarnos de que ya no puede acceder a los datos:
--
CLP:> SELECT * FROM english;
Statement failed [sqlstate = 42501].

-- Si conectamos con la base de datos como el único usuario que queda, "jsk"
--
CLP:> CONNECT TO thedatabase USER jsk USING secret;

-- Eliminamos el usuario conectado, éste ya no puede acceder a los datos.
-- Realmente, no hay manera de acceder a los datos cifrados de nuevo.
--

CLP:> REVOKE ENCRYPT ON DATABASE FROM "jsk";

-- Asegurarnos de que no queda ningún usuario:
--
CLP:> SELECT username, grantorname FROM "DB2eSYSUSERS";
```

```
USERNAME            GRANTORNAME
------------------- -------------------
0 row(s) returned.

-- Ahora no debemos poder acceder a los datos cifrados.
--
CLP:> SELECT * FROM english;
Statement failed [sqlstate = 42501].

-- Y así concluye la sesión de ejemplo.
```

Tareas afines:

- “Visión general del cifrado local de los datos” en la página 81
- “Cómo otorgar privilegios de cifrado a un usuario” en la página 83
- “Creación de una tabla cifrada” en la página 84
- “Gestión de los privilegios de cifrado” en la página 84

Parte 3. Aplicaciones de ejemplo

Capítulo 13. Aplicaciones C/C++ de ejemplo . . 93

Capítulo 14. Las aplicaciones Java de ejemplo 95
Visión general de las aplicaciones Java de ejemplo 95
Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS 97
Instalación de WCE Tooling para WSDD para destinos de Palm OS 97
Creación de un proyecto de WSDD para DB2eAppl.java para destinos de Palm OS 98
Cómo añadir el controlador de JDBC de DB2 Everyplace y el paquete java.sql a la vía de acceso de creación 99
Importación de DB2eAppl.java a WSDD para Palm OS 99
Ejecución de DB2eAppl.java en un emulador de Palm OS 100
Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS 102
Instalación de WCE Tooling para WSDD para destinos no Palm OS 103
Cómo crear un proyecto WSDD y añadir archivos jar a la vía de acceso de creación para DB2eAppl.java para destinos no Palm OS 103
Importación de DB2eAppl.java a WSDD para destinos no Palm OS 104
Ejecución de las aplicaciones Java de ejemplo . . 105
Ejecución de DB2eAppl.java en Win32 105
Ejecución de DB2eAppl.java en Windows CE 106
Ejecución de DB2eAppl.java en QNX Neutrino o Linux incorporado. 108
Ejecución de DB2eAppl.java en Symbian . . . 108

Capítulo 15. La aplicación de Visual Basic de ejemplo 111
Visión general de la aplicación Visual Basic de ejemplo 111
Compilación y prueba del programa Visual Basic de ejemplo 114

Capítulo 16. Las aplicaciones JSP de ejemplo 117

Capítulo 17. Aplicaciones de sincronización de ejemplo 119
La aplicación Sync Client C/C++ de ejemplo . . . 119
Las aplicaciones de sincronización nativa de Java de ejemplo 121
Las aplicaciones de sincronización MIDP de Java de ejemplo 125
Desarrollo de la aplicación isync4j para MIDP con Sun Wireless Toolkit 129
Desarrollo de la aplicación isync4j para MIDP con ANT y la línea de mandatos de Sun Wireless Toolkit. 131
Compilación y ejecución de la aplicación de sincronización de Java de ejemplo GoISyncConsole. 132

© Copyright IBM Corp. 1999,2003

Capítulo 13. Aplicaciones C/C++ de ejemplo

Se proporciona, como mínimo, una aplicación C/C++ de ejemplo para cada sistema operativo. Consulte el directorio de cliente apropiado para ver todas las aplicaciones de ejemplos con código fuente.

Tareas afines:

- "Cómo desarrollar aplicaciones C/C++ de DB2 Everyplace" en la página 9

Consulta relacionada:

- "Herramientas de desarrollo de C/C++ soportadas" en la página 10
- "Sistemas operativos soportados de C/C++" en la página 11

© Copyright IBM Corp. 1999,2003

Capítulo 14. Las aplicaciones Java de ejemplo

Este capítulo proporciona información sobre las aplicaciones Java de ejemplo. Los temas que se tratan son:

- "Visión general de las aplicaciones Java de ejemplo"
- "Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS" en la página 97
- "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102

Visión general de las aplicaciones Java de ejemplo

Este tema describe las aplicaciones de ejemplo `DB2eAppl.java` y `DB2eJavaCLP.java`.

Ejemplo 1: DB2eAppl.java:

`DB2eAppl.java` demuestra cómo codificar una aplicación JDBC para DB2 Everyplace.

Los pasos principales de la aplicación `DB2eAppl.java` son:

Paso 1:
Importar el paquete `java.sql`.

Paso 2:
Cargar el controlador JDBC de DB2 Everyplace `com.ibm.db2e.jdbc.DB2eDriver`.

Paso 3:
Conectar con la base de datos en el directorio actual, el directorio en el que se ejecutará el programa `DB2eAppl.java`.

Paso 4:
Crear un objeto de Sentencia.

Paso 5:
Configurar la base de datos de ejemplo muy simple, que consta de una tabla EMPLOYEE que contiene dos registros. Esto se hace utilizando el método `executeUpdate(String sql)` de la interfaz `java.sql.Statement`.

Paso 6:
Seleccionar todos los registros de la tabla EMPLOYEE y recuperar las filas utilizando el método `next()` de la interfaz `java.sql.ResultSet`.

Paso 7:
Eliminar la tabla EMPLOYEE de la base de datos y liberar los recursos de base de datos y de JDBC.

El código fuente de `DB2eAppl.java` que aparece a continuación contiene comentarios que muestran dónde se utilizan los pasos explicados más arriba.

```
import java.sql.*; //Paso 1

public class DB2eAppl
{
    public static void main(String[] args) {

        String driver = "com.ibm.db2e.jdbc.DB2eDriver";
        String url    = "jdbc:db2e:mysample";
```

© Copyright IBM Corp. 1999,2003

```
        try {
            Class.forName(driver);//Paso 2
            Connection con = DriverManager.getConnection(url);//Paso 3
            Statement st = con.createStatement();//Paso 4

            //Crear tabla: employee  //Paso 5
            st.executeUpdate("CREATE TABLE employee (EMPNO CHAR(6), FIRSTNAME VARCHAR(12))");
            System.out.println("*** Created table: employee");

    //Añadir registros a employee
            st.executeUpdate("INSERT INTO employee VALUES ('112233','John')");
            st.executeUpdate("INSERT INTO employee VALUES ('445566','Mary')");
            System.out.println("*** Inserted two records");

    //Consultar y mostrar resultados //Paso 6
            ResultSet rs = st.executeQuery("SELECT * FROM employee");
            System.out.println("*** Query results:");
            while (rs.next()) {
                System.out.print("EMPNO=" + rs.getString(1) + ", ");
                System.out.println("FIRSTNAME=" + rs.getString(2));
            }

            //Suprimir tabla: employee  //Paso 7
            st.executeUpdate("Drop table employee");
            System.out.println("*** Deleted table: employee");

            rs.close();
            st.close();
            con.close();

        } catch (SQLException sqlEx) {
            while(sqlEx !=null)
            {
                System.out.println("[SQLException] " +
                                   "SQLState: " + sqlEx.getSQLState() +
                    ",Message:"+sqlEx.getMessage()+
                    ",Vendor:"+sqlEx.getErrorCode());
                sqlEx =sqlEx.getNextException();
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

Ejemplo 2: DB2eJavaCLP.java:

`DB2eJavaCLP.java` es un procesador de línea de mandatos Java para DB2 Everyplace.

Restricción: En Palm OS, la aplicación de ejemplo `DB2eJavaCLP.java` no está soportada.

Tareas afines:

- "Desarrollo de aplicaciones Java de DB2 Everyplace" en la página 17
- "Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS" en la página 97
- "Creación de un proyecto de WSDD para DB2eAppl.java para destinos de Palm OS" en la página 98
- "Cómo crear un proyecto WSDD y añadir archivos jar a la vía de acceso de creación para DB2eAppl.java para destinos no Palm OS" en la página 103
- "Ejecución de DB2eAppl.java en Win32" en la página 105
- "Ejecución de DB2eAppl.java en Windows CE" en la página 106
- "Ejecución de DB2eAppl.java en un emulador de Palm OS" en la página 100
- "Ejecución de DB2eAppl.java en QNX Neutrino o Linux incorporado" en la página 108
- "Ejecución de DB2eAppl.java en Symbian" en la página 108

- "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102

Conceptos afines:

- "Las aplicaciones de sincronización nativa de Java de ejemplo" en la página 121

Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS

El conjunto de temas siguiente describe el modo de compilar y ejecutar el código Java `DB2eAppl.java` de ejemplo en destinos de Palm OS.

Recomendamos la utilización de WebSphere Studio Device Developer (WSDD) como entorno de desarrollo. WSDD utiliza la MV J9, que es posible que no soporte el tipo de procesador de su dispositivo. Si utiliza otro entorno de desarrollo y una JVM, asegúrese de que la JVM soporte la JNI, puesto que el controlador JDBC de DB2 Everyplace usa la JNI. Otras JVM compatibles incluyen Sun PersonalJava, Insignia Jeode y NSIcom CrEme. En la actualidad, si el destino es Palm OS, *deberá* utilizar la J9 VM que acompaña a WSDD. La versión de evaluación de WSDD se puede bajar desde la dirección http://www.ibm.com/software/pervasive/products/wsdd/.

Requisitos previos:

1. Asegúrese de que tiene instalado el software siguiente:
   - WSDD 5.5, que incluye la Máquina virtual Java (Java Virtual Machine) J9
2. Prepare el destino y el entorno de desarrollo según la documentación de WSDD. Verifique la instalación de WSDD creando y ejecutando aplicaciones WSDD de ejemplo.
3. Instale DB2 Everyplace en el dispositivo de destino. Consulte el manual *DB2 Everyplace Guía del usuario y de instalación* para obtener instrucciones más detalladas.

Procedimiento:

Para compilar y ejecutar el código Java de ejemplo en destinos de Palm OS:

1. Instale WCE Tooling for WSDD.
2. Cree un proyecto WSDD para `DB2eAppl.java`.
3. Añada el controlador JDBC de DB2 Everyplace y el paquete java.sql a la vía de acceso de creación.
4. Importe `DB2eAppl.java` a WSDD.
5. Ejecute `DB2eAppl.java` en un emulador de Palm OS.

Tareas afines:

- "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102

Instalación de WCE Tooling para WSDD para destinos de Palm OS

Esta tarea forma parte de la tarea más importante de Compilación y ejecución de aplicaciones Java de ejemplo en destinos Palm OS. Después de completar estos pasos, vuelva al apartado "Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS".

Procedimiento:

1. En WSDD, pulse **Ayuda —> Actualizaciones de software —> Gestor de actualización** para abrir la perspectiva de Instalación/Actualización
2. En la Vista de actualizaciones de características de La Perspectiva de Instalación/Actualización amplíe lo siguiente: **Sitios a visitar—> Entorno personalizado de WebSphere —> Entorno personalizado de WebSphere**.
3. Seleccione WCE Tooling para WSDD 5.5.0.
4. En la Vista previa, pulse **Instalar**.
5. Siga las instrucciones de instalación para instalar el WCE Tooling para característica de WSDD.
6. Si selecciona la utilización de la biblioteca de clase jclXtr, deberá instalar asimismo la Biblioteca de clase jclXtr de WCE siguiendo pasos similares.

Vuelva a "Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS" en la página 97.

Creación de un proyecto de WSDD para DB2eAppl.java para destinos de Palm OS

Existen dos versiones de Controlador JDBC de DB2 Everyplace para Palm OS disponibles. Una versión es compatible con la Configuración de CLDC de J2ME. La otra versión es compatible con la Configuración personalizada de JCL Extreme Palm que proporciona WSDD. Siga los pasos correspondientes para crear el tipo de proyecto que necesite.

Esta tarea forma parte de la tarea más importante de Compilación y ejecución de aplicaciones Java de ejemplo en destinos Palm OS. Después de completar estos pasos, vuelva al apartado "Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS" en la página 97.

Procedimiento:

Para crear un proyecto de Palm OS de WSDD utilizando la configuración de jclCldc::

1. En WSDD, pulse **Ventana —>Abrir perspectiva —>Java** para conmutar a la perspectiva de Java.
2. Cree el proyecto DB2 Everyplace Sample for Palm OS CLDC:
   a. Pulse **Archivo —>Nuevo —>Otro**.
   b. En la página Seleccionar de la ventana Nuevo proyecto, seleccione J2ME para J9 en el panel izquierdo, seleccione Crear proyecto J2ME en el panel derecho y después pulse **Siguiente**.
   c. En la página Proyecto J2ME de la ventana Proyecto nuevo, escriba `DB2 Everyplace Sample for Palm OS CLDC` en el campo de nombre de Proyecto y después pulse **Siguiente**.
   d. En la página de selección de Biblioteca de la ventana Proyecto nuevo, seleccione jclCldc de WME (jclCldc) y después pulse **Finalizar**.

Para crear un proyecto de Palm OS de WSDD utilizando la configuración de jclXtr::

1. En WSDD, pulse **Ventana —>Abrir perspectiva —>Java** para conmutar a la perspectiva de Java.
2. Cree el proyecto DB2 Everyplace Sample for Palm OS XTR:

a. Pulse **Archivo —>Nuevo —>Otro**.
b. En la página Seleccionar de la ventana Nuevo proyecto, seleccione WCE para J9 en el panel izquierdo, seleccione Crear proyecto WCE en el panel derecho y después pulse **Siguiente**.
c. En la página Proyecto personalizado de la ventana Proyecto nuevo, escriba `DB2 Everyplace Sample for Palm OS XTR` en el campo de nombre de Proyecto y después pulse **Siguiente**.
d. En la página de selección de Biblioteca de la ventana Proyecto nuevo, seleccione jclXtr de WCE (jclXtr) y después pulse **Finalizar**.

Vuelva a "Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS" en la página 97.

Cómo añadir el controlador de JDBC de DB2 Everyplace y el paquete java.sql a la vía de acceso de creación

Los pasos siguientes se aplican al proyecto DB2 Everyplace Sample for Palm OS CLDC. El proyecto DB2 Everyplace Sample for Palm OS XTR implica pasos similares:

Esta tarea forma parte de la tarea más importante de Compilación y ejecución de aplicaciones Java de ejemplo en destinos Palm OS. Después de completar estos pasos, vuelva al apartado "Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS" en la página 97.

Procedimiento:

Para añadir el controlador JDBC de DB2 Everyplace (y el paquete `java.sql` a la vía de acceso de creación:

1. Pulse el botón derecho del ratón en el proyecto de DB2 Everyplace Sample for Palm OS CLDC de la vista del Explorador de paquetes de la Perspectiva de Java y después pulse **Propiedades** en el menú emergente.
2. En la ventana de propiedades que se abra, pulse Vía de acceso de creación de Java en el panel de la izquierda y después pulse la pestaña Bibliotecas en el panel de la derecha.
3. Pulse **Añadir JAR externos**. En la ventana Selección de JAR, navegue hacia `<DB2Everyplace>\Clients\PalmOS\database\JDBC\cldc\db2ejdbc.jar` y después pulse **Abrir**.
4. Repita el paso anterior para añadir `database_enabler_cldc.jar` y `DB2eJDBC_Cldc_maps.jar` a la vía de acceso de creación.
5. De vuelta en la ventana Propiedades para DB2 Everyplace Sample for Palm OS CLDC, pulse **Bien**.

Vuelva a "Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS" en la página 97.

Importación de DB2eAppl.java a WSDD para Palm OS

Los pasos siguientes se aplican al proyecto DB2 Everyplace Sample for Palm OS CLDC. El proyecto DB2 Everyplace Sample for Palm OS XTR implica pasos similares:

Esta tarea forma parte de la tarea más importante de Compilación y ejecución de aplicaciones Java de ejemplo en destinos Palm OS. Después de completar estos pasos, vuelva al apartado "Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS" en la página 97.

Procedimiento:

Para importar `DB2eAppl.java` a WSDD:

1. En la vista del Explorador de paquetes de la Perspectiva Java, pulse el botón derecho del ratón en la carpeta `src` del proyecto DB2 Everyplace Sample for Palm OS CLDC y después pulse **Importar** en el menú emergente.
2. En la página Seleccionar de la ventana Importar, seleccione Sistema de archivos como fuente de importación y después pulse **Siguiente**.
3. En la página Sistema de archivos de la ventana Importar, pulse **Examinar**.
4. Navegue a la carpeta `<DB2Everyplace>\Clients\PalmOS\database\JDBC\cldc\sample`, después pulse **Bien**.
5. Seleccione el recuadro de selección **DB2eAppl.java** del panel derecho y después pulse **Finalizar**.

Vuelva a "Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS" en la página 97.

Ejecución de DB2eAppl.java en un emulador de Palm OS

Los pasos siguientes se aplican al proyecto DB2 Everyplace Sample for Palm OS CLDC. El proyecto DB2 Everyplace Sample for Palm OS XTR implica pasos similares:

Esta tarea forma parte de la tarea más importante de Compilación y ejecución de aplicaciones Java de ejemplo en destinos Palm OS. Después de completar estos pasos, vuelva al apartado "Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS" en la página 97.

Requisitos previos:

Si todavía no ha configurado el sistema para utilizar el controlador JDBC de DB2 Everyplace, instale los siguientes archivos para el controlador JDBC de su dispositivo:

<DB2 Everyplace>`\Clients\PalmOS\database\JDBC\cldc\DB2eJDBC.prc`
<DB2 Everyplace>`\Clients\PalmOS\database\JDBC\cldc\DB2eJDBC_Cldc.prc`

Si está trabajando con el proyecto DB2 Everyplace Sample for Palm OS XTR, instale los siguientes archivos para el controlador JDBC de su dispositivo:

<DB2 Everyplace>`\Clients\PalmOS\database\JDBC\xtr\DB2eJDBC.prc`
<DB2 Everyplace>`\Clients\PalmOS\database\JDBC\xtr\DB2eJDBC_Xtr.prc`

Procedimiento:

Para ejecutar `DB2eAppl.java` en un emulador Palm OS:

1. Configure el emulador Palm OS:
   a. Pulse **Dispositivos —> Configurar**.

b. En la ventana Configuraciones de dispositivo, seleccione Palm Emulator en el panel izquierdo y después pulse **Nueva**.

c. En la configuración que aparece a la derecha, entre la siguiente información:

- En el campo **Nombre de dispositivo**, escriba `DB2 Everyplace Palm Emulator`.
- En el campo **Ejecutable de emulador de Palm**, examine el <*PalmEmulator*>\Emulator.exe, donde <*PalmEmulator*> es el directorio en el que ha instalado el Palm Emulator.
- En el campo **Argumentos de ejecución de emulador**, escriba -psf <*archivo*>.psf, donde <*archivo*>.psf es un archivo .psf que tenga instalado DB2 Everyplace y el J9 VM.

d. Pulse Aplicar y después pulse Bien.

2. Cree `DB2eAppl.java`.

a. En la vista del Explorador de paquetes de la Perspectiva Java, pulse dos veces en el archivo `wsddbuild.xml` para el proyecto DB2 Everyplace Sample for Palm OS CLDC.

b. En el editor para `wsddbuild.xml`, pulse **Añadir creación**.

c. Seleccione J9 para Palm 68k en la Lista de **plataformas**, conserve los valores por omisión en los campos **Clase principal** y **Nombrecreación** y después pulse **Siguiente**.

d. En la página de valores de PalmOS, escriba `DB2e` en el campo **ID de creador** y escriba `DB2eAppl` en el campo **Nombre de aplic.** y después pulse **Siguiente**.

e. En la página Opciones de Jxelink, conserve los valores por omisión y pulse **Finalizar**.

3. Modifique el archivo `DB2eAppl.jxeLinkOptions`:

a. En la vista del Explorador de paquetes de la Perspectiva Java, amplíe la carpeta `palm68k` para el proyecto DB2 Everyplace Sample for Palm OS CLDC. Pulse dos veces en DB2eAppl.jxeLinkOptions para abrir el editor para DB2eAppl.jxeLinkOptions.

b. En el editor, pulse la pestaña Entrada. Pulse **Nueva** para la sección Leer clases en archivos de correlación (requisito previo). En la ventana Añadir requisito previo que aparecerá, entre `DB2eJDBC_Cldc` para el requisito previo y después pulse **Bien**. **Nota:** Si está trabajando con el proyecto DB2 Everyplace Sample for Palm OS XTR, entre `DB2eJDBC_Xtr` como requisito previo y sáltese los dos pasos siguientes.

c. En el editor, pulse la pestaña Jxe. En Información de plataforma de Jxe, pulse **Nueva** para Utilizar opciones de VM al ejecutar la sección de jxe.

d. En la ventana Añadir opción de VM que aparecerá, entre `-jcl:cldc:loadlibrary=db2ejdbc` para la opción de VM y después pulse **Bien**.

e. Escriba Control+S para guardar los cambios.

f. En el editor para `wsddbuild.xml`, seleccione jxe2prc palm68k/DB2eAppl y después pulse **Realizar creación**.

4. Ejecute `DB2eAppl.java`.

a. Pulse **Ejecutar —> Ejecutar** en el menú principal. Se abrirá la ventana Ejecutar configuraciones.

b. En la ventana Ejecutar configuraciones, seleccione Aplicación Java de dispositivo en el panel izquierdo y después pulse **Nueva**.

c. En la configuración que aparece en el panel derecho, escriba `DB2eAppl Palm CLDC` en el campo **Nombre**.
d. En el panel de Aplicación Java, entre la información siguiente:
   1) En el campo de Proyecto, examine DB2 Everyplace Sample for Palm OS CLDC.
   2) Pulse **Buscar** en el campo de Aplicación de Java.
   3) En la ventana Seleccionar destino, seleccione `DB2eAppl.prc (Target "jxe2prc palm68k/DB2eAppl" in wsddbuild.xml)` y después pulse **Finalizar**.
   4) Seleccione DB2 Everyplace Palm Emulator en la lista de **Dispositivos o JRE**.
   5) De nuevo en la ventana Ejecutar configuraciones, pulse **Aplicar** y después pulse **Ejecutar**. Un Emulador de Palm Emulator debe iniciar y ejecutar DB2eAppl. Debe ver la salida para la aplicación de ejemplo en la pantalla Emulador de Palm o en el archivo `j9stdout.txt` del directorio en el que está el archivo .psf. Si no modificó la preferencia ″Visualizar stdout en pantalla″ de J9 Java VM, la salida estará en el archivo `j9stdout.txt`. Compruebe asimismo si hay errores en `j9stderr.txt`.

Vuelva a “Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS” en la página 97.

Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS

El conjunto de temas siguiente describe el modo de compilar y ejecutar el código Java de ejemplo utilizando el WebSphere Studio Device Developer (WSDD) 5.5 y la Máquina Virtual Java (Java Virtual Machine) J9.

Recomendamos la utilización de WSDD como entorno de desarrollo. WSDD utiliza la MV J9, que es posible que no soporte el tipo de procesador de su dispositivo. Si utiliza otro entorno de desarrollo y una JVM, asegúrese de que la JVM soporte la JNI, puesto que el controlador JDBC de DB2 Everyplace usa la JNI. Otras JVM compatibles incluyen Sun PersonalJava, Insignia Jeode y NSIcom CrEme. La versión de evaluación de WSDD se puede bajar desde la dirección http://www.ibm.com/software/pervasive/products/wsdd/.

Requisitos previos:

1. Asegúrese de que tiene instalado el software siguiente:
   - WSDD 5.5, que incluye la Máquina Virtual Java (Java Virtual Machine) J9, o algún otro JVM compatible.
2. Prepare el destino y el entorno de desarrollo según la documentación de WSDD. Verifique la instalación de WSDD creando y ejecutando aplicaciones WSDD de ejemplo.
3. Instale DB2 Everyplace en el dispositivo de destino. Consulte el manual *DB2 Everyplace Guía del usuario y de instalación* para obtener instrucciones más detalladas.

Procedimiento:

Para compilar y ejecutar el código Java de ejemplo en destinos diferentes a Palm OS:

1. Instale WCE Tooling for WSDD.
2. Cree un proyecto WSDD y añada archivos jar a la vía de acceso de creación para `DB2eAppl.java`.
3. Importe `DB2eAppl.java` a WSDD.
4. Ejecute `DB2eAppl.java`. Los pasos varían en función de su sistema operativo.
   - "Ejecución de DB2eAppl.java en Win32" en la página 105
   - "Ejecución de DB2eAppl.java en Windows CE" en la página 106
   - "Ejecución de DB2eAppl.java en QNX Neutrino o Linux incorporado" en la página 108
   - "Ejecución de DB2eAppl.java en Symbian" en la página 108

Tareas afines:

- "Compilación y ejecución de aplicaciones Java de ejemplo en destinos de Palm OS" en la página 97

Instalación de WCE Tooling para WSDD para destinos no Palm OS

Esta tarea forma parte de la tarea más importante de Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS. Después de completar estos pasos, vuelva al apartado "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102.

Procedimiento:

1. En WSDD, pulse **Ayuda —> Actualizaciones de software —> Gestor de actualización** para abrir la perspectiva de Instalación/Actualización
2. En la Vista de actualizaciones de características de La Perspectiva de Instalación/Actualización, amplíe lo siguiente: **Sitios a visitar—> Entorno personalizado de WebSphere —> Entorno personalizado de WebSphere**.
3. Seleccione WCE Tooling para WSDD 5.5.0.
4. En la Vista previa, pulse **Instalar**.
5. Siga las instrucciones de instalación para instalar el WCE Tooling para característica de WSDD.
6. Instale la WCE Database Enabler Library y la WCE jclMax Class Library siguiendo pasos similares.

Vuelva a "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102.

Cómo crear un proyecto WSDD y añadir archivos jar a la vía de acceso de creación para DB2eAppl.java para destinos no Palm OS

Esta tarea forma parte de la tarea más importante de Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS. Después de completar estos pasos, vuelva al apartado "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102.

Procedimiento:

1. En WSDD, pulse **Ventana —>Abrir perspectiva —>Java** para conmutar a la perspectiva de Java
2. Cree el proyecto DB2 Everyplace Sample:
   a. Pulse **Archivo —>Nuevo —>Otro**.

b. En la página Seleccionar de la ventana Nuevo proyecto, seleccione WCE para J9 en el panel izquierdo, seleccione Crear proyecto WCE en el panel derecho y después pulse **Siguiente**.
c. En la página Proyecto personalizado de la ventana Proyecto nuevo, escriba `DB2 Everyplace Sample` como nombre de proyecto y después pulse **Siguiente**.
d. En la página de selección de Biblioteca de la ventana Proyecto nuevo, seleccione jclMax de WCE (jclMax) y después pulse **Siguiente**.
e. En la página Valores de Java de la ventana Proyecto nuevo, pulse la pestaña **Bibliotecas**.
f. Añada `db2ejdbc.jar` a la vía de acceso de creación:
   1) Pulse **Añadir JAR externos**.
   2) En la ventana Selección de JAR, navegue hacia ***<DB2Everyplace>***`\Clients\Win32\database\jdbc\db2ejdbc.jar` y después pulse **Abrir**.
g. De vuelta en la página Valores de Java, añada `database_enabler.jar` a la vía de acceso de creación.
   1) Pulse **Añadir JAR externos**.
   2) En la ventana Selección de JAR, navegue hacia *<WSDD>*`\wsdd5.0\ive\lib\jclMax\database_enabler.jar` y después pulse **Abrir**.
h. De vuelta en la página Valores de Java de la ventana Proyecto nuevo, pulse **Finalizar**.

Vuelva a "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102

Importación de DB2eAppl.java a WSDD para destinos no Palm OS

Esta tarea forma parte de la tarea más importante de Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS. Después de completar estos pasos, vuelva al apartado "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102.

Procedimiento:

Para importar `DB2eAppl.java` a WSDD para destinos no Palm OS:

1. En la vista del Explorador de paquetes de la Perspectiva Java, seleccione la carpeta `src` y después pulse **Archivo —>Importar**.
2. En la página **Seleccionar** de la ventana Importar, seleccione Sistema de archivos como fuente de importación y después pulse **Siguiente**.
3. En la página **Sistema de archivos** de la ventana Importar, pulse **Examinar** para el campo **Directorio**, navegue hacia la carpeta *<DB2Everyplace>*`\Clients\Win32\database\jdbc` y después pulse **Bien**.
4. De vuelta en la página **Sistema de archivos** de la ventana Importar, seleccione el recuadro de selección **DB2eAppl.java** del panel derecho y después pulse **Finalizar**.

Vuelva a "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102

Ejecución de las aplicaciones Java de ejemplo

Este capítulo describe el modo de ejecutar las aplicaciones Java de ejemplo. Los temas que se tratan son:

- "Ejecución de DB2eAppl.java en Win32"
- "Ejecución de DB2eAppl.java en Windows CE" en la página 106
- "Ejecución de DB2eAppl.java en QNX Neutrino o Linux incorporado" en la página 108
- "Ejecución de DB2eAppl.java en Symbian" en la página 108

Ejecución de DB2eAppl.java en Win32

Esta tarea forma parte de la tarea más importante de Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS. Después de completar estos pasos, vuelva al apartado "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102.

Requisitos previos:

Si todavía no ha configurado el sistema para utilizar el controlador JDBC de DB2 Everyplace:

1. Mediante el mandato set, incluya el directorio siguiente en la variable del sistema PATH: `<DB2Everyplace>\Clients\Win32\database\x86`
2. Mediante el mandato set, incluya el archivo siguiente en la variable del sistema CLASSPATH: `<DB2Everyplace>\Clients\Win32\database\jdbc\db2ejdbc.jar`

Nota: Si WSDD está abierto, necesitará reiniciarlo para que estos cambios se reflejen en WSDD.

Procedimiento:

Para ejecutar `DB2eAppl.java` en una estación de trabajo Windows:

1. Cree `DB2eAppl.java`:
   a. En la vista del Explorador de paquetes de la Perspectiva Java, pulse dos veces en el archivo `wsddbuild.xml` para el proyecto DB2 Everyplace Sample.
   b. En el editor para `wsddbuild.xml`, pulse **Añadir creación**.
   c. En la ventana Crear destino de creación ant nueva, pulse **Examinar** para el campo **Clase principal**.
   d. En la ventana que se abre, seleccione DB2eAppl - (paquete por omisión) - DB2 Everyplace Sample/src y después pulse **Finalizar**.
   e. De nuevo en la ventana Crear destino de creación ant nueva, seleccione J9 para Windows X86 en la lista de plataformas, conserve el valor por omisión en el campo Nombrecreación y después pulse **Siguiente**.
   f. En la página Opciones de Jxelink, conserve los valores por omisión y pulse **Finalizar**.
   g. De nuevo en el editor para `wsddbuild.xml`, seleccione smartlink winx86/DB2eAppl y después pulse **Realizar creación**.
2. Ejecute `DB2eAppl.java`.
   a. Pulse **Ejecutar —> Ejecutar**. Se abrirá la ventana Ejecutar configuraciones.
   b. En la ventana Ejecutar configuraciones, seleccione Aplicación Java en el panel izquierdo y después pulse **Nueva**.

c. En la configuración que aparece en el panel derecho, escriba `DB2eAppl Win32` en el campo **Nombre**.
d. En la página Principal, siga estos pasos:
   - Pulse **Examinar** para el campo **Proyecto**. En la ventana Selección de proyecto, seleccione DB2 Everyplace Sample y después pulse **Bien**.
   - Pulse **Buscar** para el campo **Clase principal**. En la ventana Elegir tipo principal, seleccione DB2eAppl y después pulse **Bien**.
e. De nuevo en la ventana Ejecutar configuraciones, pulse **Aplicar** y después pulse **Ejecutar**. Debe ver la salida de la aplicación de ejemplo en la Consola de WSDD.

Vuelva a "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102.

Ejecución de DB2eAppl.java en Windows CE

Esta tarea forma parte de la tarea más importante de Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS. Después de completar estos pasos, vuelva al apartado "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102.

Requisitos previos:

Si todavía no ha configurado el sistema para utilizar el controlador JDBC de DB2 Everyplace, siga los pasos siguientes:

1. Copie los archivos siguientes en el directorio \Windows del dispositivo:`<DB2Everyplace>\Clients \WinCE \database \`*`proc`* `\`*`ver`* `\db2ejdbc.dll<DB2Everyplace>\Clients \WinCE \database \jdbc \db2ejdbc.jar`donde *proc* es el tipo de procesador y *ver* es el número de versión del sistema operativo Windows CE en el dispositivo.
2. Utilizando el Editor de registro remoto de Windows CE, modifique el registro del dispositivo para incluir los archivos siguientes en la vía de clases del dispositivo:

   ```
   \Windows\db2ejdbc.jar
   \wsdd\lib\jclMax\database_enabler.jar
   ```

   suponiendo que haya instalado J9 debajo del directorio root del dispositivo.

   Alternativamente, puede actualizar el atajo DB2eAppl generado por WSDD para incluir los archivos anteriores en la vía de clases:

```
256#"\wsdd\bin\j9.exe" "-Xbootclasspath:\Windows\db2ejdbc.jar;
\wsdd\lib\jclMax\database_enabler.jar;\wsdd\lib\jclMax\classes.zip;
\wsdd\lib\jclMax\locale.zip;\wsdd\lib\charconv.zip" "-jcl:max"
"-jxe:\Temp\DB2eAppl.jxe"
```

Procedimiento:

Para ejecutar `DB2eAppl.java` en un dispositivo Windows CE:

1. Configure el dispositivo Windows CE.
   a. Pulse **Dispositivos —> Configurar**.
   b. En la ventana Configuraciones de dispositivo, seleccione Dispositivo de bolsillo PocketPC en el panel izquierdo y después pulse **Nuevo**.
   c. En la configuración que aparece a la derecha, complete los pasos siguientes:

1) En el campo **Nombre de dispositivo**, escriba `Dispositivo de bolsillo PocketPC de DB2 Everyplace`.
2) Pulse **Examinar** para el campo **Ubicación de tiempo de ejecución de J9**. En la ventana Examinar carpeta en el dispositivo, seleccione wsdd (suponiendo que haya instalado J9 en el directorio root del dispositivo) y después pulse **Bien**.
3) Pulse **Examinar** para el campo **Instalación de la ubicación de aplicación**. En la ventana Examinar carpeta en el dispositivo, seleccione Temp y después pulse **Bien**.
4) Pulse **Examinar** para el campo **Ubicación de instalación de atajo**. En la ventana Examinar carpeta en el dispositivo, seleccione Temp y después pulse **Bien**.

d. De nuevo en la ventana Configuraciones de dispositivo, pulse **Aplicar** y después pulse **Bien**.

2. Cree `DB2eAppl.java`.
   a. En el panel del Explorador de paquetes de la Perspectiva Java, pulse dos veces en el archivo `wsddbuild.xml` para el proyecto DB2 Everyplace Sample.
   b. En el editor para `wsddbuild.xml`, pulse **Añadir creación**.
   c. En la ventana Crear destino de creación ant nueva, pulse **Examinar** para el campo **Clase principal**.
   d. En la ventana que se abre, seleccione DB2eAppl - (paquete por omisión) - DB2 Everyplace Sample/src y después pulse **Finalizar**.
   e. De nuevo en la ventana Crear destino de creación ant nueva, seleccione J9 para PocketPC ARM en la Lista de **plataformas**, conserve el valor por omisión en el campo **Nombrecreación** y después pulse **Siguiente**.
   f. En la página **Opciones de Jxelink**, conserve los valores por omisión y pulse **Finalizar**.
   g. En el editor para `wsddbuild.xml`, seleccione smartlink ppcarm/DB2eAppl y después pulse **Realizar creación**.
3. Ejecute `DB2eAppl.java`:
   a. Pulse **Ejecutar —> Ejecutar**. Se abrirá la ventana Ejecutar configuraciones.
   b. En la ventana Ejecutar configuraciones, seleccione Aplicación Java de dispositivo en el panel izquierdo y después pulse **Nueva**.
   c. En la configuración que aparece en el panel derecho, escriba `DB2eAppl WinCE` en el campo **Nombre**.
   d. En la página Principal, siga estos pasos:
      1) Pulse **Examinar** para el campo **Proyecto**. En la ventana Selección de proyecto, seleccione DB2 Everyplace Sample y después pulse **Bien**.
      2) Pulse **Buscar** para el campo de Aplicación de Java. En la ventana Seleccionar destino, seleccione `DB2eAppl.jxe (Target "smartlink ppcarm/DB2eAppl" in wsddbuild.xml)` y después pulse **Finalizar**.
      3) Seleccione Dispositivo de bolsillo PocketPC de DB2 Everyplace en la lista de **Dispositivos o JRE**.
   e. De nuevo en la ventana Ejecutar configuraciones, pulse **Aplicar** y después pulse **Ejecutar**. Debe ver la salida de la aplicación de ejemplo en la Consola de J9 del dispositivo.

Vuelva a “Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS” en la página 102.

Ejecución de DB2eAppl.java en QNX Neutrino o Linux incorporado

Esta tarea forma parte de la tarea más importante de Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS. Después de completar estos pasos, vuelva al apartado "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102.

Requisitos previos:

Si todavía no ha configurado el sistema para utilizar el controlador JDBC de DB2 Everyplace:

1. Utilizando el mandato export, incluya en la variable del sistema LD_LIBRARY_PATH el directorio (o los directorios) que contiene(n) las bibliotecas libdb2e.so y libdb2ejdbc.so nativas apropiadas.

Procedimiento:

Para ejecutar `DB2eAppl.java` en un dispositivo QNX Neutrino o Linux incorporado:

1. Cree `DB2eAppl.java`.
   a. En el panel del Explorador de paquetes de la Perspectiva Java, pulse dos veces en el archivo `wsddbuild.xml` para el proyecto DB2 Everyplace Sample.
   b. En el editor para `wsddbuild.xml`, pulse **Añadir creación**.
   c. En la ventana Crear destino de creación ant nueva, pulse **Examinar** para examinar la clase principal. En la ventana Seleccionar destino que se abre, seleccione DB2eAppl - (paquete por omisión) - DB2 Everyplace Sample/src y después pulse **Finalizar**.
   d. De nuevo en la página Instalar creación de la ventana Crear destino de creación ant nueva, seleccione la plataforma apropiada en la lista de **Plataformas**, conserve el valor por omisión en el campo Nombrecreación y después pulse **Siguiente**.
   e. En la página Opciones de Jxelink, conserve los valores por omisión y pulse **Finalizar**.
   f. De nuevo en el editor para wsddbuild.xml, seleccione la creación apropiada y pulse **realizar creación**.
2. Ejecute `DB2eAppl.java`.
   a. Copie el archivo DB2eAppl.jxe apropiado en el dispositivo desde
      `<WSDD>\workspace\DB2 Everyplace Sample\<target>`

      donde <target> representa el dispositivo de destino y el tipo de procesador.
   b. Inicie la aplicación utilizando el mandato siguiente:

```
j9 -Xbootclasspath:/wsdd/lib/jclMax/classes.zip:/wsdd/lib/jclMax/database_enabler.jar -cp:
  /DB2e/db2ejdbc.jar:. -jxe:DB2eAppl.jxe
```

Vuelva a "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102.

Ejecución de DB2eAppl.java en Symbian

Esta tarea forma parte de la tarea más importante de Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS. Después de completar estos pasos, vuelva al apartado "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102.

Procedimiento:

Algunos dispositivos Symbian se suministran con una JVM. Si desea ejecutar una aplicación Java basada en el texto (por ejemplo, los programas Java de muestra), necesitará instalar Redirect (que se suministra como `Redirect.sis` en Symbian Java SDK) e iniciar la aplicación Redirect antes de iniciar la aplicación basada en el texto. Redirect capturará la salida de texto.

Vuelva a "Compilación y ejecución de aplicaciones Java de ejemplo en destinos no Palm OS" en la página 102.

Capítulo 15. La aplicación de Visual Basic de ejemplo

Este capítulo proporciona información sobre la aplicación Visual Basic de ejemplo. Los temas que se tratan son:

- "Visión general de la aplicación Visual Basic de ejemplo"
- "Compilación y prueba del programa Visual Basic de ejemplo" en la página 114

Visión general de la aplicación Visual Basic de ejemplo

La aplicación Visual Basic de ejemplo le muestra cómo acceder a los datos de DB2 Everyplace utilizando Visual Basic. Puede desarrollar aplicaciones que tengan iguales la lógica de aplicación y la interfaz de usuario en sistemas operativos Pocket PC (WinCE) y Win32. Junto con DB2 Everyplace se proporcionan dos aplicaciones Visual Basic de ejemplo. Uno de ellos es para el sistema operativo Pocket PC (WinCE) y el otro es para los sistemas operativos Win32. Ambas aplicaciones de ejemplo tienen el mismo código e interfaz de usuario. El archivo `db2evb.bas`, que contiene la lógica de la aplicación, es común a ambos sistemas operativos. Consulte 112 para obtener más información.

Archivos incluidos en la aplicación de ejemplo:

El directorio de proyectos Visual Basic, donde reside la aplicación de ejemplo, está situado en el directorio donde se instaló DB2 Everyplace. Para Pocket PC (WinCE), puede encontrar los archivos en `\db2everyplace\clients\`**`wince`**`\database\visualbasic`. Para sistemas operativos Win32, los puede encontrar en `\db2everyplace\clients\`**`win32`**`\database\visualbasic`.

La aplicación Visual Basic de ejemplo incluye los archivos siguientes:

`db2evb.bas`
El archivo `db2evb.bas` contiene la aplicación Visual Basic de ejemplo. Puede utilizar la aplicación de ejemplo como guía para escribir su propia aplicación Visual Basic.

`db2ecli.bas`
El archivo `db2ecli.bas` es la interfaz de Visual Basic que conecta con la base de datos DB2 Everyplace. Este archivo también define diversas restricciones de DB2 Everyplace que se encuentran en `sqlcli.h`, `sqlcli1.h`, `sqlext.h` y `sqlsystm.h`. Este archivo sólo contiene las restricciones de uso más frecuente. Si es necesario, puede añadir otras restricciones contenidas en `sqlcli.h`, `sqlcli1.h`, `sqlext.h` y `sqlsystm.h`.

DB2eForms (las extensiones varían según el sistema operativo)
Archivo de la interfaz de usuario de la aplicación.

DB2eSample.exe (para WinCE, DB2eSample.vb)
Archivo ejecutable de la aplicación.

DB2eSample.vbw
Archivo de proyecto de la aplicación.

DB2eSample.vbp (para WinCE, DB2eSample.ebp)
Archivo de proyecto de la aplicación.

© Copyright IBM Corp. 1999,2003

Ejemplo para Visual Basic: db2evb.bas:

Los pasos principales utilizados en la aplicación de ejemplo (`db2evb.bas`) son:

Conectar con la base de datos DB2 Everyplace.

- Paso 1: Asigne un descriptor de entorno.
- Paso 2: Asigne un descriptor de base de datos.
- Paso 3: Conecte con la base de datos.
- Paso 4: Asigne un descriptor de sentencia.

Acceder a datos de DB2 Everyplace.

- Paso 5: Cree una tabla.
- Paso 6: Inserte datos en la tabla.
- Paso 7: Recupere datos de la tabla.

Termine la aplicación de aplicación.

Nota: Antes de salir, asegúrese de que la aplicación cierra la conexión con la base de datos DB2 Everyplace.

En el presente ejemplo se han añadido comentarios para describir los pasos de la aplicación de ejemplo.

```
Option Explicit

Public henv    As Long          ' Descriptor de entorno
Public hdbc    As Long          ' Descriptor de base de datos
Public hstmt   As Long          ' Descriptor de sentencia
Public rc      As Integer       ' Código de retorno

Public dbpath As String     ' vía del sistema de archivos donde DB2e creará tablas.
Public userid As String     ' ID de usuario: no utilizado por DB2 Everyplace.
Public pass   As String     ' Contraseña: no utilizada por DB2 Everyplace

'--------------------------------------------------------------------------------
' Función: DB2eTest
'
' Descripción: Función que muestra cómo pueden hacerse llamadas a DB2 Everyplace.
'
'--------------------------------------------------------------------------------

Public Function DB2eTest() As Integer
    Dim errmsg As String
    Dim numCols As Integer
    Dim i As Integer
    Dim retLen As Long
    Dim data As String
    Dim crtStmt As String
    Dim insStmt1 As String
    Dim insStmt2 As String
    Dim selStmt As String
    On Error Resume Next 'Importante: no me pregunten porqué, pero esta línea
                         'es necesaria en cada función que llama a funciones
                         'de db2e.dll; de no incluirla, visual basic hace
                         'cosas misteriosas.    '
    dbpath = ""
    userid = ""
    pass = ""
    '
    crtStmt = "CREATE TABLE x(a INT, b TIMESTAMP)"
    insStmt1 = "INSERT INTO x VALUES(1, CURRENT TIMESTAMP)"
    insStmt2 = "INSERT INTO x VALUES(2, CURRENT TIMESTAMP)"
    selStmt = "SELECT * FROM x"
    '
    data = String(80, " ")
    ' Paso 1: asignar un descriptor de entorno.
    '

    DB2eForm.DB2eText.Text = vbCrLf & vbCrLf & "Asignar descriptor de entorno"

    rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HENV, henv)
    If (rc <> 0) Then
        rc = DB2eError()
        rc = DB2eTerminate()
```

```
    Exit Function
End If

'
' Paso 2: asignar un descriptor de base de datos
'

DB2eForm.DB2eText.Text = DB2eForm.DB2eText.Text & vbCrLf &
                         " Asignación de un descriptor de base de datos"

rc = SQLAllocHandle(SQL_HANDLE_DBC, henv, hdbc)
If (rc <> 0) Then
    rc = DB2eError()
    rc = DB2eTerminate()
    Exit Function
End If

'
' Paso 3: conectar con la base de datos
'
DB2eForm.DB2eText.Text = DB2eForm.DB2eText.Text & vbCrLf &
                         "  Conexión con la base de datos"

rc = SQLConnect(hdbc, dbpath, SQL_NTS, userid, SQL_NTS, pass, SQL_NTS)
If (rc <> 0) Then
    rc = DB2eError()
    rc = DB2eTerminate()
    Exit Function
End If

'
' Paso 4: asignar un descriptor de sentencia.
'

DB2eForm.DB2eText.Text = DB2eForm.DB2eText.Text & vbCrLf &
                         " Asignación de un descriptor de sentencia"

rc = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, hstmt)
If (rc <> 0) Then
    rc = DB2eError()
    rc = DB2eTerminate()
    Exit Function
End If

    DB2eForm.DB2eText.Text = DB2eForm.DB2eText.Text & vbCrLf

'
' Ahora puede utilizar llamadas de función CLI para ejecutar sentencias SQL.
'
' Paso 5: crear una tabla
'

DB2eForm.DB2eText.Text = DB2eForm.DB2eText.Text & vbCrLf & "  " & crtStmt
rc = SQLExecDirect(hstmt, crtStmt, SQL_NTS)
If (rc <> 0) Then
    rc = DB2eError()
    rc = DB2eTerminate()
    Exit Function
End If

'
' Cree de nuevo la misma tabla para forzar un mensaje de error y
' ver si DB2eError funciona.
'
'rc = SQLExecDirect(hstmt, "create table p(a int)", SQL_NTS)
'If (rc <> 0) Then
'   testmsg = MsgBox("BLA1", 1, "DB2 Everyplace Visual Basic")
'   rc = DB2eError()
'   testmsg = MsgBox("BLA2", 1, "DB2 Everyplace Visual Basic")
'   rc = DB2eTerminate()
'   testmsg = MsgBox("BLA3", 1, "DB2 Everyplace Visual Basic")
'   Exit Function
'End If
'

'
' Paso 6: insertar datos en la tabla.
'

DB2eForm.DB2eText.Text = DB2eForm.DB2eText.Text & vbCrLf & "  " & insStmt1
rc = SQLExecDirect(hstmt, insStmt1, SQL_NTS)
If (rc <> 0) Then
    rc = DB2eError()
    rc = DB2eTerminate()
```

```
        Exit Function

    End If

    DB2eForm.DB2eText.Text = DB2eForm.DB2eText.Text & vbCrLf & "  " & insStmt2
    rc = SQLExecDirect(hstmt, insStmt2, SQL_NTS)
    If (rc <> 0) Then
        rc = DB2eError()
        rc = DB2eTerminate()
        Exit Function

    End If

        DB2eForm.DB2eText.Text = DB2eForm.DB2eText.Text & vbCrLf

    '
    ' Paso 7: recuperar datos de la tabla.
    '

    DB2eForm.DB2eText.Text = DB2eForm.DB2eText.Text & vbCrLf & "  " & selStmt
                           & vbCrLf

    rc = SQLExecDirect(hstmt, selStmt, SQL_NTS)
    If (rc <> 0) Then
        rc = DB2eError()
        rc = DB2eTerminate()
        Exit Function
    End If


    rc = SQLNumResultCols(hstmt, numCols)
    If (rc <> 0) Then
        rc = DB2eError()
        rc = DB2eTerminate()
        Exit Function
    End If

    Do While (SQLFetch(hstmt) = SQL_SUCCESS)
        For i = 1 To numCols
            rc = SQLGetData(hstmt, i, SQL_C_CHAR, data, 80, retLen)
            DB2eForm.DB2eText.Text = DB2eForm.DB2eText.Text & "  " & data & vbCrLf
    If (rc <> 0) Then
        rc = DB2eError()
        rc = DB2eTerminate()
                Exit Function
            End If
        Next
        data = String(80, " ")
        DB2eForm.DB2eText.Text = DB2eForm.DB2eText.Text & vbCrLf
    Loop

    '
    ' Paso 8: cerrar conexión con la base de datos DB2e antes de que
    ' la aplicación termine.
    '
        rc = DB2eTerminate()

    DB2eTest = 0
End Function
```

Tareas afines:

- "Cómo desarrollar aplicaciones de Visual Basic de DB2 Everyplace" en la página 31

Compilación y prueba del programa Visual Basic de ejemplo

Procedimiento:

Para compilar y probar el programa de ejemplo de DB2 Everyplace:

1. Abra el archivo de proyecto Visual Basic `DB2eSample.vbp` (para WinCE, `DB2eSample.ebp`).
2. Cree el programa de ejemplo.
   - Para Win32: Seleccione **Archivo –> DB2eSample.exe.**. Se creará `DB2eSample.exe`.

- Para WinCE: Seleccione **Archivo –> DB2eSample.vb**. Se creará `DB2eSample.vb`.

3. Copie los archivos siguientes:
   - Para Win32: Copie `DB2e.dll` (para el sistema operativo Win32) en el directorio actual del proyecto o en la vía de acceso de `DB2e.dll` en la variable de entorno PATH.
   - Para WinCE: Copie `DB2eSample.vb`, `DB2e.dll` (para el sistema operativo Pocket PC) y Visual Basic Runtime en el directorio de su elección.
4. Ejecute `DB2Sample.exe` (para WinCE, `DB2Sample.vb`).

Conceptos afines:

- "Visión general de la aplicación Visual Basic de ejemplo" en la página 111

Consulta relacionada:

- "Resumen de las funciones de CLI de DB2" en la página 194
- "Sistemas operativos soportados de la interfaz de Visual Basic" en la página 32

Capítulo 16. Las aplicaciones JSP de ejemplo

Los archivos que se listan a continuación están relacionados con el directorio <DB2Everyplace>\SDK\JSP\sample\jsp. Todas las aplicaciones JSP de ejemplo utilizan la base de datos de ejemplo Visiting Nurse en el directorio <DB2Everyplace>\SDK\JSP\sample\data. Las guías de aprendizaje están en el directorio <DB2Everyplace>\SDK\JSP\doc.

Las aplicaciones desarrolladas utilizando WebSphere Studio Professional/Entry Edition v4.0

Planificación de Visiting Nurse

Descripción

Este ejemplo consulta dinámicamente la base de datos Visiting Nurse y muestra los resultados en una tabla.

Página de Inicio

VNSchedule_ws40\scheduleHTMLResults.jsp

Guía de aprendizaje

ws40.pdf

Las aplicaciones desarrolladas utilizando WebSphere Studio Application Developer v4.0

Planificación de Visiting Nurse

Descripción

Este ejemplo consulta dinámicamente la base de datos Visiting Nurse y muestra los resultados en una tabla. Esta aplicación de ejemplo requiere JDBC 2.0 y no puede ejecutarse en Symbian OS Versión 6.

Página de Inicio

VNSchedule_wsad40\scheduleMasterView.jsp

Otros archivos

VNSchedule_wsad40\web.xml
VNSchedule_wsad40\dbbeans.jar

Guía de aprendizaje

wsad40.pdf

Las aplicaciones desarrolladas utilizando WebSphere Studio Application Developer v5.0

Planificación de Visiting Nurse

Descripción

Este ejemplo consulta dinámicamente la base de datos Visiting Nurse y muestra los resultados en una tabla.

Página de Inicio

VNSchedule_wsad50\scheduleMasterView.jsp

Otros archivos

VNSchedule_wsad50\web.xml
VNSchedule_wsad50\dbbeans.jar
VNSchedule_wsad50\scheduleMasterViewBean.class

Guía de aprendizaje

wsad50.pdf

© Copyright IBM Corp. 1999,2003

Las aplicaciones desarrolladas fuera de WebSphere Studio
Visiting Nurse

Descripción

Consulte la publicación *DB2 Everyplace Guía del usuario y de instalación* para obtener una descripción de la aplicación de ejemplo Visiting Nurse.

Página de Inicio

```
VisitingNurse\schedule.jsp
```

Otros archivos

```
VisitingNurse\contact.jsp
  VisitingNurse\medrecord.jsp
  VisitingNurse\person.jsp
```

Tareas afines:

- "Verificación del soporte de JSP en una estación de trabajo Windows" en la página 35
- "Configuración del desarrollo de JSP en un dispositivo Windows CE" en la página 36

Conceptos afines:

- "Desarrollo de aplicaciones JSP de DB2 Everyplace" en la página 33
- "Visión general del soporte de JSP en DB2 Everyplace" en la página 34

Capítulo 17. Aplicaciones de sincronización de ejemplo

Este capítulo proporciona información sobre las aplicaciones de sincronización de ejemplo. Los temas que se tratan son:

- "La aplicación Sync Client C/C++ de ejemplo"
- "Las aplicaciones de sincronización nativa de Java de ejemplo" en la página 121
- "Las aplicaciones de sincronización MIDP de Java de ejemplo" en la página 125
- "Desarrollo de la aplicación isync4j para MIDP con Sun Wireless Toolkit" en la página 129
- "Desarrollo de la aplicación isync4j para MIDP con ANT y la línea de mandatos de Sun Wireless Toolkit" en la página 131
- "Compilación y ejecución de la aplicación de sincronización de Java de ejemplo GoISyncConsole" en la página 132

La aplicación Sync Client C/C++ de ejemplo

El ejemplo siguiente ilustra cómo utilizar un número seleccionado de funciones de la API de DB2 Everyplace Sync Client para crear una aplicación. Puede encontrar más ejemplos de código fuente en `\DB2e\Clients\clientapisample\C_API`.

```
/*****************************************************************/
/**
 * Esta función define la función de escucha de la sincronización.
 * Vea isyncore.h para obtener más información.
 * parámetro: listenerData, datos personales del usuario.
 * parámetro: suceso, objeto de suceso
 * parámetro: pExtraInfo (reservado)
 * devolución: entero, si el tipo de suceso es ISCEVTTYPE_Retry:
 *   . ISCRTNCB_ReplyYes    : reintentar menos de 3 veces
 *   . ISCRTNCB_ReplyNo     : reintentar 3 o más veces
 *     si el tipo de suceso es ISCEVTTYPE_Info:
 *   . ISCRTNCB_Done
 *     si tipo suceso es ISCEVTTYPE_Query y código suceso es ISCEVT_QueLogin:
 *   . ISCRTNCB_Done        : nombusuario y contraseña entrados correctamente
 *   . ISCRTNCB_Default     : nombusuario y contraseña no entrados
 *     otros (ISCEVTTYPE_Fatal, ISCEVTTYPE_Error, ISCEVTTYPE_Query
 *            e ISCEVTTYPE_Conflict)
 *   . ISCRTNCB_Default     : emprender la acción por omisión
 **/
static isy_INT32 syncListener(
       isy_UINT32 listenerData,
       ISCEVT    *event,
       isy_VOID  *pExtraInfo)
{
    // appEventCodeToMessage es alguna función de usuario para correlacionar
    // un suceso con algún mensaje descriptivo del suceso
    char *statusMsg = appEventCodeToMessage(event);
    int   timesRetried;

    switch (event->type) {
        case ISCEVTTYPE_Fatal:
        case ISCEVTTYPE_Error:
            printf("Error: %s\n", statusMsg);
            return ISCRTNCB_Default ;

        case ISCEVTTYPE_Retry:
            timesRetried = event->retry;
            if (timesRetried >= 3)
                return ISCRTNCB_ReplyNo;
            else {
                char ans;
                printf("%s [Y/N] ", statusMsg);
```

© Copyright IBM Corp. 1999,2003

```
                ans = getchar();
                getchar();
                if(tolower(ans) == 'y')
                        return ISCRTNCB_ReplyYes;
                else
                        return ISCRTNCB_ReplyNo;
            }

          case ISCEVTTYPE_Info:
              switch (event->code) {
                  case ISCEVT_InfSucceeded:
                  case ISCEVT_InfFailed:
                  case ISCEVT_InfCanceled:
                      printf("Conclusion: %s\n", statusMsg);
                      break;
                  case ISCEVT_InfGeneral:
                  case ISCEVT_InfCancelingSync:
                  case ISCEVT_InfPrepMsg:
                  case ISCEVT_InfSendMsg:
                  case ISCEVT_InfWaitMsg:
                  case ISCEVT_InfApplyMsg:
                      printf("Status: %s\n", statusMsg);
                      break;
                  default: // ignorar otro código de suceso
                      break;
              } // switch (event->code)
              return ISCRTNCB_Done;

          case ISCEVTTYPE_Query:
              if (event->code == ISCEVT_QueLogin) {
                  ISCLISTENARG *args = event->info;
                  isy_TCHAR *target = args->argv[0];
                  // Es sólo un ejemplo, no pretende estar libre de fugas de memoria.
                    isy_TCHAR *username = (isy_TCHAR *) calloc(18, sizeof(isy_TCHAR));
                    isy_TCHAR *password = (isy_TCHAR *) calloc(254, sizeof(isy_TCHAR));
         char c;
                    int  i;

                  printf("Query on target data(%s): %s ...\n", target, statusMsg);
                  // Solicitar el nombre de usuario
                  printf("Username: ");
                  for(i = 0; (c = getchar()) != '\n'; i++) username[i] = c;
                  username[i] = '\0';
                  if (i == 0) return ISCRTNCB_Default; // nombusuario no entrado
                  // Solicitar la contraseña
                  printf("Password: ");
                  for(i = 0; (c = getchar()) != '\n'; i++) password[i] = c;
                  password[i] = '\0';
        args->argv[1] = username;
                    args->argv[2] = password;
                 return ISCRTNCB_Done;
              }
              return ISCRTNCB_Default;

            // todos los otros tipos de suceso no importan
            default:
              return ISCRTNCB_Default;
      } // switch (event->type)
}

// Ejemplo de SyncClient
main()
{
    isy_TCHAR user[]     = isy_T("usuario1");
    isy_TCHAR password[] = isy_T("contraeña");
    HISCSERV  hServ;
    HISCCONF  hConf;
    HISCENG   hEngine;
    isy_INT32 rc;

    rc = iscConfigOpen(hServ, isy_T(".\isyncPath"), &hConf;);
    rc = iscEngineOpen(hConf, &hEngine;);
    iscEngineSetListener(hEngine, syncListener, NULL);
```

```
    iscEngineSyncConfig(hEngine); // obtener primero la configuración
    iscConfigEnableSubsSet(hConf, NULL); // habilitar todos conj. suscripción
    rc = iscEngineSync(hEngine); // sincronizar config + conjuntos suscripción

    if (rc == ISCRTN_Failed) {
        HISCCSR   hCursor;
        isy_TCHAR  id[ISCLEN_SubsSetID];
        isy_TCHAR  name[ISCLEN_SubsSetName];
        isy_INT32  enabled;

        iscConfigOpenCursor(hConf, &hCursor;);
        while (iscConfigGetNextSubsSet(hConf, hCursor, id, name)
               == ISCRTN_Succeeded) {
            enabled = iscConfigSubsSetIsEnable(hConf, id);
            if (enabled != ISCRTN_True) continue; // olvidar los que se han
                                                  // inhabilitado
            rc = iscConfigGetSubsSetStatus(hConf, id);
            if (rc != ISCRTN_Succeeded)
// En ese caso, la aplicación puede tener cierto código que
// procese los conjuntos de suscripción no satisfactorios aquí.
// Para inhabilitar el conjunto de suscripciones, llame a:

                iscConfigDisableSubsSet(hConf, id);
        }
        iscConfigCloseCursor(hConf, hCursor);
        rc = iscEngineSync(hEngine); // sincronizar config + conjuntos suscripción
    }
    // cerrar todos los descriptores
    iscEngineClose(hEngine);
    iscConfigClose(hConf);
    iscServiceClose(hServ);
} // principal
```

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Herramientas de desarrollo de C/C++ soportadas" en la página 10

Las aplicaciones de sincronización nativa de Java de ejemplo

Existen varios programas Java de ejemplo que puede utilizar como ayuda para escribir aplicaciones de sincronización Java para DB2 Everyplace.

Consulte el apartado denominado "Visión general de las aplicaciones de ejemplo" de la publicación *DB2 Everyplace Guía del usuario y de instalación* para obtener más información sobre el lugar en que están ubicados los ejemplos.

El programa de ejemplo ISyncSample.java demuestra cómo codificar una aplicación Sync Client para el proveedor de sincronización nativa de DB2 Everyplace.

Los pasos principales de la aplicación de ejemplo ISyncSample.java son:

Paso 1:

Importar los paquetes de sincronización de DB2 Everyplace.

```
import com.ibm.mobileservices.isync.*;
import com.ibm.mobileservices.isync.event.*;
```

Para el proveedor de sincronización basado en JNI, importar com.ibm.mobileservices.isync.db2e.jni.*;

Para el proveedor de sincronización basado en la detección, importar com.ibm.mobileservices.isync.db2e.sti.*;

Paso 2:

Implementar el método eventIssued de la interfaz ISyncListener para la notificación de sucesos durante la sincronización.

Paso 3:

Obtener una instancia de DB2eISyncProvider

Paso 4:

Obtener del objeto de proveedor una instancia del servicio de sincronización

Paso 5:

Obtener del objeto de servicio una instancia del almacén de configuración

Paso 6:

Obtener del objeto de almacén de configuración una instancia del controlador de sincronización

Paso 7:

Registrar el objeto de escucha de la aplicación que implementa la interfaz ISyncListener para la notificación de sucesos del objeto de controlador de sincronización durante la sincronización

Paso 8:

Realizar una sincronización sobre todos los conjuntos de suscripción habilitados. Comprobar el código de retorno y el estado de excepción de la sincronización.

Paso 9:

Cerrar y liberar todos los recursos asignados por el proveedor de sincronización

```
// Ejemplo 1:  ISync Java - Uso de la API simple
//

 // Paso 1:  importar los paquetes Java de Sync Client
      //
      import com.ibm.mobileservices.isync.*;
      import com.ibm.mobileservices.isync.event.*;
      import com.ibm.mobileservices.isync.db2e.jni.*;


      // Paso 2:  implementar el método eventIssued() en
      //    la interfaz ISyncListener si está interesado en la notificación
      //    de sucesos (opcional)
      //
      public class ISyncSample implements ISyncListener {

         public ISyncSample () {}

         public int eventIssued(ISyncEvent evt) {

            int evtType = evt.getEventType();

            switch(evtType) {

            // visualizar el estado del suceso
            case ISync.EVTTYPE_INFO:
            case ISync.EVTTYPE_ERROR:

               System.out.println ("**********************");
               System.out.println ("SubsSet:    " + evt.getSubscriptionSetName() );
               System.out.println ("Subs:       " + evt.getSubscriptionName() );
               System.out.println ("SubsType:   " + evt.getSubscriptionType() );
               System.out.println ("Event Type: " + evtType );
               System.out.println ("Event Code: " + evt.getEventCode() );
               System.out.println ("Progress:   " + evt.getSyncProgress());
               System.out.println ("**********************\n");

               return ISync.RTNCB_DONE;
```

```
            case ISync.EVTTYPE_RETRY:
               return ISync.RTNCB_REPLY_YES;

            case ISync.EVTTYPE_CONFLICT:
               return ISync.RTNCB_DONE;

            // ignorar otros tipos de suceso
            default:
               break;
         }

         // dejar que el motor de sincronización emprenda la acción por omisión
         return ISync.RTNCB_DEFAULT ;

      }

      public void runSample(String host, String port,
                            String userID, String passwrd) {

         ISyncProvider provider = null;
         ISyncService  service  = null;
         ISyncConfigStore config = null;
         ISyncDriver syncer = null;
         String      path = "data";         // dir de datos bajo dir actual
         ISyncSubscriptionSet ssArr[] = null;
         int rc = 0;

            try {

            // Paso 3:  obtener una instancia de DB2eISyncProvider
            //
            provider = DB2eISyncProvider.getInstance();

         // Paso 4:  obtener del proveedor una instancia
         //          de servicio de sincronización
            //
/*
 For the DB2j sync client, the JDBC driver and url are required

 String driver = "com.ibm.db2j.jdbc.DB2jDriver";
 String jdbcUrl = jdbc:db2j:crtlDb;create=true;
*/

if (driver != null)
    userProps.put("target.db.driver", driver);
if (jdbcUrl != null)
    userProps.put("target.db.url", jdbcUrl);

            Properties userProps = new Properties();

    userProps.put("isync.user", user);
    userProps.put("isync.password", password);
    userProps.put("isync.trace", "detailed");

    service = provider.createSyncService(uri, userProps);

            // Paso 5:  obtener una instancia del almacén de configuración
            //
            config = service.getConfigStore(path);

            // Paso 6:  obtener una instancia del controlador de
            //                                 sincronización
           syncer = config.getSyncDriver();

            // Paso 7:  establecer objeto de escucha para notificación
            //          de sucesos a partir del objeto sincronizador
            //          durante la sincronización (opcional)
            syncer.setSyncListener(this);

            // Paso 8:  realizar sincronización en todos los conjuntos
            //          de suscripción habilitados
            //
            rc = syncer.sync();

            switch (rc) {
               case ISync.RTN_SUCCEEDED:
                  System.out.println("Synchronization succeeded");
                  break;

               case ISync.RTN_CANCELED:
                  System.out.println ("Synchronization canceled");
                  break;

               default:
```

```
            System.out.println ("Synchronization failed");
            break;
      }

      ssArr = config.getSubscriptionSets();

      for (int i=0; i < ssArr.length; i++ ) {

         System.out.print ("Subscription Set: " +
                           ssArr[i].getName() + " Status: ");

         switch(ssArr[i].getStatus()) {

            case ISync.STATUS_READY:
               System.out.println("READY");
               break;

            case ISync.STATUS_COMPLETED:
               System.out.println ("COMPLETED");
               break;

            case ISync.STATUS_CANCELED:
               System.out.println ("CANCELED");
               break;

            default:
               System.out.println ("FAILED");
               break;
         }
      }
   }
   catch (ISyncException ie) {
      System.out.println("Exception code: " + ie.getCode());
      ie.printStackTrace();
   }
    catch (Exception e) {
      e.printStackTrace();
   }
   finally {
      // Paso 9:  cerrar y liberar todos los recursos asignados
      //

      try {

         if (syncer != null) {
            syncer.close();
            syncer = null;
         }

         if (config != null) {
            config.close();
            config = null;
         }

         if (service != null) {
            service.close();
            service = null;
         }
      }
      catch(ISyncException ie2) {

         System.out.println("Exception code: " + ie2.getCode());
         ie2.printStackTrace();
      }
   }
}  // end runSample()


public static void main(String args[]) {

   String host     = "hostlocal";
   String port     = "8080";
   String userID   = "nurse1";
   String passwrd  = "nurse1";

   ISyncSample isa = new ISyncSample();

   if (args.length > 0) {
   if (args.length == 4)
   {
     host     = args[0];
     port     = args[1];
     userID   = args[2];
     passwrd  = args[3];
```

```
        }
else
          System.out.println("Usage: java ISyncSample [host] [port] " +
                             "[userid] [password]");
    }
       isa.runSample(host, port, userID, passwrd);

    }  // final de principal()

 }  // final de clase ISyncSample
```

Tareas afines:

- "Instalación y verificación del proveedor de sincronización nativa basada en la detección" en la página 24

Conceptos afines:

- "Las aplicaciones de sincronización MIDP de Java de ejemplo"

Consulta relacionada:

- "Las API de IBM Java Sync" en la página 19

Las aplicaciones de sincronización MIDP de Java de ejemplo

Existen varias aplicaciones Java de ejemplo que puede utilizar como ayuda para escribir aplicaciones para DB2 Everyplace. Para el proveedor de sincronización MIDP, los ejemplos están ubicados en:

`%DSYINSTDIR%/Clients/Midp/samples`

El ejemplo principal es la aplicación Visiting Nurse, bajo `com/ibm/mobileservices/demo`, `VNurse.java` y `NursesAid.jar`. Bajo el mismo directorio de ejemplos se encuentran dos archivos que forman una aplicación simple. Esta aplicación no proporciona código de RMS (Record Store Management) ni una interfaz de usuario sólida. Los archivos son:

- ISyncSample.java : control de MIDlet
- ISyncWorker.java : trabajador responsable de sincronizar los datos

Detalles del archivo ISyncWorker.java:

El programa de ejemplo `SyncWorker.java` demuestra cómo codificar una aplicación Sync Client para el proveedor de sincronización MIDP de DB2 Everyplace.

La aplicación Java de ejemplo realiza los pasos siguientes:

1. Importar los paquetes de sincronización de DB2 Everyplace.

```
      import com.ibm.mobileservices.isync.*;
import com.ibm.mobileservices.isync.event.*;
import com.ibm.mobileservices.isync.midp.*;
```

2. Implementar el método eventIssued de la interfaz ISyncListener para la notificación de sucesos durante la sincronización.
3. Obtener una instancia de MIDPISyncProvider
4. Obtener del objeto de proveedor una instancia del servicio de sincronización
5. Obtener del objeto de servicio una instancia del almacén de configuración
6. Obtener del objeto de almacén de configuración una instancia del controlador de sincronización
7. Registrar el objeto de escucha de la aplicación que implementa la interfaz ISyncListener para la notificación de sucesos del objeto de controlador de sincronización durante la sincronización

8. Realizar una sincronización sobre todos los conjuntos de suscripción habilitados. Comprobar el código de retorno y las excepciones para ver el estado de la sincronización.
9. Cerrar y liberar todos los recursos asignados por el proveedor de sincronización.

El ejemplo ISyncSample.java:

El ejemplo siguiente contiene comentarios que hacen referencia a los pasos del apartado anterior.

```
// Ejemplo 1:  ISync Java - Uso de la API simple
//

 // Paso 1:  importar los paquetes Java de Sync Client
      //
      import com.ibm.mobileservices.isync.*;
      import com.ibm.mobileservices.isync.event.*;
      import com.ibm.mobileservices.isync.midp.*;




/**
 Clase de soporte que maneja todas las tareas de sincronización.
 La llama ISyncSample.
*/

public class SyncWorker extends Thread implements ISyncListener
{
 private ISyncSample midlet;
 private boolean mCancel;
 private ISyncProvider provider;
 private ISyncService service;
 private ISyncConfigStore config;
 private ISyncDriver syncer;
 private String eventString;

 public SyncWorker(ISyncSample midlet)
 {
  this.midlet = midlet;
  mCancel = false;
 }

   // Paso 2:  implementar el método eventIssued() en la interfaz
   //          si está interesado en la notificación de sucesos (opcional)
   //

 public int eventIssued(ISyncEvent evt)
 {
  int evtType = evt.getEventType();
  int evtCode = evt.getEventCode();
  int evtProg = evt.getSyncProgress();
         String ssName = evt.getSubscriptionSetName();
  Object listenerInfo = evt.getEventInfo();

  Exception e = null;
  ConflictReader cr = null;

  if (listenerInfo instanceof Exception)
   e = (Exception) listenerInfo;
  else if (listenerInfo instanceof ConflictReader)
   cr = (ConflictReader) listenerInfo;

  eventString += evtCode + ":";

  switch(evtType)
  {
   // visualizar el estado del suceso
    case ISync.EVTTYPE_INFO:

              switch (evtCode)
      {
                case ISync.EVT_INF_SYNCING_SUBS:
                    midlet.updateSyncStat1("Synchronizing " + ssName);
                    midlet.updateSyncStat2(" ");
                    break;
                case ISync.EVT_INF_SYNC_STARTED:
                    midlet.updateSyncStat1("Synchronization started");
```

```
                midlet.updateSyncStat2(" ");
                break;
            case ISync.EVT_INF_PREP_MSG:
                midlet.updateSyncStat2("Preparing message...");
                break;
            case ISync.EVT_INF_SEND_MSG:
                midlet.updateSyncStat2("Sending message...");
                break;
            case ISync.EVT_INF_WAIT_MSG:
               midlet.updateSyncStat2("Awaiting server reply...");
               break;
            case ISync.EVT_INF_APPLY_MSG:
                midlet.updateSyncStat2("Applying server message...");
                break;
            case ISync.EVT_INF_SYNC_CANCELED:
     midlet.updateSyncStat1("Synchronization canceled");
     midlet.updateSyncStat2(" ");
     break;
     case ISync.EVT_INF_SYNC_SUCCEEDED:
     midlet.updateSyncStat1("Synchronization succeeded");
     midlet.updateSyncStat2(" ");
     break;
            case ISync.EVT_INF_SYNC_FAILED:
     midlet.updateSyncStat1("Synchronization failed");
     midlet.updateSyncStat2(" ");
     break;
            default:
                break;
          }

  return ISync.RTNCB_DONE;

  case ISync.EVTTYPE_ERROR:
   midlet.updateSyncStat2("Error: " + evtCode);
   return ISync.RTNCB_DONE;

  case ISync.EVTTYPE_RETRY:
   midlet.updateSyncStat2("Retry: " + evtCode);
   return ISync.RTNCB_REPLY_YES;

  case ISync.EVTTYPE_CONFLICT:
  if (evtCode == ISync.EVT_CFT_REJECT)
  {
   String tabName = evt.getSubscriptionName();
                    midlet.updateSyncStat2("Conflict: " + tabName);

   /*
    La aplicación tiene que hacer lo correcto con conflictRow.
   */

   // System.out.println("Conflict table " + tabName
    // + " row: " + conflictRow);
  }
  return ISync.RTNCB_DONE;

  // ignorar otros tipos de suceso
  default:
   break;
  }

  // dejar que el motor de sincronización emprenda la acción por omisión
 return ISync.RTNCB_DEFAULT ;

}  // final de eventIssued()
/*
 Se implementa la sincronización en una hebra para permitir que el
 usuario cancele la petición que, siendo de una sola hebra, se puede
 colgar en una petición de E/S
*/

public void run()
{
 sync();
}

public void cancel()
{
 try
 {
  if (syncer != null)
   syncer.cancelSync();
 }
 catch (ISyncException iex)
 {}
```

```
 mCancel = true;
}

private void sync()
{
 try
 {
  eventString = " ";

  String user = "nurse1";
  String password = "nurse1";
  String host = "hostlocal";
  String port = "9080";

  /*
   Si el archivo jad tiene valores, utilícelos, vea
   DeployManifest.java en las herramientas.

   En Sun WirelessToolkit, bajo Settings, puede entrar
   valores en la pestaña User Defined.
  */

  String x = midlet.getAppProperty("Db2eSyncUserName");
  if (x != null)
   user = x;
  x = midlet.getAppProperty("Db2eSyncPassword");
  if (x != null)
   password = x;
  x = midlet.getAppProperty("Db2eSyncHost");
  if (x != null)
   host = x;
  x = midlet.getAppProperty("Db2eSyncPort");
  if (x != null)
   port = x;
  midlet.appendForm(host + ":" + port + " " + user + "/" +
          password);


        // Paso 3:  obtener una instancia de MIDPISyncProvider
        //
  provider = MIDPISyncProvider.getInstance();

        // Paso 4:  obtener del proveedor una instancia
        //          de servicio de sincronización
        //
   Hashtable ht = new Hashtable();
   ht.put("isync.user", userName);
   ht.put("isync.password", password);
   ht.put("isync.trace", "detailed");

   service = provider.createSyncService(URI, ht);

        // Paso 5:  obtener una instancia del almacén de configuración
        //
  config = service.getConfigStore(null);

        // Paso 6:  obtener una instancia del controlador de
        //          sincronización para realizar la sincronización
        //
  syncer = config.getSyncDriver();

        // Paso 7:  establecer objeto de escucha para la notificación
        //     de sucesos desde objeto sincronizador durante la sincronización
        //
  syncer.setSyncListener(this);

        // Paso 8:  realizar sincronización en todos los conjuntos
        //          de suscripción habilitados
        //

  int rc = syncer.sync();

  switch (rc)
  {
    case ISync.RTN_SUCCEEDED:
   midlet.reportSyncStatus("Synchronization succeeded "
    + eventString);
   break;

    case ISync.RTN_CANCELED:
   midlet.reportSyncStatus("Synchronization canceled "
     + eventString);
   break;
```

```
    default:
   midlet.reportSyncStatus("Synchronization failed "
     + eventString);
   break;
  }
 //  Paso 9: Cerrar todos los recursos
 //

  close();
 }
 catch (ISyncException iex)
 {
  midlet.reportSyncStatus("Exception Code: "
   + iex.getCode() + ", Event codes: " + eventString);
 }
 catch (Exception e)
 {
  midlet.reportSyncStatus(e.toString());
 }
 finally
 {
  mCancel = false;
 }
}

private void close() throws ISyncException
{
 if (syncer != null)
 {
  syncer.close();
  syncer = null;
 }
 if (config != null)
 {
  config.close();
  config = null;
 }

 if (service != null)
 {
  service.close();
  service = null;
 }

 provider = null;
}
}
```

Tareas afines:

- "Desarrollo de la aplicación isync4j para MIDP con Sun Wireless Toolkit"
- "Desarrollo de la aplicación isync4j para MIDP con ANT y la línea de mandatos de Sun Wireless Toolkit" en la página 131

Conceptos afines:

- "Las aplicaciones de sincronización MIDP de Java de ejemplo" en la página 125

Desarrollo de la aplicación isync4j para MIDP con Sun Wireless Toolkit

En este tema se describe cómo desarrollar ISYNC4J de DB2 Everyplace para MIDP dentro de la aplicación Sun Wireless Toolkit. Los ejemplos utilizados en este apartado se basan en la aplicación de ejemplo VNurse.

Requisitos previos:

Consulte "Visión general de los proveedores de sincronización de DB2 Everyplace" en la página 19 para obtener información más detallada sobre los requisitos previos de hardware y software para utilizar el proveedor de sincronización DB2 Everyplace J2ME para MIDP.

Procedimiento:

1. Activación del Wireless Toolkit. Desde un indicador de línea de mandatos, cambie al directorio `bin` en el que está instalado Sun Wireless Toolkit. Escriba **ktoolbar.bat**.

   Nota: Es aconsejable utilizar un indicador de mandatos en lugar del menú **Inicio** de Windows.
2. Cree un nuevo proyecto para la aplicación de ejemplo `isync4j`:
   a. Abra el J2ME Wireless Toolkit
   b. Seleccione **New Project**.
   c. Escriba el nombre de proyecto (por ejemplo, VNurse)
   d. Escriba un nombre de clase MIDlet (por ejemplo, com.ibm.mobileservices.demo.VNurse)
   e. Pulse **Create Project**.
   f. Copie el archivo `ISyncMidp.jar` a la biblioteca de proyectos J2ME. Por ejemplo:

      ```
      c:\>copy %DSYINSTDIR%\Clients\Midp\lib\ISyncMidp.jar \
        j2me_install_dir\apps\VNurse\lib.
      ```
   g. Opcional: Si desea ver la salida del rastreo mientras se ejecuta el MIDlet, copie `ISyncMidpDebug.jar` a *dir_instalación_j2me* `\apps \VNurse \lib`.

      Nota: No utilice el rastreo cuando cree un archivo JAR que se vaya a instalar en el teléfono. El archivo JAR resultante será demasiado grande para instalarlo.
   h. Opcional: Para utilizar la ofuscación (reducir el tamaño del código), copie el `archivo retroguard.jar` al directorio bin en que está instalado J2ME.
   i. Pulse **Settings** . Se abre la ventana Settings para el proyecto. Pulse la pestaña **User Defined** y **Add** para escribir las entradas **Key** y **Value** siguientes:
      - Db2eSyncPassword, nurse1 [default]
      - Db2eSyncUserName, nurse1 [default]
      - PacketDownSize, 2800 [default 30000]
      - PacketUpSize, 1400 [default 30000]
      - Db2eSyncHost, localhost [default]
      - Db2eSyncPort, 9080 [default]

      Sun Wireless Toolkit coloca estos valores en el archivo .jad y el MIDlet los lee durante la ejecución.
   j. La aplicación de ejemplo de DB2 Everyplace (VNurse) visualizará un archivo de imágenes PNG. Pulse la pestaña **MIDlets** y seleccione **MIDlet-1**. Pulse **Edit** y cambie `VNurse.png` por `ibm.png`. Tendrá que copiar `ibm.png` desde el directorio `Midp\samples\images` al directorio *dir_instalación_J2ME*`\apps\VNurse\res`.
3. Importe los archivos de ejemplo de DB2 Everyplace al proyecto. Por ejemplo, copie la estructura de directorios de `%DSYINSTDIR%\Clients \Midp \samples \com` al directorio *dir_instalación_J2ME* `\apps \VNurse \src`.
4. Cree y ejecute la aplicación de ejemplo VNurse. En la ventana de Sun Wireless Toolkit, pulse **Build** y **Run**.

Tareas afines:

- "Desarrollo de la aplicación isync4j para MIDP con ANT y la línea de mandatos de Sun Wireless Toolkit" en la página 131

Conceptos afines:

- "Visión general de los proveedores de sincronización de DB2 Everyplace" en la página 19

Desarrollo de la aplicación isync4j para MIDP con ANT y la línea de mandatos de Sun Wireless Toolkit

En este tema se describe cómo desarrollar ISYNC4J de DB2 Everyplace para MIDP con ANT y la Línea de mandatos de Sun Wireless Toolkit.

Requisitos previos:

Baje e instale el software siguiente para trabajar con los ejemplos suministrados:

- Sun Microsystems JavaTM 2 Platform Micro Edition, Wireless Toolkit
- Apache ANT
- RetroGuard Ofuscator

Procedimiento:

1. Opcional: Recompile las demos si las desea modificar.

   El directorio `lib` contiene archivos JAD y JAR precompilados. Se proporcionan los scripts `build.bat` y `build.xml` para ilustrar el uso de Apache ANT, la herramienta DeployManifest y el ofuscador RetroGuard.

   a. Añada `retroInstallDir\lib\retroguard.jar` a la variable CLASSPATH. Establezca las variables siguientes en el entorno:
      - ANT_HOME – en la raíz de la instalación de ANT
      - DB2m_HOME – en el directorio %DSYINSTDIR%\Clients\Midp
      - J2MEWTK_HOME – en la raíz de la instalación de Sun Wireless Toolkit
      - JAVA_HOME – en la raíz de la instalación de jdk13 o jdk131 (únicamente)
      - JAVA14_HOME – en la raíz del directorio jdk14.

   b. Ejecute el archivo `build.bat` en la raíz del directorio de clientes MIDP para repoblar el directorio `lib` bajo MIDP con nuevos archivos JAR y JAD. Existe un archivo JAR y varios archivos JAD para cada configuración de id de usuario y de dispositivo.

      Encontrará varios directorios build*classes nuevos, que se utilizan a efectos de verificación previa y ofuscación. Existe un archivo JAR y varios archivos JAD para cada configuración de ID de usuario y de dispositivo. Vea en los archivos JAD cómo están establecidos el ID de usuario, la contraseña y el ID de dispositivo, y cómo se pasan a la aplicación MIDLet.

   c. La clase DeployManifest está incluida en lib\FilterServlet*.jar y se la llama desde el archivo `build.xml`. Utilice esta clase para generar el archivo JAR Manifest y el archivo JAD. Cuando genere los archivos, utilice la sintaxis siguiente.

      Para generar el archivo Manifest:

      ```
      java DeployManifest -m <nombreMidlet> <nombreClase> <nombreArchivoImágenes> \
        <nombreArchivoSalida>
      ```

      Para generar el archivo JAD:

      ```
      java DeployManifest -j <nombreJarMidlet> -U <máxPaquetesCarga> -D \
        <máxPaquetesBajada> -n <númClientes> <nombreBaseJad> <nombreArchivoSalida>
      ```

   d. Apache ANT llama internamente a la clase DeployManifest desde `build.xml`. Edite las entradas setJad del archivo `build.xml` para cambiar de forma permanente el ID de usuario, la contraseña u otros atributos. Los valores por omisión son *nurse1* y *nurse1*.

2. Ejecute la aplicación insync4j.

   La instalación de DB2 Everyplace crea una base de datos Vnurse con conjuntos de suscripción, usuarios y grupos.

   a. Seleccione **Inicio –> DB2 Everyplace –> Start MDAC** y verifique que existe un usuario llamado *nurse1*. La contraseña de este usuario está establecida en *nurse1*. Puede utilizar este nombre de usuario o puede editar el archivo `lib\<midlet>.jad` que pasa al script de ejecución. Observe que, cada vez que compile los cambios efectuados, se sobregrabarán los archivos JAD. Para cambiar el usuario y la contraseña de forma permanente, vea `samples\DeployManifest.java`.

   b. Debe iniciar el Sync Server utilizando Tomcat o Websphere Versión 4.0 o posteriores. Las conexiones HTTP desde teléfonos MIDP utilizan la codificación de transferencia de HTTP, que requiere un motor de servlet que soporte la especificación HTTP Servlet 2.3 y HTTP 1.1.

   c. Ejecute el archivo BATCH del directorio `Midp\bin` pasándole el nombre de un archivo JAD del directorio `Midp\lib`:

      - Para ejecutar la versión sin depuración del demo, escriba:

        ```
        run VNurse
        ```

      - Para ejecutar la versión con depuración utilizando ″nurse3″ como ID de usuario y contraseña del dispositivo número 213, escriba:

        ```
        run VNurseDebug3
        ```

El J2ME MIDP Sync Client utiliza las interfaces y clases que están definidas en el paquete com.ibm.mobileservices.isync.midp, así como las de los paquetes com.ibm.mobileservices.isync y com.ibm.mobileservices.isync.event.

Tareas afines:

- “Desarrollo de la aplicación isync4j para MIDP con Sun Wireless Toolkit” en la página 129

Consulta relacionada:

- “Las API de IBM Java Sync” en la página 19
- “Sistemas operativos soportados por la API de Java Sync” en la página 19

Compilación y ejecución de la aplicación de sincronización de Java de ejemplo GoISyncConsole

GoISyncConsole es una aplicación de ejemplo para mostrar la utilización de la API de Java de DB2 Everyplace Sync Client.

Contenido del archivo de GoISyncConsole:

- `GoISyncConsole.java`
- `GoISyncConstants.java`
- `GoISyncListener.java`
- `isyncdb2.properties`
- `isyncdb2e.properties`
- `isyncdb2j.properties`

Requisitos previos:

- Instalación y configuración del DB2 Everyplace Sync Server.

- Instalación de los binarios del Cliente de sincronización en el dispositivo. Están ubicados en los directorios `Clients\`*`plataforma`*`\sync`.
- Si está utilizando un cliente de Cloudscape, instalación de Cloudscape en el dispositivo.

Procedimiento:

1. Compile la aplicación de GoISyncConsole:

   Esto requiere el archivo `isync4j.jar`, el cual es uno de los binarios de Cliente de sincronización.

   a. Abra un indicador de mandatos.

   b. Escriba el mandato siguiente:

      ```
      javac -classpath isync4j.jar *.java
      ```

2. Configure el entorno:

   El entorno de vía de acceso debe configurarse de modo que puedan ubicarse los binarios de Cliente de sincronización.

   - Para Win32: Establezca la variable PATH para incluir la carpeta en la que están ubicados los binarios del Cliente de sincronización.
   - Para Linux o Neutrino: Exporte la LD_LIBRARY_PATH para incluir la carpeta en la que están ubicados los binarios del Cliente de sincronización.

3. Ejecute el ejemplo:

   GoISyncConsole puede utilizarse con el cliente C o con el cliente DB2j de Java. Se utiliza un archivo de propiedades para determinar el cliente que ha de utilizarse. Se proporcionan archivos de propiedades de ejemplo tanto para DB2e como para DB2j.

   - Para utilizar el cliente C, pase el archivo `isyncdb2e.properties` proporcionado escribiendo el mandato siguiente:

     ```
     java -classpath isync4j.jar;. GoISyncConsole isyncdb2e.properties
     ```

   - Para utilizar el cliente de DB2j de Java, incluya el jar de DB2j Sync Client, el archivo jar de Cloudscape y pase el archivo `isyncdb2j.properties` escribiendo el mandato siguiente (modifique el texto en cursiva en el caso de que el directorio de instalación de Cloudscape sea diferente):

```
java -classpath c:\cloudscape_5.1\lib\db2j.jar;
db2jisync.jar GoISyncConsole isyncdb2j.properties
```

   La aplicación se inicia con un menú de texto que contiene las opciones siguientes:

   - (1) Realizar sincronización
   - (2) Habilitar, inhabilitar o restaurar conjuntos de suscripciones
   - (3) Cambiar valores de servidor
   - (4) Ver las anotaciones cronológicas
   - (5) Acerca del Cliente de sincronización
   - (6) Salir

4. Especifique la opción (3) para configurar los valores de servidor. Esta opción le permitirá especificar la dirección IP del Servidor de sincronización, la contraseña y nombre de usuario de sincronización y otras opciones.
5. Especifique la opción (1) para realizar la sincronización.

   La aplicación GoISyncConsole crea otro archivo de propiedad denominado `ISync.properties` para guardar sus preferencias. Si cambia una propiedad en `isyncdb2e.properties` o en `isyncdb2j.properties`, debería suprimir

`ISync.properties` antes de volver a ejecutar GoISyncConsole para asegurarse de que los cambios nuevos entren en vigor.

Conceptos afines:

- "Visión general de los proveedores de sincronización de DB2 Everyplace" en la página 19

Parte 4. Consulta

Capítulo 18. Interfaces de programación de aplicaciones (las API) 137
Soporte de sentencia SQL de DB2 Everyplace. . . 137
Visión general del soporte de sentencia de SQL de DB2 Everyplace 137
CALL 138
CREATE INDEX 141
CREATE TABLE 143
DELETE 150
DROP 153
EXPLAIN 154
GRANT 156
INSERT 157
REORG TABLE. 160
REVOKE 161
SELECT 162
UPDATE 171
Compatibilidad entre tipos de datos para las operaciones de asignación y comparación . . . 176
Tipos de datos por omisión y simbólicos de SQL 177
Atributos de tipos de datos. 177
Listado de los SQLSTATE 180
Resumen de códigos de clase de SQLState. . . 180
Mensajes de SQLSTATE notificados por SQL 181
Mensajes de SQLSTATE notificados por CLI . . 185
Mensajes de SQLState notificados por JDBC . . 193
Funciones de CLI de DB2 soportadas 194
Resumen de las funciones de CLI de DB2 . . . 194
Clave para las descripciones de funciones de CLI de DB2 198
SQLAllocConnect—Asignar descriptor de conexión 199
SQLAllocEnv—Asignar descriptor de entorno 200
SQLAllocHandle—Asignar descriptor de contexto 200
SQLAllocStmt—Asignar un descriptor de sentencia 203
SQLBindCol—Enlazar una columna a una variable de aplicación 203
SQLBindParameter—Enlazar un marcador de parámetro a un almacenamiento intermedio . . 207
SQLConnect—Conectar con una fuente de datos 211
SQLColumns - Obtener información de columna para una tabla 216
SQLDescribeCol—Devolver un conjunto de atributos de una columna 220
SQLDisconnect—Desconectar de una fuente de datos 222
SQLEndTran—Solicitar COMMIT o ROLLBACK 223
SQLError—Recuperar información sobre errores 225
SQLExecDirect—Ejecutar una sentencia directamente 225
SQLExecute—Ejecutar una sentencia 227
SQLFetch—Recuperar la fila siguiente 229
SQLFetchScroll—Recuperar conjunto de filas y devolver datos para todas las columnas enlazadas. 231
SQLForeignKeys—Obtener la lista de columnas de clave foránea 238
SQLFreeConnect—Liberar descriptor de conexión 241
SQLFreeEnv—Liberar descriptor de entorno . . 242
SQLFreeHandle—Liberar recursos de descriptor de contexto 242
SQLFreeStmt—Liberar (o restaurar) un descriptor de sentencia 244
SQLGetConnectAttr—Obtener el valor actual de un atributo de conexión 246
SQLGetCursorName—Obtener nombre de cursor 248
SQLGetData—Obtener datos de una columna 250
SQLGetDiagRec—Obtener varios valores de campos del registro de diagnósticos 254
SQLGetInfo—Obtener información general . . 257
SQLGetStmtAttr—Obtener el valor actual de un atributo de sentencia 260
SQLNumParams - Obtener número de parámetros en una sentencia SQL 264
SQLNumResultCols—Obtener número de columnas resultantes 265
SQLPrepare—Preparar una sentencia 266
SQLPrimaryKeys—Obtener columnas de clave primaria de una tabla 268
SQLRowCount—Obtener número de filas . . . 271
SQLSetConnectAttr—Establecer opciones referentes a una conexión 272
SQLSetStmtAttr—Establecer opciones referentes a una sentencia. 276
SQLTables - Obtener información de tabla . . . 283
Conversión de datos por funciones de CLI de DB2 285
Métodos de JDBC soportados 287
Visión general del soporte de JDBC de DB2 Everyplace 287
Interfaces en el paquete de java.sql 287
Interfaz Blob 288
Interfaz CallableStatement 288
Interfaz Connection 290
Clase de DB2eConnection 291
Interfaz DatabaseMetaData 292
Interfaz Driver 295
Interfaz PreparedStatement 296
Interfaz ResultSet 297
Interfaz ResultSetMetaData 301
Interfaz Statement 303
Clase de DB2eStatement 304
Interfaces en el paquete javax.sql 305
Interfaz DataSource 306
Clases .NET soportadas 307
Miembros de DB2eCommandBuilder 307

© Copyright IBM Corp. 1999,2003

Miembros de DB2eCommand 308
Miembros de DB2eConnection. 309
Miembros de DB2eDataAdapter 309
Miembros de DB2eDataReader 310
Miembros de DB2eError. 312
Miembros de DB2eException 312
Miembros de DB2eParameter 312
Miembros de DB2eTransaction. 313
Enumeración de DB2eType 314
API C de IBM Sync Client 315
Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2 315
Resumen de funciones de la API C de IBM Sync Client 317
Tipos de datos de la API C de IBM Sync Client 319
Descripciones de funciones de la API C de IBM Sync Client 321
Clave para las descripciones de funciones de la API C de IBM Sync Client 321
iscGetVersion() 321
iscServiceOpen() 322
iscServiceOpenEx() 324
iscServiceClose() 325
iscConfigOpen() 326
iscConfigClose() 327
iscConfigPurge() 328
iscConfigOpenCursor() 328
iscConfigCloseCursor() 330
iscConfigGetNextSubsSet() 331
iscConfigEnableSubsSet() 332
iscConfigDisableSubsSet() 333
iscConfigResetSubsSet() 334
iscConfigSubsSetIsEnabled() 335
iscConfigSubsSetIsReset() 336
iscConfigGetSubsSetStatus() 337
iscEngineOpen() 338
iscEngineClose() 339
iscEngineGetInfo(). 340
iscEngineSetListener() 341
iscEngineListenerPF 342
iscEngineSetPref() 348
iscEngineGetPref(). 350
iscEngineSync(). 351
iscEngineSyncConfig() 352

Capítulo 19. Tablas base del catálogo del sistema DB2 Everyplace 355

Capítulo 20. Límites de DB2 Everyplace . . . 357

Capítulo 21. Palabras reservadas de DB2 Everyplace 359

Capítulo 22. Soporte de idioma nacional (NLS) 361
Soporte NLS por sistema operativo de DB2 Everyplace 361
Codificación de caracteres en aplicaciones Java . . 363
Habilitadores de idioma de DB2 Everyplace . . . 363
Soporte UNICODE en DB2 Everyplace 364

Capítulo 23. El conjunto de información de DB2 Everyplace 367
Archivos PDF y HTML de DB2 Everyplace . . . 367
Documentación en línea sobre DB2 Everyplace . . 368

Capítulo 18. Interfaces de programación de aplicaciones (las API)

Soporte de sentencia SQL de DB2 Everyplace

Este capítulo contiene diagramas de sintaxis soportados, descripciones semánticas, reglas y ejemplos sobre la utilización de las sentencias de SQL soportadas por DB2 Everyplace. Los temas que se tratan son:

- "Visión general del soporte de sentencia de SQL de DB2 Everyplace"
- "Compatibilidad entre tipos de datos para las operaciones de asignación y comparación" en la página 176
- "Marcadores de parámetro soportados de DB2 Everyplace" en la página 76
- "Listado de los SQLSTATE" en la página 180
- "Resumen de códigos de clase de SQLState" en la página 180
- "Mensajes de SQLSTATE notificados por SQL" en la página 181
- "Mensajes de SQLSTATE notificados por CLI" en la página 185

Visión general del soporte de sentencia de SQL de DB2 Everyplace

Las sentencias ejecutables de SQL soportadas se pueden emitir interactivamente desde el dispositivo portátil, utilizando el procesador de línea de mandatos (CLP), o se pueden utilizar en programas de aplicación para acceder a datos de una base de datos de DB2 Everyplace. La Tabla 15 lista las sentencias de SQL que están soportadas por DB2 Everyplace.

Tabla 15. Sentencias de SQL soportadas

sentencia de SQL	Función
CALL	Llama a un procedimiento almacenado remoto utilizando el adaptador de Consulta remota y Procedimiento almacenado (AgentAdapter) de DB2 Everyplace Sync Server.
CREATE INDEX	Crea un índice.
CREATE TABLE	Define una tabla.
DELETE	Suprime una o más filas de una tabla.
DROP	Suprime una tabla o índice de una base de datos.
EXPLAIN	Obtiene información sobre la selección de una vía de acceso para una sentencia SELECT.
GRANT	Otorga privilegios de cifrado a un usuario.
INSERT	Inserta una o más filas en una tabla.
REORG TABLE	Elimina o reduce el espacio de almacenamiento desaprovechado correspondiente a la tabla especificada.
REVOKE	Revoca privilegios de cifrado de un usuario.
SELECT	Especifica una tabla resultante consultada a partir de una o más tablas.
UPDATE	Actualiza los valores de una o más columnas en una o más filas de una tabla.

© Copyright IBM Corp. 1999,2003

La sección “Mensajes de SQLSTATE notificados por SQL” en la página 181 lista todos los SQLSTATE notificados por el motor SQL de DB2 Everyplace.

La longitud de una sentencia de SQL no puede ser mayor que 64.000 caracteres.

El catálogo incluye las tablas del sistema DB2 Everyplace siguientes, que DB2 Everyplace gestiona: DB2eSYSTABLES, DB2eSYSRELS y DB2eSYSCOLUMNS.

Consulta relacionada:

- “Compatibilidad entre tipos de datos para las operaciones de asignación y comparación” en la página 176
- “Marcadores de parámetro soportados de DB2 Everyplace” en la página 76
- “Listado de los SQLSTATE” en la página 180
- “Resumen de códigos de clase de SQLState” en la página 180

CALL

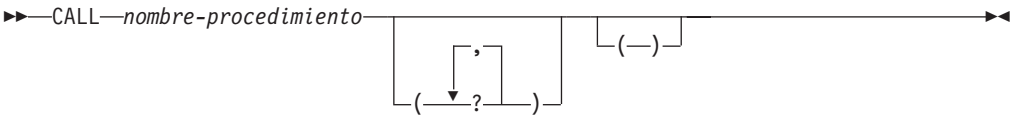
Invoca a un procedimiento almacenado definido con el adaptador de Consulta remota y Procedimiento almacenado para el DB2 Everyplace Sync Server. Por ejemplo, un procedimiento almacenado se ejecuta en la ubicación de la base de datos remota y devuelve los datos a la aplicación cliente de DB2 Everyplace.

Los programas que utilizan la sentencia de SQL CALL están diseñados para que se ejecuten en dos partes, una en el cliente y la otra en el servidor.

Invocación:

Los procedimientos almacenados remotos se invocan desde una aplicación de DB2 Everyplace pasando la sintaxis de sentencia CALL siguiente a `SQLPrepare()`, seguida por `SQLExecute()`.

Sintaxis:



Descripción:

nombre-procedimiento
Identifica el procedimiento que se debe llamar en el servidor remoto. El procedimiento identificado debe estar definido en la suscripción AgentAdapter del Sync Server actual.

? El signo de interrogación *?* en el diagrama de la sintaxis de la sentencia CALL denota un marcador de parámetro que se corresponde a un argumento para un procedimiento almacenado. Todos los argumentos se deben pasar utilizando los marcadores de parámetros.

Reglas:

Ninguna

Notas:

La sentencia CALL utiliza el adaptador de consulta remota y procedimiento almacenado que se incluye con el DB2 Everyplace Sync Server. DB2 Everyplace Sync Server debe utilizar la sentencia CALL en aplicaciones de DB2 Everyplace. DB2 Everyplace no da soporte a procedimientos almacenados locales.

Para obtener información adicional, consulte la sección sobre fuentes de datos en el manual *DB2 Everyplace Sync Server Administration Guide*.

Ejemplo:

Un ejemplo completo sobre cómo utilizar la sentencia CALL y el adaptador de consulta remota y procedimiento almacenado está disponible en el manual *DB2 Everyplace Sync Server Administration Guide*. El ejemplo siguiente sólo muestra el código de la sentencia CALL en una aplicación de ejemplo.

Se define un procedimiento almacenado `MYPROC()` en el servidor fuente para la base de datos `mysample`. Se define una suscripción AgentAdapter en el DB2 Everyplace Sync Server con los atributos siguientes:

```
ID de usuario:  db2admin
Contraseña:  db2admin
Otros:  dbname=mysample;procname= db2e.MYPROC
```

Programa de ejemplo utilizando la sentencia CALL:

```
int main(int argc, char * argv[])
{
  SQLHENV  henv;
  SQLHDBC  hdbc;
  SQLHSTMT hstmt;
  SQLRETURN rc;
 SQLCHAR  strSQL[] = "CALL db2e.MYPROC(?,?,?,?,?)";
 int   nInd4, nInd5;
 int   nSaving = 0, nChecking =0 ;
 int   nCmd =0, nAmount=0;
  SQLCHAR  strConnect[254];

 //*************************************************************
 //* Check input parameters
 //*************************************************************
 if ( argc < 4 ){
  printf("\nUsage : myClient AccountName Cmd Amount");
  printf("\n     cmd 1 : query balance");
  printf("\n     cmd 2 : Transfer from Saving to Checking");
  printf("\n     cmd 3 : Transfer from Checking to Saving");
  return (99);
 }
 nCmd = atoi(argv[2]);
 nAmount = atoi(argv[3]);

 //*************************************************************
 //* Allocate handles
 //*************************************************************
 rc = SQLAllocHandle( SQL_HANDLE_ENV,
      SQL_NULL_HANDLE,
      &henv; //checkerror
 rc = SQLAllocHandle( SQL_HANDLE_DBC,
      henv,
      &hdbc); //checkerror
 if (argc == 5){
  strcpy(strConnect,"http://");
  strcat(strConnect,argv[4]);
  strcat(strConnect,"/servlet/com.ibm.mobileservices.adapter.agent.AgentServlet?DB=mysample");
 }else{
  strcpy(strConnect,
             "http://127.0.0.1:8080/db2e/servlet/

              com.ibm.mobileservices.adapter.agent.AgentServlet?DB=mysample");
 }

 //*************************************************************
 //* Connect to remote database
 //*************************************************************
 rc = SQLConnect(hdbc,
```

```
 strConnect,
 SQL_NTS,
 "userex", SQL_NTS,
 "userex", SQL_NTS  ); //checkerror
rc = SQLAllocHandle( SQL_HANDLE_STMT,
     hdbc,
     &hstmt); //checkerror
//*************************************************************
//* Prepare, Bind , and Execute the statement
//*************************************************************
rc = SQLPrepare(hstmt,strSQL, SQL_NTS); //checkerror
rc = SQLBindParameter(hstmt,
  1,
  SQL_PARAM_INPUT,
  SQL_C_CHAR,
  SQL_CHAR,
  0,
  0,
  (SQLPOINTER)argv[1],
  0,
  NULL ); //checkerror
rc = SQLBindParameter(hstmt,
  2,
  SQL_PARAM_INPUT,
  SQL_C_LONG,
  SQL_INTEGER,
  0,
  0,
  (SQLPOINTER)&nCmd,
  sizeof(int),
  NULL); //checkerror
rc = SQLBindParameter(hstmt,
  3,
  SQL_PARAM_INPUT,
  SQL_C_LONG,
  SQL_INTEGER,
  0,
  0,
  (SQLPOINTER)&nAmount,
  sizeof(int),
  NULL ); //checkerror
rc = SQLBindParameter(hstmt,
  4,
  SQL_PARAM_OUTPUT,
  SQL_C_LONG,
  SQL_INTEGER,
  0,
  0,
  (SQLPOINTER)&nSaving,
  sizeof(int),
  &nInd4 ); //checkerror
rc = SQLBindParameter(hstmt,
  5,
  SQL_PARAM_OUTPUT,
  SQL_C_LONG,
  SQL_INTEGER,
  0,
  0,
  (SQLPOINTER)&nChecking,
  sizeof(int),
  &nInd5 ); //checkerror
rc = SQLExecute(hstmt); //checkerror
//*************************************************************
//* Print the balance
//*************************************************************
printf("\nSaving = %d",nSaving);
printf("\nChecking = %d",nChecking);

SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
SQLDisconnect(hdbc);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
return 0;
```

Consulta relacionada:

- “Visión general del soporte de sentencia de SQL de DB2 Everyplace” en la página 137
- “Compatibilidad entre tipos de datos para las operaciones de asignación y comparación” en la página 176
- “Marcadores de parámetro soportados de DB2 Everyplace” en la página 76

- "Listado de los SQLSTATE" en la página 180
- "Resumen de códigos de clase de SQLState" en la página 180

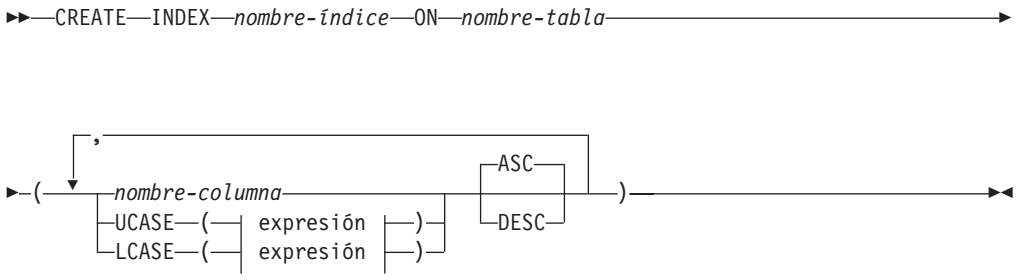
CREATE INDEX

La sentencia CREATE INDEX se utiliza para crear un índice sobre una tabla de DB2 Everyplace.

Invocación:

Esta sentencia puede utilizarse en un programa de aplicación utilizando las funciones de CLI de DB2 o emitirse a través del CLP.

Sintaxis:



Descripción:

INDEX *nombre-índice*
Designa el índice.

ON *nombre-tabla*
El parámetro *nombre-tabla* designa una tabla en la que debe crearse un índice.

nombre-columna
Para un índice, el nombre de columna identifica una columna que debe formar parte de la clave de índice.

Cada nombre de columna debe ser un nombre no calificado que identifique una columna de la tabla. Utilice ocho columnas o menos; los nombres de columna no se pueden repetir (SQLSTATE 42711).

La longitud de cada una de las columnas especificadas no puede ser mayor que 1024 bytes.

ASC Ordena las entradas de índice en orden ascendente para cada columna. Es el valor por omisión.

DESC Ordena las entradas de índice en orden descendente para cada columna.

LCASE / UCASE

La función LCASE o LOWER devuelve una serie en la que todos los caracteres SBCS se han convertido a caracteres en minúsculas. Es decir, los caracteres A-Z se convertirán en los caracteres a-z y los caracteres con marcas diacríticas se convertirán a sus equivalentes en minúsculas en el caso de que éstas existan.

El argumento debe ser una expresión cuyo valor sea un tipo de datos CHAR o VARCHAR.

El resultado de la función tiene el mismo tipo de datos y atributo de longitud que el argumento. Si el argumento puede ser nulo, el resultado también; si el argumento es nulo, el resultado es el valor nulo.

Asegúrese de que los caracteres del valor de la columna JOB de la tabla EMPLOYEE se devuelven en caracteres en minúsculas. Por ejemplo:

```
SELECT LCASE(JOB)
     FROM EMPLOYEE
     WHERE EMPNO = '000020';
```

Reglas:

- Se puede crear un máximo de 15 índices en una tabla sin una clave primaria. Se puede crear un máximo de 14 índices en una tabla con una clave primaria.
- La sentencia CREATE INDEX fallará si se intenta crear un índice que coincida con un índice existente. Se considera que dos descripciones de índice son iguales si se cumplen estas dos condiciones:
  - El conjunto de columnas y su orden en el índice es el mismo que el de un índice existente.
  - Los atributos de ordenación son iguales.
- Las columnas que contienen datos de tipo BLOB no se pueden utilizar en una sentencia CREATE INDEX.

Notas:

- La sentencia CREATE INDEX puede contener un máximo de 8 columnas.
- DB2 Everyplace da soporte a la exploración bidireccional de índices. Los dos índices siguientes tienen el mismo propósito aunque tienen definiciones diferentes.

  ```
  CREATE INDEX IDX1 ON EMPLOYEE (JOB ASC)
  CREATE INDEX IDX1 ON EMPLOYEE (JOB DESC)
  ```

  En general, los índices se deberían crear sin especificar la dirección del orden. Normalmente, tener menos índices viene a significar unos costes de mantenimiento de índices menor.
- DB2 Everyplace da soporte a la exploración de prefijos de los índices. Considere el siguiente ejemplo. Se crea el índice siguiente.

  ```
  CREATE INDEX J1 ON T (A, B, C, D, E, F, G, K)
  ```

  No es necesario crear otro índice en `T (A,B,C,D)`.
- Si la tabla no contiene datos, CREATE INDEX crea una descripción del índice; las entradas de índice se crean al insertar datos en la tabla.
- Para crear un índice para el bit de modificación, siga este ejemplo:

  ```
  CREATE INDEX <nombre de índice>
    ON <nombre de tabla>
    ($dirty)
  ```

  Consulte 278 para obtener más información sobre el bit de modificación.

Ejemplo:

Cree un índice llamado JOB_BY_DPT en la tabla EMPLOYEE. Las entradas de índice se disponen en orden ascendente para cada puesto de trabajo (JOB) dentro de cada departamento (WORKDEPT).

```
CREATE INDEX JOB_BY_DPT
ON EMPLOYEE (WORKDEPT, JOB)
```

Consulta relacionada:

- "Visión general del soporte de sentencia de SQL de DB2 Everyplace" en la página 137
- "Compatibilidad entre tipos de datos para las operaciones de asignación y comparación" en la página 176
- "Marcadores de parámetro soportados de DB2 Everyplace" en la página 76
- "Listado de los SQLSTATE" en la página 180
- "Resumen de códigos de clase de SQLState" en la página 180

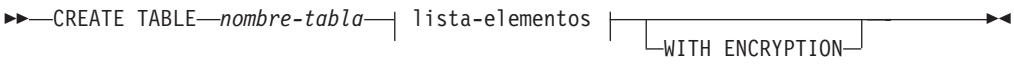
CREATE TABLE

La sentencia CREATE TABLE define una tabla. La definición debe incluir el nombre de la tabla y los nombres y atributos de sus columnas. La definición puede también incluir otros atributos de la tabla, tales como su clave primaria.

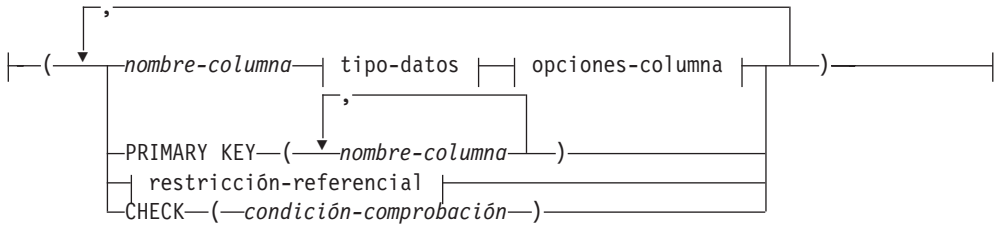
Invocación:

Esta sentencia puede utilizarse en un programa de aplicación utilizando las funciones de CLI de DB2 o emitirse a través del CLP.

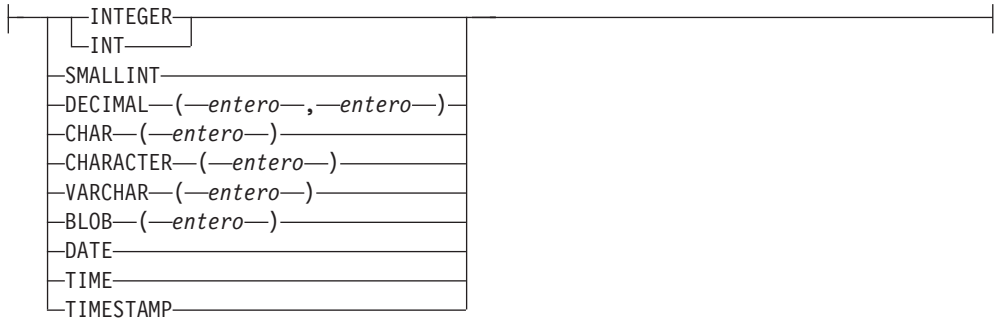
Sintaxis:



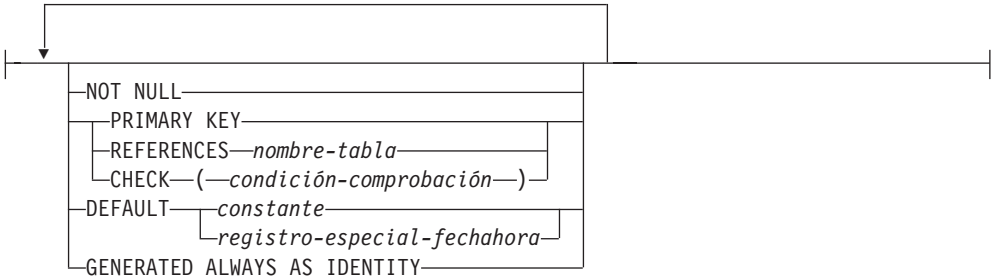
lista-elementos:



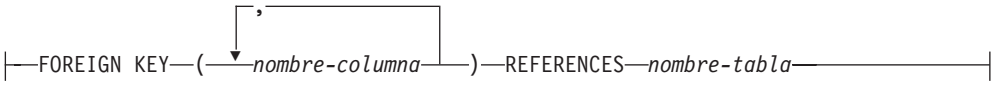
tipo-datos:



opciones-columna:



restricción-referencial:



Descripción:

nombre-tabla
Designa la tabla. El nombre puede tener hasta 18 caracteres de longitud. El nombre no debe identificar una tabla contenida en el catálogo. El nombre debe ser exclusivo para el dispositivo portátil.

Los nombres de tabla se convierten a mayúsculas antes de almacenarse en el catálogo. Puede utilizar identificadores delimitados (con comillas dobles) para impedir esa conversión. Debe utilizar identificadores delimitados cuando un nombre de tabla contenga espacios en blanco o caracteres especiales.

Los nombres de tabla pueden incluir caracteres DBCS (de doble byte).

Restricción: Los archivos de datos creados por el sistema que corresponden a tablas creadas y denominadas por nombres de usuario no distinguen entre caracteres en mayúsculas y minúsculas. Por ejemplo, el archivo de datos para una tabla denominada TB se denomina DSY_TB. El archivo de datos para una tabla denominada ″tb″ también es DSY_TB. Por consiguiente le recomendamos firmemente que, para asegurarse de la integridad de los datos, no denomine una tabla utilizando una serie de caracteres idéntica, a excepción de la consideración sobre mayúsculas y minúsculas, a la de un nombre de tabla existente.

WITH ENCRYPTION
Crea una tabla de usuario cifrada. Para cifrar una tabla, el usuario tiene que estar autentificado y conectado. Se le tiene que otorgar explícitamente el privilegio de cifrado. (Para obtener más información, consulte el apartado “GRANT” en la página 156.)

Una tabla de usuario únicamente se puede cifrar en el momento de su creación. Una vez que se ha creado, no se puede añadir ni eliminar cifrado, si no es suprimiendo la tabla.

nombre-columna
Designa una columna de la tabla. El nombre puede tener hasta 18 caracteres de longitud. El nombre no puede estar calificado y no puede utilizarse un mismo nombre para más de una columna de la tabla.

Los nombres de columna se convierten a mayúsculas antes de almacenarse en el catálogo. Puede utilizar identificadores delimitados (con comillas dobles)

para impedir esa conversión. También debe utilizar identificadores delimitados cuando un nombre de columna contenga espacios en blanco o caracteres especiales.

Los nombres de columna pueden incluir caracteres DBCS.

tipo-datos
Es uno de los tipos mostrados en la lista siguiente. Utilice:

INTEGER o **INT**
Para un entero de cuatro bytes, con signo, comprendido entre 2147483647 y -2147483648.

SMALLINT
Para un entero de dos bytes, con signo, comprendido entre -32768 y 32767.

DECIMAL*(entero-precisión, entero-escala)*
Para un número decimal. El primer entero indica la precisión del número; es decir, el número total de dígitos; su valor está comprendido entre 1 y 31. El segundo entero es la escala del número; es decir, el número de dígitos situados a la derecha de la coma decimal; su valor está comprendido entre 0 y la precisión del número.

CHAR*(entero)*
Para una serie de caracteres de longitud fija, cuya longitud es *entero* y puede oscilar entre 1 y 32767.

CHARACTER*(entero)*
Para una serie de caracteres de longitud fija, cuya longitud es *entero* y puede oscilar entre 1 y 32767.

VARCHAR*(entero)*
Para una serie de caracteres de longitud variable, cuya longitud máxima es *entero* y puede oscilar entre 1 y 32767.

BLOB*(entero)*
Para un gran objeto binario de tipo serie, cuya longitud máxima en bytes es la especificada.

La longitud está comprendida entre 1 y 32767 bytes.

entero es la longitud máxima.

DATE
Para una fecha. Un valor de entrada puede tener uno de estos formatos: MM/DD/AAAA, AAAA-MM-DD o MM.DD.AAAA. El valor de fecha se imprime sólo en el formato ISO: AAAA-MM-DD.

El registro especial CURRENT DATE también genera la fecha actual en el formato ISO.

TIME
Para una hora. Es un valor de entrada que puede tener uno de los formatos siguientes: HH:MM am (o pm), HH:MM:SS, HH.MM am (o pm), o HH.MM.SS. Los SS, segundos, son opcionales con los formatos HH:MM:SS o HH.MM.SS. El valor de hora sólo se imprime en el formato ISO: HH:MM:SS.

El registro especial CURRENT TIME también genera la hora actual en el formato ISO.

TIMESTAMP
Para una indicación de hora. Un valor de entrada debe tener el siguiente

formato: AAAA-MM-DD-HH.MM.SS.ZZZZZZ. Un valor de indicación de hora se imprime con el siguiente formato: AAAA-MM-DD-HH.MM.SS.ZZZZZZ.

El registro especial CURRENT TIMESTAMP también genera la indicación de hora actual.

opciones-columna
Define opciones adicionales referentes a las columnas de la tabla.

NOT NULL
Impide que la columna contenga valores nulos.

Si no se especifica NOT NULL, la columna puede contener valores nulos, y su valor por omisión es el valor nulo o el valor proporcionado por la cláusula DEFAULT.

PRIMARY KEY
Proporciona un atajo para definir una clave primaria formada por una sola columna. Por tanto, si se especifica PRIMARY KEY en la definición de la columna C, el efecto es el mismo que si se especifica PRIMARY KEY(C) como cláusula separada.

Vea la descripción de PRIMARY KEY en la página 147.

REFERENCES *nombre-tabla*
Vea la descripción de REFERENCES en la página 147.

CHECK (*condición-comprobación***)**
Vea la descripción de CHECK en la página 148.

DEFAULT
Proporciona un valor por omisión cuando no se especifica un valor en una sentencia INSERT.

Si no se especifica DEFAULT en una definición de columna, se utiliza el valor nulo como valor por omisión de la columna. Si dicha columna se define como NOT NULL, no tendrá un valor por omisión válido.

constante
Especifica la constante como valor por omisión de la columna. La constante especificada debe:

- Representar un valor que se pueda asignar a la columna.
- No tener dígitos distintos de cero que sobrepasen la escala del tipo de datos de la columna si la constante es una constante decimal (por ejemplo, 1,234 no puede ser el valor por omisión de una columna DECIMAL(5,2)).

registro-especial-fechahora
Especifica como valor por omisión de la columna el valor que tiene el registro especial de fecha-hora (CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP) cuando se ejecuta INSERT. El tipo de datos de la columna debe corresponder al registro especial especificado (por ejemplo, el tipo de datos debe ser DATE cuando se especifica CURRENT DATE).

GENERATED ALWAYS AS IDENTITY
Al crear una tabla, un usuario puede especificar una columna como "GENERATED ALWAYS AS IDENTITY". Posteriormente, el valor de esta columna lo generará DB2 Everyplace cada vez que el usuario realiza una operación INSERT o INSERT con sub–SELECT. Esta columna tiene que ser de tipo numérico, (tipo INTEGER, SMALLINT o DECIMAL) y DB2

Everyplace genera automáticamente números serie exclusivos comenzando por 1 y aumentando dicho valor en 1 cada vez.

El valor generado para la columna IDENTITY comienza en 1 y aumenta de 1 en 1 cada vez que una fila se inserta en la tabla. De este modo, se garantiza la exclusividad, aunque DB2 Everyplace no crea automáticamente un índice en una columna IDENTITY. Si desea disponer de un índice en una columna IDENTITY, deberá o crear un índice explícitamente, o especificar la columna como PRIMARY KEY. Cuando se agota el rango de los valores de una columna IDENTITY (se llega al valor máximo), las sentencias INSERT adicionales ocasionarán un error (SQLSTATE 23522). El valor máximo de una columna IDENTITY de los tipos INT y SMALLINT son los valores máximos que admitan esos 2 tipos. El valor máximo de una columna IDENTITY de un tipo DECIMAL se determina mediante (1) la definición del tipo de datos (precisión, escala) y (2) el valor máximo que se permite para la columna IDENTITY: 2.15* (10∧18) (19 dígitos decimales). El valor más pequeño para el (1) y (2) es el límite de rango. Para una columna IDENTITY de un tipo DECIMAL, la parte fraccionaria es siempre 0 y la parte entera aumenta de 1 en 1 cada vez.

La especificación IDENTITY sólo puede definirse en las columnas cuyo tipo de datos es uno de los 3 tipos numéricos: INT, SMAIINT, DECIMAL. En caso contrario, se suscita un error (SQLSTATE 42815). Como mínimo puede haber una columna IDENTITY por tabla (en caso contrario el error SQLSTATE 428C1). El usuario no puede proporcionar un valor para una columna IDENTITY en una sentencia INSERT (debe tomar por omisión el valor generado por el sistema de DB2 Everyplace), ni tampoco actualizar (UPDATE) una columna IDENTITY.

PRIMARY KEY (*nombre-columna***, ...)**
Define una clave primaria formada por las columnas identificadas. Esta cláusula no se puede especificar más de una vez y las columnas indicadas no deben estar definidas como NOT NULL. Cada nombre de columna debe identificar una columna de la tabla y una misma columna no se debe identificar más de una vez.

El número de columnas especificadas no debe ser mayor que 8.

Se creará un índice exclusivo de modo automático en las columnas especificadas.

Sólo se puede definir una sola clave primaria para una tabla.

El atributo de longitud de cada una de las columnas especificadas no puede ser mayor que 1024 bytes.

restricción-referencial
Define una restricción referencial.

FOREIGN KEY (*nombre-columna***, ...)**
Define una restricción referencial con el nombre de restricción especificado.

Supongamos que T1 denota la tabla objeto de la sentencia. La clave foránea de la restricción referencial está formada por las columnas identificadas. Cada nombre de la lista de nombres de columna debe identificar una columna de T1 y una misma columna no se debe identificar más de una vez. El número de columnas especificadas no debe ser mayor que 8. DB2 Everyplace no da soporte a las claves foráneas.

REFERENCES *nombre-tabla*

La tabla especificada en la cláusula REFERENCES debe identificar una tabla base que esté descrita en el catálogo, pero no debe denotar una tabla del catálogo.

Una restricción referencial es un duplicado si su clave foránea es igual que la tabla de clave foránea de una restricción referencial especificada anteriormente.

En la explicación siguiente, supongamos que T2 representa la tabla padre identificada y T1 representa la tabla que se está creando.

La clave foránea debe tener el mismo número de columnas que la clave padre de T2 y la descripción de la columna enésima de la clave foránea debe ser comparable con la descripción de la columna enésima de esa clave padre. Las columnas de fecha-hora no se consideran comparables con las columnas de tipo serie para los efectos de esta regla. DB2 Everyplace no da soporte a las claves foráneas.

CHECK (*condición-comprobación***)**

Define una restricción de comprobación. Una *condición-comprobación* es una condición de búsqueda. Una referencia de columna debe ser una columna de la tabla que se está creando. Los valores que se insertan o actualizan en una tabla deben cumplir las restricciones de comprobación que existan.

Si una restricción de comprobación se especifica como parte de una definición de columna, la referencia de columna sólo puede hacerse a la misma columna. Las restricciones de comprobación especificadas como parte de una definición de tabla pueden tener referencias que identifican columnas definidas previamente en la sentencia CREATE TABLE. No se verifica si las restricciones de comprobación contienen incoherencias, condiciones duplicadas o condiciones equivalentes. Por tanto, se pueden definir restricciones de comprobación contradictorias o redundantes.

Aunque se puede especificar la condición de comprobación ″IS NOT NULL″, es recomendable utilizar el atributo NOT NULL para habilitar directamente la capacidad de una columna para contener nulos. Por ejemplo, CHECK (salario + prima > 30000) es aceptable si salario = NULL, pues las restricciones de comprobación deben ser ciertas o desconocidas, y en este caso el salario es un valor desconocido. En cambio, CHECK (salario IS NOT NULL) sería una condición falsa y vulneraría la restricción si salario = NULL.

Las restricciones de comprobación se aplican cuando se actualizan o insertan filas en una tabla.

Todas las restricciones de comprobación definidas en una sentencia CREATE TABLE se combinan y almacenan en el catálogo del sistema. En DB2 Everyplace, esta restricción de comprobación combinada puede tener un tamaño máximo de 512 bytes.

Reglas:

- El total real del número de bytes de una fila no debe ser mayor que 65 536.

  Consulte el apartado 148 para obtener más información.
- Las columnas cuyo tipo de datos es BLOB no pueden tener restricciones de comprobación, predefinidas ni de clave foránea (SQLSTATE 42962).
- Las columnas que contienen datos de tipo BLOB no se pueden utilizar en la clave primaria de una sentencia CREATE TABLE.

Notas:

- Las tablas y las columnas se deben crear utilizando nombres en mayúsculas. Los nombres que mezclan mayúsculas y minúsculas pueden ocasionar errores en algunos lenguajes.
- Si crea una nueva tabla en su dispositivo portátil, la tabla no se creará automáticamente en la base de datos corporativa al sincronizar el dispositivo portátil con el servidor. Es necesario crear la tabla en la base de datos corporativa antes de realizar la sincronización.
- Números de bytes de datos: La lista siguiente contiene los números de bytes de las columnas de acuerdo con su tipo de datos.Esto puede cambiar en cada release. Cada registro también incluye información sobre los valores NULL. La información NULL requiere 4 bytes para cada grupo de 32 columnas. Un valor NULL sigue utilizando el tamaño fijo de la columna.

Tipo de datos	**Número de bytes de la columna**
INTEGER	4
SMALLINT	4
DECIMAL(n, m)	4 – 20
CHAR(n)	n+1
VARCHAR(n)	i+5 donde i es la longitud real
BLOB	i+4 donde i es la longitud real
DATE	4
TIME	4
TIMESTAMP	12

Ejemplo:

En este ejemplo se crea una tabla EMPLOYEE con los nombres de columna EMPNO, FIRSTNAME, LASTNAME, DEPT, PHONENO, SALARY y HIREDATE. CHAR significa que la columna contendrá datos de tipo carácter. NOT NULL significa que la columna no puede contener un valor nulo. VARCHAR significa que la columna contendrá datos de tipo carácter de longitud variable. La clave primaria está formada por la columna EMPNO.

```
CREATE TABLE EMPLOYEE
  (EMPNO       CHAR(3)    PRIMARY KEY,
   FIRSTNAME   VARCHAR(12)   NOT NULL,
   LASTNAME    VARCHAR(15)   NOT NULL,
   DEPT        CHAR(3),
   PHONENO     CHAR(4),
   SALARY      INT,
   HIREDATE    DATE)
```

Consulta relacionada:

- “Visión general del soporte de sentencia de SQL de DB2 Everyplace” en la página 137
- “Compatibilidad entre tipos de datos para las operaciones de asignación y comparación” en la página 176
- “Marcadores de parámetro soportados de DB2 Everyplace” en la página 76
- “Listado de los SQLSTATE” en la página 180
- “Resumen de códigos de clase de SQLState” en la página 180

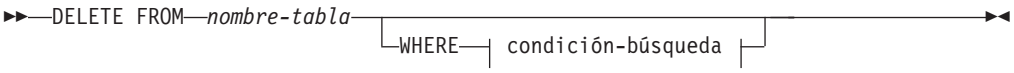
DELETE

La sentencia DELETE suprime una o más filas de una tabla.

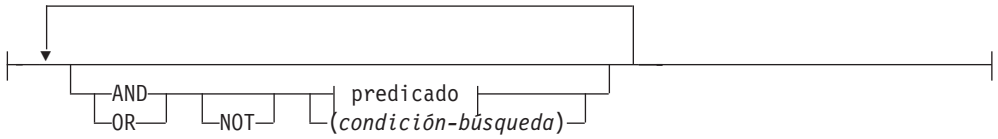
Invocación:

Esta sentencia puede utilizarse en un programa de aplicación utilizando las funciones de CLI de DB2 o emitirse a través del CLP.

Sintaxis:

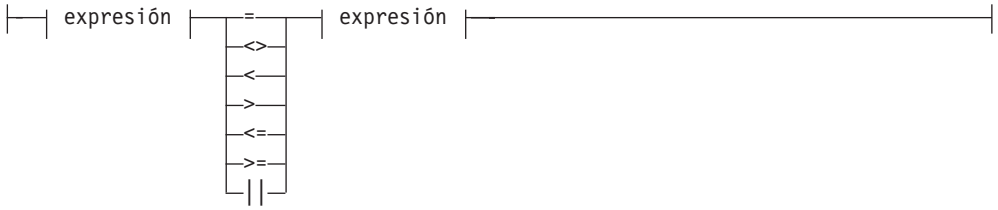


condición-búsqueda:

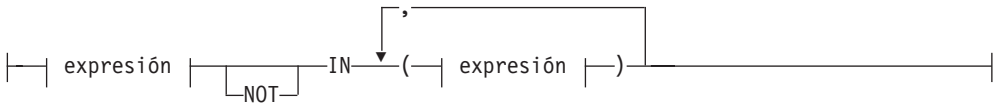


predicado:

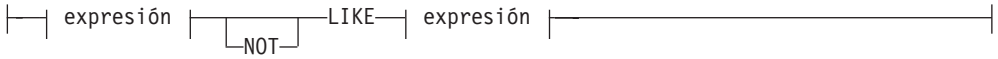
predicado básico:



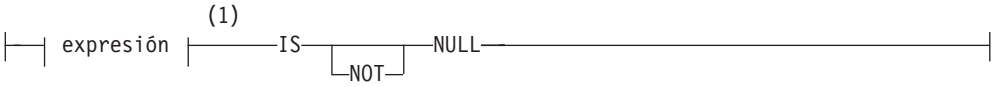
predicado IN:



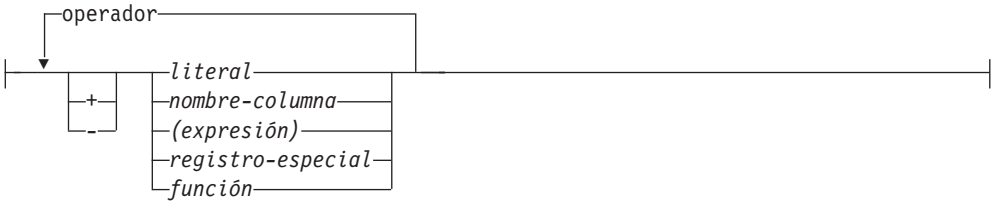
predicado LIKE:



predicado NULL:



expresión:



operador:

Notas:

1 Las expresiones BLOB sólo se permiten en predicados NULL.

Descripción:

FROM *nombre-tabla*
Designa la tabla en la que deben suprimirse filas. El nombre debe especificar una tabla existente en el catálogo, pero no una tabla de catálogo.

WHERE
Especifica una condición que selecciona las filas que deben suprimirse. Se puede omitir la cláusula o especificar una condición de búsqueda. Si se omite la cláusula, se suprimen todas las filas de la tabla.

condición-búsqueda
La *condición de búsqueda* especifica una condición que es verdadera, falsa o desconocida para una fila determinada.

El resultado de una *condición de búsqueda* se obtiene aplicando los *operadores lógicos* especificados (AND, OR, NOT) al resultado de cada predicado especificado. Un predicado compara dos valores. Si no se especifican operadores lógicos, el resultado de la condición de búsqueda es el resultado del predicado especificado.

Las condiciones de búsqueda que están entre paréntesis se evalúan en primer lugar. Si no se especifica un orden de evaluación mediante el uso de paréntesis, NOT se aplica antes que AND y AND se aplica antes que OR. El orden en el que se evalúan los operadores con igual nivel de prioridad es indefinido, para permitir la optimización de las condiciones de búsqueda.

La *condición de búsqueda* se aplica cada fila de la tabla y las filas suprimidas son aquéllas para las cuales el resultado de la *condición de búsqueda* es verdadero.

Cada *nombre de columna* especificado en la condición de búsqueda debe identificar una columna de la tabla.

NOT
Si se especifica NOT, se invierte el resultado del predicado.

expresión
Identifica un operando del predicado. La *expresión* puede ser un literal, nombre de columna, registro especial o función.

No se da soporte a las operaciones aritméticas sobre los tipos de datos BLOB(n), DATE, TIME y TIMESTAMP.

literal
Un *literal* puede ser un valor cuyo tipo de datos es INTEGER, SMALLINT, DECIMAL, CHAR(n), VARCHAR(n), BLOB(n), DATE, TIME o TIMESTAMP.

nombre-columna
Identifica la columna que es un operando del predicado.

registro-especial
Identifica el registro especial que es un operando del predicado. Se pueden utilizar los registros especiales CURRENT DATE, CURRENT TIME y CURRENT TIMESTAMP para generar la fecha, la hora o la indicación de hora actuales.

función
Puede incluir sólo las funciones MOD, LENGTH y RTRIM.

operador relacional
Puede ser cualquiera de los operadores siguientes:

= Igual a.

<> No igual a.

< Menor que.

> Mayor que.

<= Menor o igual que.

>= Mayor o igual que.

LIKE Coincide con una serie de caracteres. Utilice un símbolo de subrayado de SBCS (juego de caracteres de un solo byte) para representar un carácter SBCS individual. Utilice un símbolo de subrayado de DBCS (juego de caracteres de doble byte) para representar un carácter DBCS individual. Por ejemplo, la condición `WHERE PART_NUMBER LIKE '_0'` obtiene todos los números de pieza de 2 dígitos que terminan en 0 (20, 30 y 40, por ejemplo). Utilice un símbolo de porcentaje (del juego de caracteres SBCS o DBCS) para representar una cadena de caracteres SBCS o DBCS, o la ausencia de caracteres. Por ejemplo, la condición `WHERE DEPT_NUMBER LIKE '2%'` obtiene todos los números de departamento (DEPT_NUMBER) que comienzan con el número 2 (por ejemplo, 20, 27 ó 234).

NOT LIKE
Indica que al menos uno de los caracteres es diferente.

IS NULL
Contiene el valor nulo.

IS NOT NULL
No contiene el valor nulo.

AND
Si se especifica, el operador lógico AND se aplica al resultado de cada predicado especificado.

OR
Si se especifica, el operador lógico OR se aplica al resultado de cada predicado especificado.

Reglas:

Ninguna.

Notas:

- Un DELETE lógico no se aplica nunca a registros suprimidos lógicamente.

Ejemplo:

Supresión del número de empleado (EMPNO) 003002 de la tabla EMPLOYEE:

```
DELETE FROM EMPLOYEE
  WHERE EMPNO = '003002'
```

Consulta relacionada:

- "Visión general del soporte de sentencia de SQL de DB2 Everyplace" en la página 137
- "Compatibilidad entre tipos de datos para las operaciones de asignación y comparación" en la página 176
- "Marcadores de parámetro soportados de DB2 Everyplace" en la página 76
- "Listado de los SQLSTATE" en la página 180
- "Resumen de códigos de clase de SQLState" en la página 180

DROP

La sentencia DROP suprime una tabla o índice.

Invocación:

Esta sentencia puede utilizarse en un programa de aplicación utilizando las funciones de CLI de DB2 o emitirse a través del CLP.

Sintaxis:

Descripción:

TABLE *nombre-tabla*
Designa la tabla base que se debe eliminar. *nombre-tabla* debe identificar una tabla que esté descrita en el catálogo (SQLSTATE 42704).

INDEX *nombre-índice*
Designa el índice que se debe eliminar. *nombre-índice* debe identificar un índice que esté descrito en el catálogo (SQLSTATE 42704). No puede ser un índice que el sistema necesite para una clave primaria (SQLSTATE 42704).

Reglas:

Ninguna.

Notas:

- No se deben eliminar tablas ni índices mientras se está utilizando una tabla (existe un descriptor de sentencia activo sobre una consulta que hace uso de esa tabla o índice). La eliminación de tablas e índices que están en uso invalidará los descriptores de sentencia que impliquen la tabla o el índice.

Ejemplo:

Eliminación de la tabla EMPLOYEE:

```
DROP TABLE EMPLOYEE
```

Consulta relacionada:

- "Visión general del soporte de sentencia de SQL de DB2 Everyplace" en la página 137
- "Compatibilidad entre tipos de datos para las operaciones de asignación y comparación" en la página 176
- "Marcadores de parámetro soportados de DB2 Everyplace" en la página 76
- "Listado de los SQLSTATE" en la página 180
- "Resumen de códigos de clase de SQLState" en la página 180

EXPLAIN

La sentencia EXPLAIN obtiene información sobre la selección de una vía de acceso para una sentencia SELECT. La información obtenida se coloca en una tabla de usuario llamada DB2ePLANTABLE.

La sentencia EXPLAIN se puede utilizar en los sistemas operativos siguientes:

- Win32 (Windows 95, Windows 98, Windows NT, Windows 2000 y Windows XP)
- Linux

Invocación:

Esta sentencia puede utilizarse en un programa de aplicación utilizando las funciones de CLI de DB2 o emitirse a través del CLP.

Sintaxis:

Descripción:

SET QUERYNO = entero
Asocia *entero* a la sentencia SELECT. Se asigna el valor *entero* a la columna QUERYNO en cada fila insertada por la sentencia EXPLAIN en la tabla PLAN.

Sentencia SELECT
Especifica un conjunto de columnas nuevas en el formato de la tabla resultante de una sentencia SELECT.

Reglas:

El valor *entero* debe ser positivo.

Notas:

- Cuando utiliza la sentencia EXPLAIN, se crea automáticamente por omisión la tabla DB2ePLANTABLE, si no existe.
- Para crear explícitamente la tabla DB2ePLANTABLE, siga este ejemplo:

```
create table "DB2ePLANTABLE"
 (query_no int, plan_no int, table_name char(18), index_name char(18), sort_temp char(1),
   expl_timestamp timestamp, remarks varchar(300))
```

La Tabla 16 describe las columnas de DB2ePLANTABLE.

Tabla 16. Información sobre las columnas de DB2ePLANTABLE

Nombre de la columna	Descripción
query_no	Número entero que asocia la sentencia EXPLAIN con los datos de salida dentro de DB2ePLANTABLE.
plan_no	Número entero que representa los pasos en los que se ejecuta la sentencia (en orden ascendente).
table_name	El nombre de la tabla o nombre correlacionado que identifica unívocamente la tabla, o un valor nulo si no es aplicable un nombre.
index_name	El nombre del índice (si se utiliza) para acceder a la tabla. Devuelve un valor nulo si no se utiliza ningún índice.
sort_temp	'Y' significa que es necesario realizar una clasificación en una tabla temporal para procesar una sentencia GROUP BY o ORDER BY. Si se devuelve un valor nulo, significa que no es necesaria ninguna tabla temporal de clasificación.
expl_timestamp	Indicación horaria del momento en que se ejecuta la sentencia EXPLAIN.
comentarios	La columna de comentarios contiene el valor nulo. El usuario puede añadir comentarios a esta columna con fines contables.

- DB2ePLANTABLE es una tabla de usuario que puede ser modificada o eliminada por cualquier aplicación.

Ejemplo:

Al desarrollar una nueva aplicación, es aconsejable determinar qué vía de acceso se elige para una sentencia SELECT. En este ejemplo, una nueva aplicación consulta las tablas SALES y EMPLOYEES. La sentencia EXPLAIN muestra si se han elegido los índices apropiados para la sentencia SELECT.

```
 EXPLAIN SET QUERYNO = 100 FOR
  SELECT E.EMPNAME, S.SALES_AMOUNT
      FROM SALES S, EMPLOYEES E
            WHERE S.EMPNO = E.EMPNO
                  AND S.MONTH = ?

Index XSALES on SALES(MONTH)
Index XEMP on EMPLOYEES(EMPNO)

SELECT QUERY_NO, PLAN_NO, TABLE_NAME, INDEX_NAME,  SORT_TEMP
   FROM "DB2ePLANTABLE"

QUERY_NO  PLAN_NO  TABLE_NAME  INDEX_NAME  SORT_TEMP
--------------------------------------------------------------------------------
   100     1        SALES        XSALES        -
   100     2        EMPLOYEE     XEMP          -
```

Consulta relacionada:

- "Visión general del soporte de sentencia de SQL de DB2 Everyplace" en la página 137
- "Compatibilidad entre tipos de datos para las operaciones de asignación y comparación" en la página 176
- "Marcadores de parámetro soportados de DB2 Everyplace" en la página 76
- "Listado de los SQLSTATE" en la página 180
- "Resumen de códigos de clase de SQLState" en la página 180

GRANT

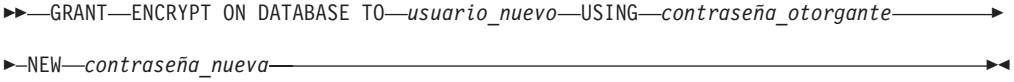
La sentencia GRANT proporciona permiso para crear, consultar y manipular tablas cifradas dentro de la base de datos. Para realizar la operación GRANT, el usuario debe estar conectado y autentificado. Si una base de datos no está cifrada, el primer usuario de la misma puede otorgarse a sí mismo la autentificación necesaria para realizar la operación GRANT. (Para obtener más información sobre cómo hacerlo, vea el ejemplo 1 siguiente.)

Para cambiar su propia contraseña, debe realizar una operación GRANT sobre su propio ID de usuario.

Invocación:

Esta sentencia puede utilizarse en un programa de aplicación utilizando las funciones de CLI de DB2 o emitirse a través del CLP.

Sintaxis:



Descripción:

usuario_nuevo
Identifica el usuario al que se le otorgan los privilegios de cifrado.

contraseña_otorgante
Contraseña del usuario autentificado al que se están otorgando los privilegios de cifrado del usuario nuevo.

contraseña_nueva
Contraseña del usuario al que se le otorgan los privilegios de cifrado.

Reglas:

- Los parámetros tanto de nombre de usuario como de contraseña tienen una longitud limitada de 254 bytes.
- Para caracteres de byte múltiple, internamente se utiliza la codificación UTF-8 para el almacenamiento. Por consiguiente, los nombres de usuario escritos utilizando juegos de caracteres internacionales tienen una longitud limitada.
- DB2 Everyplace requiere que el otorgante (es decir, el usuario conectado en ese momento) vuelva a entrar la contraseña de otorgante para poder otorgar privilegios al usuario nuevo. Estas restricciones aseguran que el otorgante esté presente físicamente en el dispositivo.
- Las contraseñas e idusuarios deben estar delimitados mediante comillas dobles.

Notas:

- Si es usted un usuario existente, debe estar conectado y autentificado para cambiar su propia contraseña Sólo puede cambiar su propia contraseña.
- La sentencia GRANT no se puede utilizar con marcadores de parámetros ni con la función SQLPrepare().
- Intentar conseguir los privilegios de GRANT mientras se está conectado con un usuario no autorizado devuelve SQLSTATE 42502. Especificar una contraseña incorrecta con la sentencia GRANT ocasiona un SQLSTATE 42506.

Ejemplo:

Ejemplo 1: El primer usuario se otorga a sí mismo la autentificación necesaria para realizar la operación GRANT, sobre una base de datos que todavía no se ha cifrado:

```
GRANT ENCRYPT ON DATABASE TO "jsk" USING "foo" NEW "foo"
```

Ejemplo 2: Ahora se crea y autentifica el usuario ″jsk″ (del *Ejemplo 1* anterior), que posee la conexión. Para que ″jsk″ añada otro usuario:

```
GRANT ENCRYPT ON DATABASE TO "xin" USING "foo" NEW "bar"
```

Ejemplo 3: El usuario ″jsk″, conectado actualmente, cambia su propia contraseña:

```
GRANT ENCRYPT ON DATABASE TO "jsk" USING "foo" NEW "fie"
```

Ejemplo 4: El usuario ″jsk″, que sigue conectado actualmente, utiliza su nueva contraseña para añadir otro usuario:

```
GRANT ENCRYPT ON DATABASE TO "thf" USING "fie" NEW "fum"
```

Consulta relacionada:

- “Visión general del soporte de sentencia de SQL de DB2 Everyplace” en la página 137
- “Compatibilidad entre tipos de datos para las operaciones de asignación y comparación” en la página 176
- “Marcadores de parámetro soportados de DB2 Everyplace” en la página 76
- “Listado de los SQLSTATE” en la página 180
- “Resumen de códigos de clase de SQLState” en la página 180

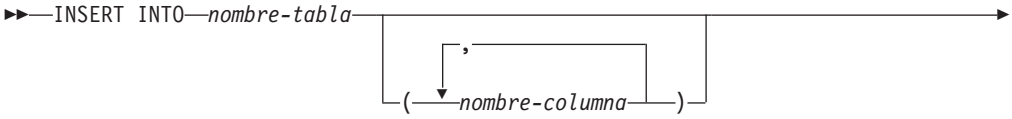
INSERT

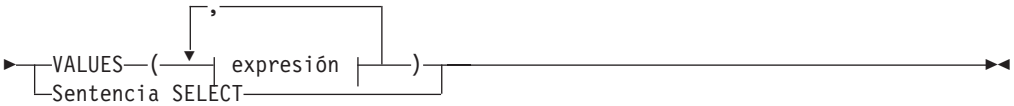
La sentencia INSERT inserta una o más filas en una tabla utilizando los valores proporcionados.

Invocación:

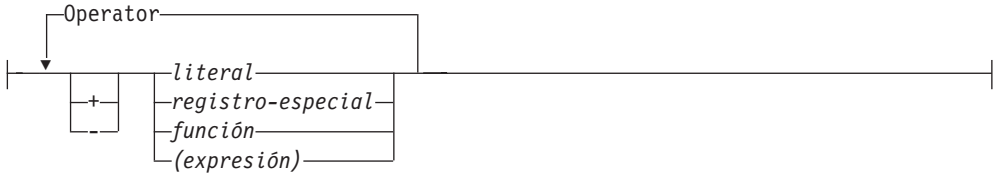
Esta sentencia puede utilizarse en un programa de aplicación utilizando las funciones de CLI de DB2 o emitirse a través del CLP.

Sintaxis:

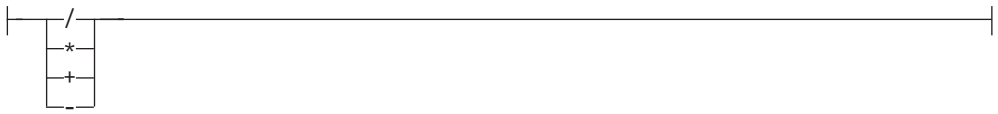




expresión:



operador:



Descripción:

INTO *nombre-tabla*
Designa la tabla de la operación de inserción. El nombre debe especificar una tabla existente, pero no una tabla de catálogo.

(*nombre-columna***,...)**
Especifica las columnas para las que se proporcionan valores de inserción. Cada nombre de columna debe ser un nombre no calificado que identifica una columna de la tabla. Una misma columna no puede estar especificada más de una vez.

La omisión de la lista de columnas equivale a una especificación implícita de una lista que designa, de izquierda a derecha, cada columna de la tabla.

VALUES
Presenta una fila de valores para insertar.

El número de valores para cada fila debe ser igual al número de nombres de la lista de columnas. El primer valor se inserta en la primera columna de la lista, el segundo valor en la segunda columna y así sucesivamente.

expresión
La expresión puede ser un literal, registro especial, función o expresión compleja.

No se da soporte a las operaciones aritméticas sobre los tipos de datos CHAR, VARCHAR, BLOB(n), DATE, TIME y TIMESTAMP.

literal
Un literal puede ser un valor de cualquier tipo de datos soportado como INTEGER, SMALLINT, DECIMAL, CHAR(n), VARCHAR(n), BLOB(n), DATE, TIME o TIMESTAMP.

registro-especial
Se pueden utilizar los registros especiales CURRENT DATE, CURRENT TIME y CURRENT TIMESTAMP para generar la fecha, la hora y la indicación de hora actuales.

Sentencia SELECT
Especifica un conjunto de columnas nuevas en el formato de la tabla resultante de una sentencia SELECT. Pueden haber una, más de una o ninguna. Si la tabla resultante está vacía, SQLCODE se establece en +100 y SQLSTATE se establece en '02000'. El objeto base de la sentencia SELECT no puede ser el objeto base de la sentencia INSERT.

Reglas:

Valores por omisión
Se inserta un valor por omisión o nulo en las columnas que no aparecen en la lista de columnas. Las columnas que no aceptan valores por omisión o nulos se deben incluir en la lista de columnas.

Longitud
Si el valor para insertar en una columna es un número, la columna debe ser una columna numérica que pueda representar la parte entera del número. Si el valor para insertar en una columna es una cadena de caracteres, la columna debe ser una columna de tipo carácter cuya longitud sea como mínimo la longitud de la cadena.

Asignación
Los valores de inserción se asignan a las columnas de acuerdo con las reglas de asignación descritas en el manual *DB2 Universal Database Consulta de SQL*.

Ejemplos:

Ejemplo 1: Inserción en la tabla EMPLOYEE de un empleado cuyas especificaciones son las siguientes:

- Número de empleado (EMPNO): 002001
- Nombre (FIRSTNAME): John
- Apellido (LASTNAME): Harrison
- Número de departamento (DEPT): 600
- Número de teléfono (PHONENO): 4900
- Salario (SALARY): 50000
- Fecha de contrato (HIREDATE): 01/12/1989

```
INSERT INTO EMPLOYEE
 VALUES ('002001', 'John', 'Harrison', '600', '4900', 50000, '01/12/1989')
```

Ejemplo 2: Inserción en la tabla EMPLOYEE de un nuevo empleado cuyas especificaciones son las siguientes:

- Número de empleado (EMPNO): 003002
- Nombre (FIRSTNAME): Jim
- Apellido (LASTNAME): Gray

```
INSERT INTO EMPLOYEE (EMPNO, FIRSTNAME, LASTNAME)
  VALUES ('003002', 'Jim', 'Gray')
```

Ejemplo 3: Creación de una tabla EMP_ACT_COUNT. Cargar EMP_ACT_COUNT con las filas de la tabla EMP_ACT con número de empleado (EMPNO) y número de proyectos comprometidos.

```
CREATE TABLE EMP_ACT_COUNT
   ( EMPNO CHAR(6)  NOT NULL,
     COUNT          INTEGER)
```

```
INSERT INTO EMP_ACT_COUNT
   SELECT EMPNO, COUNT(*)
   FROM EMP_ACT
   GROUP BY EMPNO
```

Restricciones:

1. Los tipos de datos de columna de la sentencia SELECT deben ser idénticos a las definiciones de columna de la tabla de destino (exceptuando la capacidad para contener nulos).
2. No están permitidas las cláusulas ORDER BY y LIMIT.

Consulta relacionada:

- "Visión general del soporte de sentencia de SQL de DB2 Everyplace" en la página 137
- "Compatibilidad entre tipos de datos para las operaciones de asignación y comparación" en la página 176
- "Marcadores de parámetro soportados de DB2 Everyplace" en la página 76
- "Listado de los SQLSTATE" en la página 180
- "Resumen de códigos de clase de SQLState" en la página 180

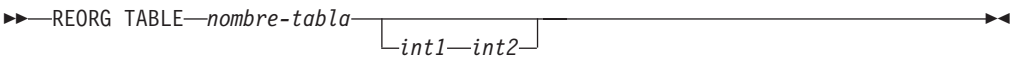
REORG TABLE

La sentencia REORG TABLE comprime los datos de la tabla especificada.

Invocación:

Esta sentencia puede utilizarse en un programa de aplicación utilizando las funciones de CLI de DB2 o emitirse a través del CLP.

Sintaxis:



Descripción:

REORG TABLE *nombre-tabla*
Identifica la tabla que es objeto de la operación de reorganización. El nombre debe identificar una tabla existente.

int1
Porcentaje mínimo opcional de bytes que es necesario recuperar.

int2
Número mínimo de bytes que es necesario recuperar para que se ejecute la compresión de la tabla.

Reglas:

- Los valores opcionales *int1* y *int2* deben especificarse conjuntamente o no especificar ninguno de ellos.
- El valor opcional *int1* debe ser un número no negativo.
- El valor opcional *int1* debe estar comprendido entre 0 y 100.

Notas:

- DB2 Everyplace puede invocar internamente una reorganización de tabla.

- El primer parámetro opcional es el porcentaje de bytes no utilizables que la tabla debe contener (es decir, un 10 por ciento significa que ″como mínimo el 10 por ciento del espacio no es utilizable″). El segundo parámetro opcional es el número de bytes no utilizables que la tabla debe contener (es decir, 1000 significará que ″como mínimo 1000 bytes deben ser espacio no utilizable″). Para que tenga lugar una reorganización real de la tabla, se deben cumplir ambos criterios.
- Si no se especifica ningún parámetro, DB2 Everyplace utiliza valores por omisión para estas opciones. El porcentaje por omisión es 30 y el número de bytes por omisión es 6144. Así, ″reorg table t1″ es lo mismo que ″reorg table t1 30 6144″.
- Si la modalidad de reorganización se establece en habilitada, DB2 Everyplace reorganizará automáticamente una tabla. Si la reorganización está habilitada, después de ejecutar una sentencia DELETE o UPDATE se ejecuta una operación ″reorg table nombre_tabla 50 30270″ para la tabla de destino. Si la reorganización está habilitada, al final del proceso de una sentencia DROP TABLE se ejecuta una operación ″reorg table DB2eSYSTABLES 30 10240″ (también para DB2eSYSCOLUMNS y DB2eSYSRELS).
- En un programa C/C++, la modalidad de reorganización se establece utilizando la función SQLSetStmtAttr de CLI/ODBC con el atributo SQL_ATTR_REORG_MODE. En un programa JAVA, establece la modalidad de reorganización el método enableReorg de la interfaz DB2eStatement. El valor por omisión consiste en que la reorganización esté habilitada.
- La reorganización de una tabla comprime el archivo de datos que contiene la tabla, reclamando físicamente el espacio no utilizable creado por supresiones y actualizaciones. Luego se actualizan los índices de la tabla de forma que apunten a la nueva dirección física de las filas.
- Las Tablas base del catálogo de DB2 Everyplace se pueden reorganizar.
- No se debe producir ninguna otra actividad en la base de datos mientras se ejecuta una sentencia REORG TABLE.

Ejemplos:

La tabla VNNURSE se comprime utilizando los valores por omisión.

```
REORG TABLE VNNURSE
```

Consulta relacionada:

- “Visión general del soporte de sentencia de SQL de DB2 Everyplace” en la página 137
- “Compatibilidad entre tipos de datos para las operaciones de asignación y comparación” en la página 176
- “Marcadores de parámetro soportados de DB2 Everyplace” en la página 76
- “Listado de los SQLSTATE” en la página 180
- “Resumen de códigos de clase de SQLState” en la página 180

REVOKE

La sentencia REVOKE permite que un usuario conectado y autentificado revoque los privilegios de cifrado de un usuario existente.

Invocación:

Esta sentencia puede utilizarse en un programa de aplicación utilizando las funciones de CLI de DB2 o emitirse a través del CLP.

Sintaxis:

Descripción:

usuario
Identifica el usuario cuyos privilegios de cifrado se revocan.

Reglas:

- El parámetro de usuario debe ser un identificador delimitado. Tiene una longitud limitada a 254 bytes.
- Para caracteres de byte múltiple, internamente se utiliza la codificación UTF-8 para el almacenamiento. Por consiguiente, los nombres de usuario escritos utilizando juegos de caracteres internacionales tienen una longitud limitada.
- Si se eliminan todos los usuarios que tienen privilegios de cifrado, se puede seguir accediendo a las tablas cifradas durante la sesión actual. Una vez que finalice la sesión actual, las tablas cifradas dejarán de estar disponibles.

Notas:

- Para revocar privilegios de un usuario existente, un usuario debe estar conectado y autentificado. Si es usted un usuario conectado y autentificado, puede revocar privilegios de cualquier usuario, incluido usted mismo.
- La sentencia REVOKE no se puede utilizar con marcadores de parámetros ni con la función SQLPrepare().
- Intentar conseguir los privilegios de REVOKE mientras se está conectado como usuario no autorizado devuelve SQLSTATE 42502. Intentar revocar (REVOKE) privilegios de un usuario inexistente da como resultado SQLSTATE 42501.

Ejemplo:

El usuario conectado y autentificado actualmente elimina los privilegios de cifrado del usuario ″jsk″:

```
REVOKE ENCRYPT ON DATABASE FROM "jsk"
```

Consulta relacionada:

- “Visión general del soporte de sentencia de SQL de DB2 Everyplace” en la página 137
- “Compatibilidad entre tipos de datos para las operaciones de asignación y comparación” en la página 176
- “Marcadores de parámetro soportados de DB2 Everyplace” en la página 76
- “Listado de los SQLSTATE” en la página 180
- “Resumen de códigos de clase de SQLState” en la página 180

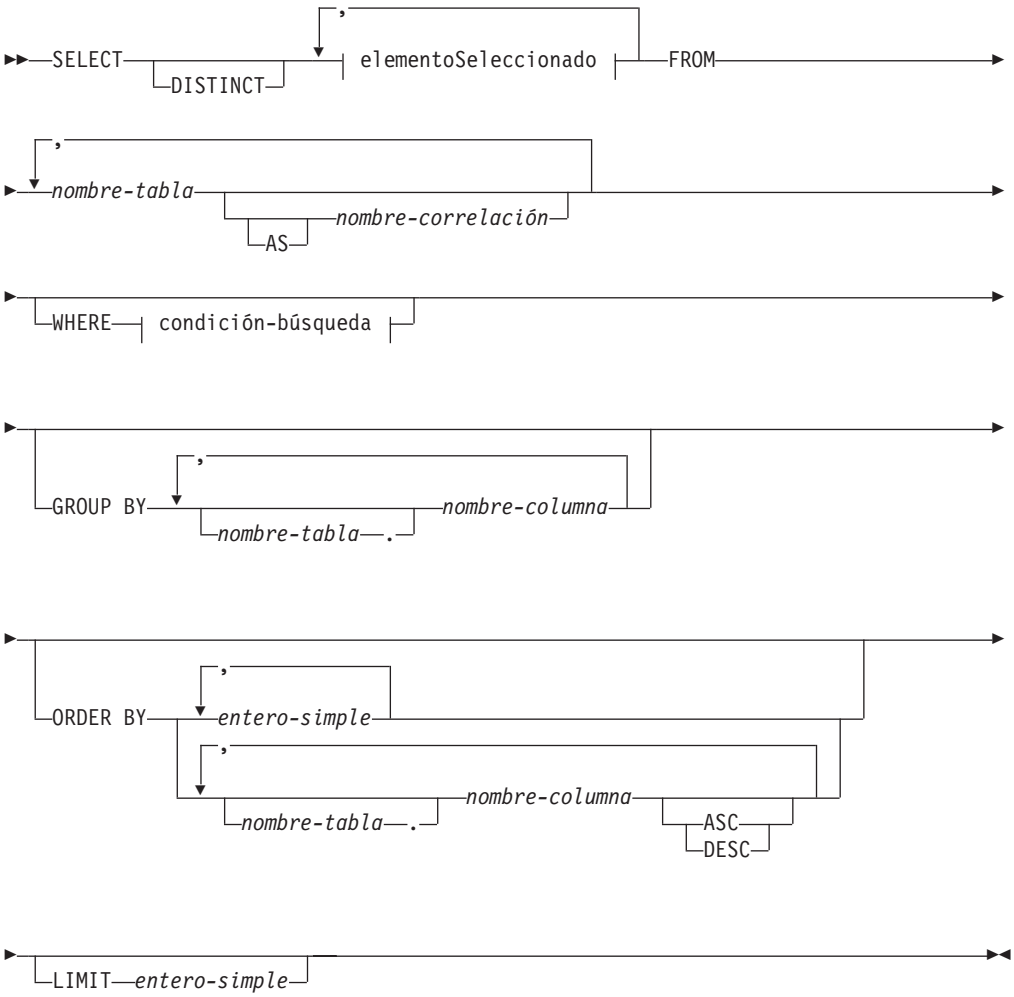
SELECT

La sentencia SELECT es una modalidad de consulta.

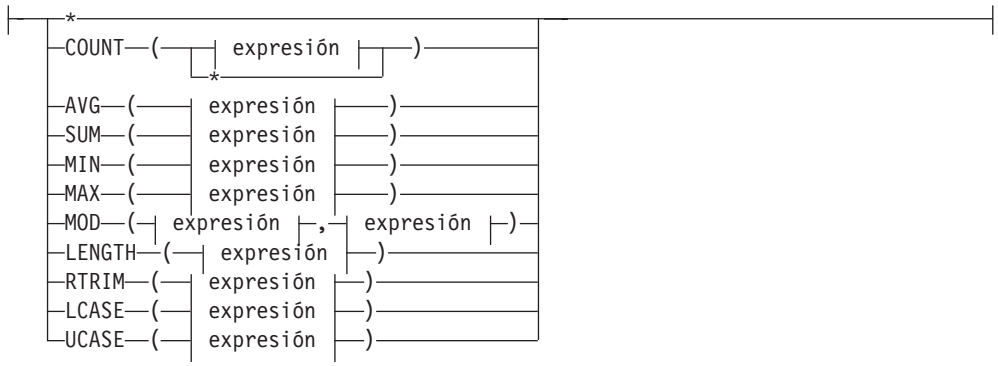
Invocación:

Esta sentencia puede utilizarse en un programa de aplicación utilizando las funciones de CLI de DB2 o emitirse a través del CLP.

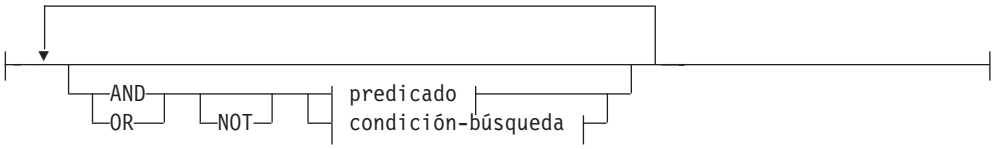
Sintaxis:



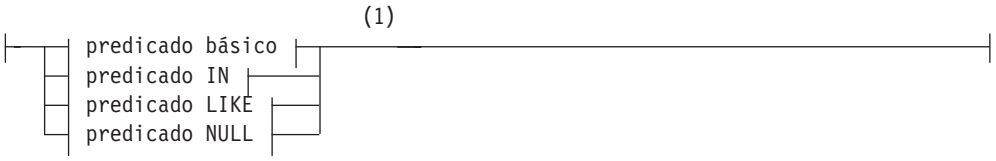
elementoSeleccionado:



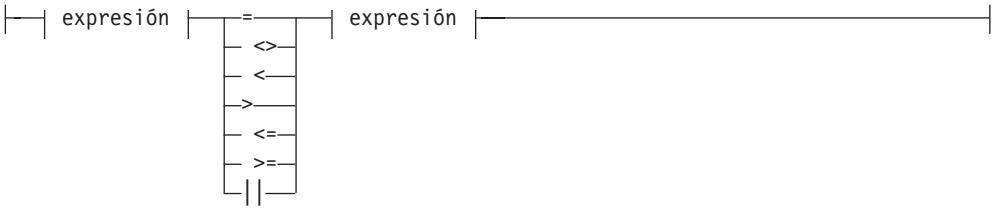
condición-búsqueda:



predicado:

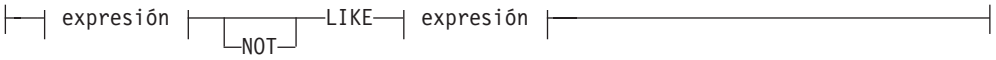


predicado básico:

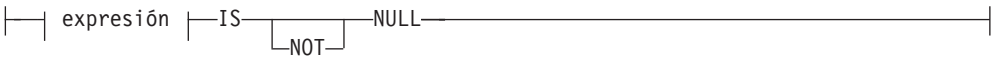


predicado IN:

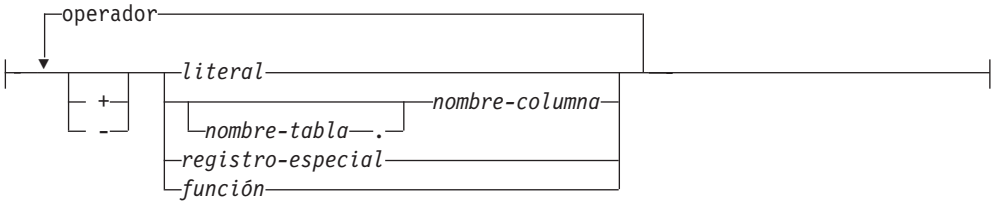
predicado LIKE:



predicado NULL:



expresión:



operador:

Notas:

1 Las expresiones BLOB sólo están permitidas en predicados NULL.

Descripción:

elementoSeleccionado

* Especifica todas las columnas. Si se especifica un asterisco (*), debe ser el único elemento seleccionado.

COUNT(*)
La función COUNT obtiene el número de filas o valores de un conjunto de filas o valores. El argumento de COUNT(*) es un conjunto de filas. El resultado es el número de filas del conjunto. En la cuenta se incluyen también las filas que sólo contengan valores nulos.

expresión
La *expresión* puede ser un literal, nombre de columna, registro especial o función. Las funciones válidas son: COUNT, AVG, SUM, MIN, MAX, MOD, LENGTH y RTRIM.

No se da soporte a las operaciones aritméticas sobre los tipos de datos CHAR, VARCHAR, BLOB(n), DATE, TIME y TIMESTAMP.

literal
Un *literal* puede ser un valor cuyo tipo de datos es INTEGER, SMALLINT, DECIMAL, CHAR(n), VARCHAR(n), BLOB(n), DATE, TIME y TIMESTAMP.

nombre-tabla
Designa la tabla donde reside la columna que es objeto de la consulta.

. Carácter que separa las dos partes que forman el identificador de columna, *nombre-tabla.nombre-columna*.

nombre-columna
Designa la columna que es objeto de la consulta.

COUNT(*expresión***)**
El argumento de COUNT(*expresión*) es un conjunto de filas. La función se aplica al conjunto de filas que se obtiene a partir de los valores argumento por eliminación de los valores nulos. El resultado es el número de valores no nulos del conjunto, incluidos los duplicados.

AVG(*expresión***)**
La función AVG(*expresión*) obtiene el valor promedio de los valores de *expresión*. Los valores argumento deben ser números y su suma debe estar dentro del rango del tipo de datos del resultado. La función se aplica al conjunto de valores que se obtiene a partir de los valores argumento por eliminación de los valores nulos. El resultado puede ser nulo.

SUM(*expresión***)**
La función SUM(*expresión*) obtiene la suma de los valores de *expresión*. Los valores argumento deben ser números y su suma debe estar dentro del

rango del tipo de datos del resultado. La función se aplica al conjunto de valores que se obtiene a partir de los valores argumento por eliminación de los valores nulos.

MIN(*expresión***)**
La función MIN(*expresión*) obtiene el valor mínimo del conjunto de valores de *expresión*. Los valores argumento pueden ser cualquier tipo de datos interno excepto BLOB. La función se aplica al conjunto de valores que se obtiene a partir de los valores argumento por eliminación de los valores nulos.

MAX(*expresión***)**
La función MAX(*expresión*) obtiene el valor máximo del conjunto de valores de *expresión*. Los valores argumento pueden ser cualquier tipo de datos interno excepto BLOB. La función se aplica al conjunto de valores que se obtiene a partir de los valores argumento por eliminación de los valores nulos.

MOD(*expresión, expresión***)**
La función MOD(*expresión, expresión*) obtiene el resto de dividir el primer argumento por el segundo. El resultado es negativo sólo si el primer argumento es negativo.

El primer y segundo argumento puede ser de tipo SMALLINT o INTEGER.

El resultado de la función es SMALLINT si ambos argumentos son SMALLINT; en otro caso, el resultado es INTEGER. El resultado puede ser nulo; si cualquiera de los argumentos es nulo, el resultado es el valor nulo.

(*expresión* || *expresión***)**
La función *expresión* || *expresión*) la concatenación de dos argumentos de serie. Los dos argumentos deben ser de tipos compatibles.

El resultado de la función es una serie. Su longitud es la suma de las longitudes de los dos argumentos. Si el argumento puede ser nulo, el resultado también; si el argumento es nulo, el resultado es el valor nulo.

LENGTH(*expresión***)**

La función LENGTH(*expresión*) obtiene la longitud de un valor.

El argumento puede ser una expresión que devuelve un valor de los siguientes tipos de datos incorporados.

- VARCHAR
- CHAR
- BLOB

El resultado de la función es un entero. Si el argumento puede ser nulo, el resultado también; si el argumento es nulo, el resultado es el valor nulo.

El resultado es la longitud del argumento. La longitud de una serie de longitud variable es la longitud real, no la longitud máxima.

La longitud de un BLOB es el número de bytes utilizados para representar el valor.

Considere una columna de tipo VARCHAR(50) llamada ADDRESS con un valor de '895 Don Mills Road'. `LENGTH(ADDRESS)` devuelve el valor 18.

RTRIM(*expresión***)**

La función RTRIM(*expresión*) elimina espacios en blanco del final de la serie.

El argumento puede ser de tipo de datos CHAR o VARCHAR.

El tipo de datos resultante de la función es siempre VARCHAR.

El parámetro de longitud del tipo de datos devuelto es el mismo que el parámetro de longitud del tipo de datos del argumento.

La longitud real del resultado para series de caracteres es la longitud de la expresión de la serie menos el número de bytes de caracteres en blanco eliminados. La longitud real del resultado para series de gráficos es la longitud (en el número de caracteres de doble byte) de la expresión de la serie menos el número de caracteres en blanco de doble byte eliminados. Si se eliminan todos los caracteres, el resultado es una serie de longitud variable y vacía (con longitud cero).

Si el argumento puede ser nulo, el resultado también; si el argumento es nulo, el resultado es el valor nulo.

Considere una columna de tipo CHAR(50) llamada NAME con un valor de 'Cliff '. `RTRIM(NAME)` devuelve 'Cliff'. `LENGTH(RTRIM(NAME))` devuelve 5.

LCASE / UCASE

La función LCASE o LOWER devuelve una serie en la que todos los caracteres SBCS se han convertido a caracteres en minúsculas. Es decir, los caracteres A-Z se convertirán en los caracteres a-z y los caracteres con marcas diacríticas se convertirán a sus equivalentes en minúsculas en el caso de que éstas existan.

El argumento debe ser una expresión cuyo valor sea un tipo de datos CHAR o VARCHAR.

El resultado de la función tiene el mismo tipo de datos y atributo de longitud que el argumento. Si el argumento puede ser nulo, el resultado también; si el argumento es nulo, el resultado es el valor nulo.

Asegúrese de que los caracteres del valor de la columna JOB de la tabla EMPLOYEE se devuelven en caracteres en minúsculas. Por ejemplo:

```
SELECT LCASE(JOB)
     FROM EMPLOYEE
     WHERE EMPNO = '000020';
```

registro-especial
Se pueden utilizar los registros especiales CURRENT DATE, CURRENT TIME y CURRENT TIMESTAMP para generar la fecha, la hora y la indicación de hora actuales.

FROM
La cláusula FROM especifica una tabla resultante intermedia.

Si se especifica una referencia de tabla, la tabla resultante intermedia es simplemente el resultado de esa referencia de tabla. Si se especifica más de una tabla de referencia, la tabla resultante intermedia consta de todas las combinaciones posibles de las filas de las referencias de tabla especificadas (el producto cartesiano). Cada fila del resultado es una fila de la primera referencia de tabla concatenada con una fila de la segunda referencia de tabla, concatenada a su vez con una fila de la tercera y así sucesivamente. El número de filas del resultado es el producto del número de filas de todas las referencias de tabla individuales. Se puede especificar un máximo de 20 tablas en la cláusula FROM.

nombre-tabla
Cada *nombre-tabla* especificado como referencia de tabla debe identificar una tabla existente.

AS
Identifica la definición de tabla.

nombre-correlación
Cada *nombre-correlación* identifica el *nombre-tabla* inmediato anterior. Si se especifica un nombre de correlación para una tabla, cualquier referencia calificada a una columna de la tabla debe utilizar el nombre de correlación en lugar del nombre de tabla. Si se especifica dos veces un mismo *nombre de tabla*, al menos una de las especificaciones debe ir seguida por un *nombre de correlación*. El *nombre de correlación* se utiliza para calificar referencias a las columnas de la tabla. Como calificador, el nombre de correlación se puede utilizar para evitar ambigüedades o para establecer una referencia correlacionada. Puede también utilizarse simplemente como nombre abreviado para una tabla.

WHERE
Especifica una condición que selecciona filas. Se puede omitir la cláusula o especificar una condición de búsqueda. Si se omite la cláusula, se seleccionan todas las filas de la tabla.

condición-búsqueda
La *condición de búsqueda* especifica una condición que es verdadera, falsa o desconocida para una fila determinada.

El resultado de una *condición de búsqueda* se obtiene aplicando los *operadores lógicos* especificados (AND, OR, NOT) al resultado de cada predicado especificado. Un predicado compara dos valores. Si no se especifican operadores lógicos, el resultado de la condición de búsqueda es el resultado del predicado especificado.

Las condiciones de búsqueda que están entre paréntesis se evalúan en primer lugar. Si no se especifica un orden de evaluación mediante el uso de paréntesis, NOT se aplica antes que AND y AND se aplica antes que OR. El orden en el que se evalúan los operadores con igual nivel de prioridad es indefinido, para permitir la optimización de las condiciones de búsqueda.

La *condición de búsqueda* se aplica a cada fila de la tabla y las filas seleccionadas son aquellas para las cuales el resultado de la *condición de búsqueda* es verdadero.

Cada *nombre de columna* especificado en la condición de búsqueda debe identificar una columna de la tabla.

NOT
Si se especifica NOT, se invierte el resultado del predicado.

expresión
La *expresión* puede ser un literal, nombre de columna, registro especial o función.

No se da soporte a las operaciones aritméticas sobre los tipos de datos CHAR, VARCHAR, BLOB(n), DATE, TIME y TIMESTAMP.

literal
Un *literal* puede ser un valor cuyo tipo de datos es INTEGER, SMALLINT, DECIMAL, CHAR(n), VARCHAR(n), BLOB(n), DATE, TIME o TIMESTAMP.

nombre-tabla
Designa la tabla donde reside la columna que es un operando del predicado.

. Carácter que separa las dos partes que forman el identificador de columna, *nombre-tabla.nombre-columna*.

nombre-columna
Identifica la columna que es un operando del predicado.

registro-especial
Identifica el registro especial que es un operando del predicado. Se pueden utilizar los registros especiales CURRENT DATE, CURRENT TIME y CURRENT TIMESTAMP para generar la fecha, la hora y la indicación de hora actuales.

función
Puede incluir las funciones LCASE, UCASE, MOD, LENGTH y RTRIM.

operador
Puede ser cualquiera de los operadores siguientes:

= Igual a.

<> No igual a.

< Menor que.

> Mayor que.

<= Menor o igual que.

>= Mayor o igual que.

|| Devuelve la concatenación de dos argumentos de serie.

LIKE Coincide con una serie de caracteres. Utilice un símbolo de subrayado de SBCS (juego de caracteres de un solo byte) para representar un carácter SBCS individual. Utilice un símbolo de subrayado de DBCS (juego de caracteres de doble byte) para representar un carácter DBCS individual. Por ejemplo, la condición `WHERE PART_NUMBER LIKE '_0'` obtiene todos los números de pieza de 2 dígitos que terminan en 0 (20, 30 y 40, por ejemplo). Utilice un símbolo de porcentaje (del juego de caracteres SBCS o DBCS) para representar una serie de caracteres SBCS o DBCS, o la ausencia de caracteres. Por ejemplo, la condición `WHERE DEPT_NUMBER LIKE '2%'` obtiene todos los números de departamento (DEPT_NUMBER) que comienzan con el número 2 (por ejemplo, 20, 27 ó 234).

NOT LIKE
Indica que al menos uno de los caracteres es diferente.

IN Encuentra las coincidencias de una colección de valores. El predicado IN compara un valor con una colección de valores.

Ejemplos:

SELECT lname, fname FROM emp WHERE state IN (’CA’, ’AZ’, ’OR’);

SELECT c1 FROM t1 WHERE c1*5-6 IN (mod(c2,2)+5,c3+4/2);

NOT IN
No coincide con una colección de valores. El predicado NOT IN compara un valor con una colección de valores.

Ejemplos:

SELECT empid FROM emp WHERE city NOT IN ('San Jose', 'Morgan Hill', 'Santa Clara');

IS NULL
Contiene el valor nulo.

IS NOT NULL
No contiene el valor nulo.

AND
Si se especifica, el operador lógico AND se aplica al resultado de cada predicado especificado.

OR
Si se especifica, el operador lógico OR se aplica al resultado de cada predicado especificado.

GROUP BY
Especifica una tabla resultante intermedia formada por un agrupamiento de las filas de R. R es el resultado de la cláusula anterior de la subselección.

ORDER BY
Especifica una ordenación de las filas de la tabla resultante.

nombre-columna
Suele designar una columna de la tabla resultante. En este caso, *nombre-columna* debe ser el nombre de una columna que aparece en la lista de selección.

entero-simple
Debe ser mayor que 0 y no ser mayor que el número de columnas de la tabla resultante. El entero *n* identifica la columna que ocupa la posición *n* en la tabla resultante.

ASC
Utiliza los valores de la columna en orden ascendente.

DESC
Utiliza los valores de la columna en orden descendente.

LIMIT *entero-simple*
Limita el número de filas que se devuelven a la aplicación al primer número *n* de filas del conjunto de respuestas donde *n* es un entero. Debe ser mayor que 0.

Operadores relacionales
Puede ser uno de los operadores siguientes:

+ Sumar

- Restar

* Multiplicar

/ Dividir por

Reglas:

Las columnas con datos de tipo BLOB no se pueden utilizar en cláusulas GROUP BY, ORDER BY ni DISTINCT.

Notas:

- Una sentencia SELECT DISTINCT puede contener un máximo de ocho columnas.
- Una cláusula GROUP BY puede contener un máximo de ocho columnas.
- Una cláusula ORDER BY puede contener un máximo de ocho columnas.
- Todas las columnas que se han especificado en la cláusula ORDER BY deben aparecer en la lista de selección. Por ejemplo, la consulta siguiente no es válida:

  ```
  SELECT EMPNO, FIRSTNAME FROM EMPLOYEE ORDER BY LASTNAME
  ```

  La consulta siguiente es válida:

  ```
  SELECT LASTNAME, EMPNO, FIRSTNAME FROM EMPLOYEE ORDER BY LASTNAME
  ```

Ejemplos:

Ejemplo 1: Este ejemplo selecciona los empleados (EMPNO y LASTNAME) de la tabla EMPLOYEE que se contrataron después de la fecha 01/01/1980 y los ordena de acuerdo con su apellido (LASTNAME).

```
SELECT EMPNO, LASTNAME FROM EMPLOYEE
 WHERE HIREDATE > '01/01/1980'
 ORDER BY LASTNAME
```

Ejemplo 2: Este ejemplo calcula el salario promedio para cada departamento de la tabla EMPLOYEE.

```
SELECT DEPT, AVG(SALARY) FROM EMPLOYEE
 GROUP BY DEPT
```

Ejemplo 3: Calcular el volumen máximo de ventas de cada región de ventas y visualizar el resultado por región, en orden de mayor a menor volumen de ventas.

```
SELECT REGION, MAX(SALES_VOL) FROM SALES
 GROUP BY REGION ORDER BY 2 DESC
```

Consulta relacionada:

- “Visión general del soporte de sentencia de SQL de DB2 Everyplace” en la página 137
- “Compatibilidad entre tipos de datos para las operaciones de asignación y comparación” en la página 176
- “Marcadores de parámetro soportados de DB2 Everyplace” en la página 76
- “Listado de los SQLSTATE” en la página 180
- “Resumen de códigos de clase de SQLState” en la página 180

UPDATE

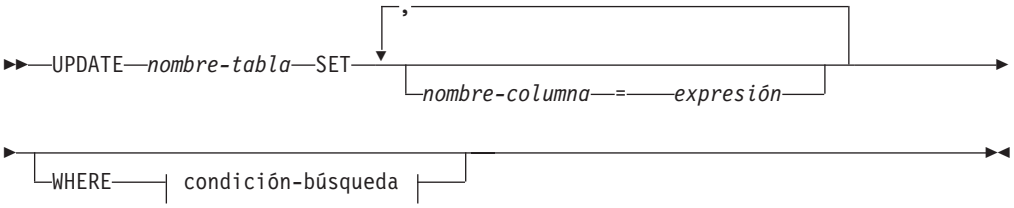
La sentencia UPDATE actualiza los valores de columnas especificadas en las filas de una tabla.

Invocación:

Esta sentencia puede utilizarse en un programa de aplicación utilizando las funciones de CLI de DB2 o emitirse a través del CLP.

Sintaxis:

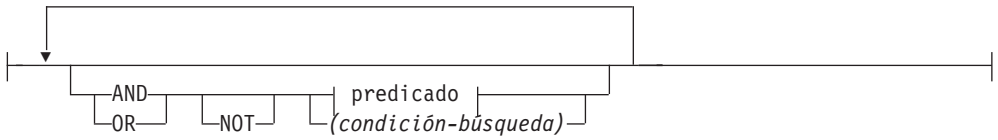
UPDATE



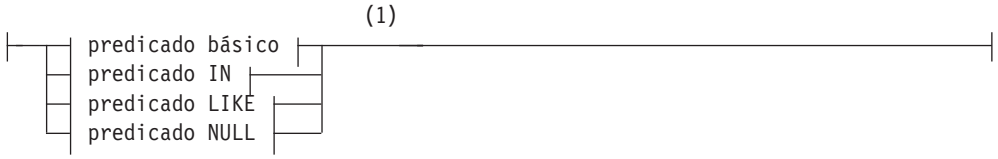
expresión:

operador:

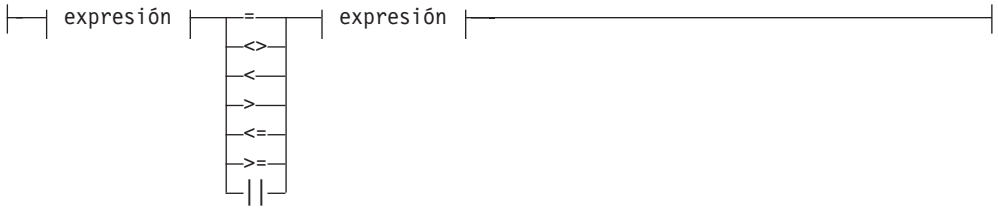
condición-búsqueda:



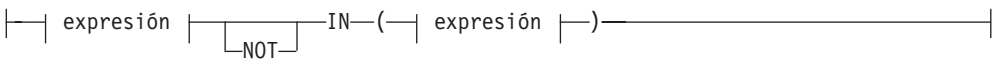
predicado:



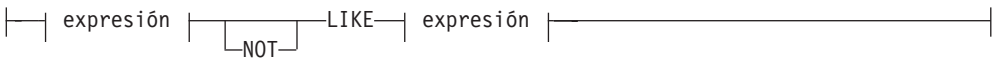
predicado básico:



predicado IN:



predicado LIKE:



predicado NULL:

operador relacional:

Notas:

1 Las expresiones BLOB sólo están permitidas en predicados NULL.

Descripción:

nombre-tabla
Es el nombre de la tabla para actualizar. El nombre debe identificar una tabla descrita en el catálogo, pero no una tabla de catálogo.

SET
Define la asignación de valores a nombres de columna.

nombre-columna
Designa una columna para actualizar. El *nombre de columna* debe identificar una columna de la tabla especificada. No se puede especificar una columna más de una vez (SQLSTATE 42701).

expresión
Una *expresión* puede ser un literal, nombre de columna o registro especial.

No se da soporte a las operaciones aritméticas sobre los tipos de datos BLOB(n), DATE, TIME y TIMESTAMP.

literal
Un *literal* puede ser un valor cuyo tipo de datos es INTEGER, SMALLINT, DECIMAL, CHAR(n), VARCHAR(n), BLOB(n), DATE, TIME o TIMESTAMP.

registro-especial
Se pueden utilizar los registros especiales CURRENT DATE, CURRENT TIME y CURRENT TIMESTAMP para generar la fecha, la hora y la indicación de hora actuales.

WHERE
Define una condición que indica las filas que deben actualizarse. Se puede omitir la cláusula o proporcionar una condición de búsqueda. Si se omite la cláusula, se actualizan todas las filas de la tabla.

condición-búsqueda
La *condición de búsqueda* especifica una condición que es verdadera, falsa o desconocida para una fila determinada.

El resultado de una *condición de búsqueda* se obtiene aplicando los *operadores lógicos* especificados (AND, OR, NOT) al resultado de cada predicado especificado. Un predicado compara dos valores. Si no se especifican operadores lógicos, el resultado de la condición de búsqueda es el resultado del predicado especificado.

Las condiciones de búsqueda que están entre paréntesis se evalúan en primer lugar. Si no se especifica un orden de evaluación mediante el uso de paréntesis, NOT se aplica antes que AND y AND se aplica antes que OR. El orden en el que se evalúan los operadores con igual nivel de prioridad es indefinido, para permitir la optimización de las condiciones de búsqueda.

La *condición de búsqueda* se aplica cada fila de la tabla y las filas actualizadas son aquéllas para las cuales el resultado de la *condición de búsqueda* es verdadero.

Cada *nombre de columna* especificado en la condición de búsqueda debe identificar una columna de la tabla.

Puede utilizar las funciones CONCAT, MOD, LENGTH y RTRIM en la expresión predicado de la condición de búsqueda. Para obtener más información sobre la función MOD, vea la página 166.

NOT
Si se especifica NOT, se invierte el resultado del predicado.

operador relacional
Puede ser cualquiera de los operadores siguientes:

= Igual a.

<> No igual a.

< Menor que.

> Mayor que.

<= Menor o igual que.

>= Mayor o igual que.

LIKE Coincide con una serie de caracteres. Utilice un símbolo de subrayado de SBCS (juego de caracteres de un solo byte) para representar un carácter SBCS individual. Utilice un símbolo de subrayado de DBCS (juego de caracteres de doble byte) para representar un carácter DBCS individual. Por ejemplo, la condición `WHERE PART_NUMBER LIKE '_0'` obtiene todos los números de pieza de 2 dígitos que terminan en 0 (20, 30 y 40, por ejemplo). Utilice un símbolo de porcentaje (del juego de caracteres SBCS o DBCS) para representar una cadena de caracteres SBCS o DBCS, o la

ausencia de caracteres. Por ejemplo, la condición `WHERE DEPT_NUMBER LIKE '2%'` obtiene todos los números de departamento (DEPT_NUMBER) que comienzan con el número 2 (por ejemplo, 20, 27 ó 234).

NOT LIKE
Indica que al menos uno de los caracteres es diferente.

IS NULL
Contiene el valor nulo.

IS NOT NULL
No contiene el valor nulo.

AND
Si se especifica, el operador lógico AND se aplica al resultado de cada predicado especificado.

OR
Si se especifica, el operador lógico OR se aplica al resultado de cada predicado especificado.

Reglas:

- ***Asignación:*** Los valores de actualización se asignan a las columnas de acuerdo con las reglas de asignación descritas en el manual *DB2 Universal Database Consulta de SQL*.
- UPDATE no se aplica nunca a registros suprimidos lógicamente.

Notas:

- En la modalidad del sistema, el bit de modificación se activa por omisión. Si está ejecutando su aplicación en la modalidad del sistema (SQL_DIRTYBIT_SET_BY_SYSTEM), no puede activar manualmente el bit de modificación. Si intenta activar el bit de modificación, se produce un error.

  Consulte el apartado en la página 278 para obtener más información.

Ejemplo:

Este ejemplo cambia el número de teléfono (PHONENO) del número de empleado (EMPNO) '003002' para que sea '1234' en la tabla EMPLOYEE.

```
UPDATE EMPLOYEE
 SET PHONENO = '1234'
 WHERE EMPNO = '003002'
```

Consulta relacionada:

- "Visión general del soporte de sentencia de SQL de DB2 Everyplace" en la página 137
- "Compatibilidad entre tipos de datos para las operaciones de asignación y comparación" en la página 176
- "Marcadores de parámetro soportados de DB2 Everyplace" en la página 76
- "Listado de los SQLSTATE" en la página 180
- "Resumen de códigos de clase de SQLState" en la página 180

Compatibilidad entre tipos de datos para las operaciones de asignación y comparación

Las operaciones de asignación se realizan durante la ejecución de las sentencias INSERT y UPDATE. Las operaciones de comparación se realizan durante la ejecución de las sentencias que incluyen predicados. Los tipos de datos de los operandos que intervienen deben ser compatibles, tal como se muestra en las tablas de la Tabla 17 a la Tabla 19.

Si la columna del tipo de datos contiene:

X Los tipos de datos de los operandos son compatibles.

en blanco
Los tipos de datos de los operandos no son compatibles.

Tabla 17. Compatibilidad entre tipos de datos, tabla 1

SQL, tipo de datos	INT	SMALLINT	DECIMAL	BLOB
INT	X	X	X	
VARCHAR				
BLOB				X
DECIMAL	X	X	X	
CHAR				
SMALLINT	X	X	X	
DATE				
TIME				
TIMESTAMP				

Tabla 18. Compatibilidad entre tipos de datos, tabla 2

SQL, tipo de datos	CHAR	VARCHAR
INT		
VARCHAR	X	X
BLOB		
DECIMAL		
CHAR	X	X
SMALLINT		
DATE	X	X
TIME	X	X
TIMESTAMP	X	X

Tabla 19. Compatibilidad entre tipos de datos, tabla 3

SQL, tipo de datos	DATE	TIME	TIMESTAMP
INT			
VARCHAR	X	X	X
BLOB			
DECIMAL			
CHAR	X	X	X

Tabla 19. Compatibilidad entre tipos de datos, tabla 3 (continuación)

SQL, tipo de datos	DATE	TIME	TIMESTAMP
SMALLINT			
DATE	X		
TIME		X	
TIMESTAMP			X

Consulta relacionada:

- "Visión general del soporte de sentencia de SQL de DB2 Everyplace" en la página 137
- "Marcadores de parámetro soportados de DB2 Everyplace" en la página 76
- "Listado de los SQLSTATE" en la página 180

Tipos de datos por omisión y simbólicos de SQL

Tabla 20. Tipos de datos por omisión y simbólicos de SQL

Tipos de datos de SQL	Tipos de datos simbólicos de SQL	Tipos de datos C simbólicos por omisión
BLOB	SQL_BLOB	SQL_C_BINARY
CHAR	SQL_CHAR	SQL_C_CHAR
DATE	SQL_TYPE_DATE	SQL_C_TYPE_DATE
DECIMAL	SQL_DECIMAL	SQL_C_CHAR
INTEGER	SQL_INTEGER	SQL_C_LONG
SMALLINT	SQL_SMALLINT	SQL_C_SHORT
TIME	SQL_TYPE_TIME	SQL_C_TYPE_TIME
TIMESTAMP	SQL_TYPE_TIMESTAMP	SQL_C_TYPE_TIMESTAMP
VARCHAR	SQL_VARCHAR	SQL_C_CHAR

Atributos de tipos de datos

Se muestra información para los atributos de tipo de datos siguientes:

- Precisión
- Escala
- Longitud
- Tamaño de pantalla

Precisión:

La precisión de una columna numérica o parámetro hace referencia al número máximo de dígitos que utiliza el tipo de datos de la columna o parámetro. La precisión de una columna no numérica o parámetro hace referencia generalmente a la longitud máxima o a la longitud definida de la columna o parámetro. La tabla siguiente define la precisión para cada tipo de datos SQL.

Tabla 21. Precisión

fSqlType	Precisión
SQL_CHAR SQL_VARCHAR	La longitud definida de la columna o parámetro. Por ejemplo, la precisión de una columna definida como CHAR(10) es 10.
SQL_DECIMAL	El número máximo de dígitos definido. Por ejemplo, la precisión de una columna definida como DECIMAL(10,3) es 10.
SQL_SMALLINT a	5
SQL_INTEGER a	10
SQL_BLOB	La longitud definida de la columna o parámetro. Por ejemplo, la precisión de una columna definida como BLOB(10) es 10.
SQL_DATE a	10 (el número de caracteres en el formato aaaa-mm-dd).
SQL_TIME a	8 (el número de caracteres en el formato hh:mm:ss).
SQL_TIMESTAMP	26 (El número de caracteres en el formato "aaaa-mm-dd-hh.mm.ss.ffffff" que utiliza el tipo de datos de TIMESTAMP.)

a: El argumento *cbParamDef* de `SQLBindParameter()` se ignora para este tipo de datos.

Escala:

La escala de una columna numérica o parámetro hace referencia al número máximo de dígitos a la derecha de la coma decimal. La tabla siguiente define la escala para cada tipo de datos SQL.

Tabla 22. Escala

fSqlType	Escala
SQL_CHAR SQL_VARCHAR	No aplicable.
SQL_DECIMAL	El número de dígitos definido a la derecha de la coma decimal. Por ejemplo, la escala de una columna definida como DECIMAL(10, 3) es 3.
SQL_SMALLINT SQL_INTEGER	0
SQL_BLOB	No aplicable.
SQL_DATE SQL_TIME	No aplicable.
SQL_TIMESTAMP	6 (El número de dígitos a la derecha de la coma decimal en el formato "aaaa-mm-dd-hh.mm.ss.ffffff".)

Longitud:

La longitud de una columna es el número máximo de bytes devueltos a la aplicación en el momento en que los datos se transfieren a su tipo de datos C por

omisión. Para los datos de tipo carácter, la longitud no incluye el byte de finalización de nulo. Tenga en cuenta que la longitud de una columna puede ser diferente del número de bytes que se necesitan para almacenar los datos en la fuente de datos.

La tabla siguiente define la longitud para cada tipo de datos SQL.

Tabla 23. Longitud

fSqlType	Longitud
SQL_CHAR SQL_VARCHAR	La longitud definida de la columna. Por ejemplo, la longitud de una columna definida como CHAR(10) es 10.
SQL_DECIMAL	El número máximo de dígitos más dos. Puesto que estos tipos de datos se devuelven como series de caracteres, se necesitan caracteres para los dígitos, un signo y una coma decimal. Por ejemplo, la longitud de una columna definida como DECIMAL(10,3) es 12.
SQL_SMALLINT	2 (dos bytes).
SQL_INTEGER	4 (cuatro bytes).
SQL_BLOB	La longitud definida de la columna. Por ejemplo, la longitud de una columna definida como BLOB(10) es 10.
SQL_DATE SQL_TIME	6 (el tamaño de la estructura DATE_STRUCT o TIME_STRUCT).
SQL_TIMESTAMP	16 (el tamaño de la estructura TIMESTAMP_STRUCT).

Tamaño de pantalla:

El tamaño de pantalla de una columna es el número máximo de bytes necesarios para visualizar datos en formulario de caracteres. La tabla siguiente define el tamaño de pantalla para cada tipo de datos SQL.

Tabla 24. Tamaño de pantalla

fSqlType	Tamaño de pantalla
SQL_CHAR SQL_VARCHAR	La longitud definida de la columna. Por ejemplo, el tamaño de pantalla de una columna definida como CHAR(10) es 10.
SQL_DECIMAL	La precisión de la columna más dos (un signo, dígitos de precisión y una coma decimal). Por ejemplo, el tamaño de pantalla de una columna definida como DECIMAL(10,3) es 12.
SQL_SMALLINT	6 (un signo y 5 dígitos).
SQL_INTEGER	11 (un signo y 10 dígitos).
SQL_BLOB	La longitud definida de la columna por 2 (cada byte binario está representado por un número hexadecimal de 2 dígitos). Por ejemplo, el tamaño de pantalla de una columna definida como BLOB(10) es 20.

Tabla 24. Tamaño de pantalla (continuación)

fSqlType	Tamaño de pantalla
SQL_DATE	10 (una fecha en el formato aaaa-mm-dd).
SQL_TIME	8 (una hora en el formato hh:mm:ss).
SQL_TIMESTAMP	26 (una indicación de fecha y hora en el formato aaaa-mm-dd-hh.mm.ss.ffffff).

Listado de los SQLSTATE

Esta sección le ayudará a interpretar los mensajes de error generados desde SQL o CLI.

- "Resumen de códigos de clase de SQLState" contiene un listado de las categorías generales de los errores.
- "Mensajes de SQLSTATE notificados por SQL" en la página 181, "Mensajes de SQLSTATE notificados por CLI" en la página 185 y "Mensajes de SQLState notificados por JDBC" en la página 193 contienen descripciones de cada uno de los errores y para SQL, proporcionan el nombre de la función que lo generó.

Puede también obtener descripciones de los estados SQL (SQLSTATE) utilizando un procesador de línea de mandatos de DB2, en el caso de que haya instalado DB2 UDB:

1. Para abrir el procesador de línea de mandatos, seleccione **Inicio —> Programa —> DB2 —> Procesador de línea de mandatos**.
2. En la línea de mandatos, escriba `? [SQLSTATE]`.

Consulta relacionada:

- "Visión general del soporte de sentencia de SQL de DB2 Everyplace" en la página 137
- "Compatibilidad entre tipos de datos para las operaciones de asignación y comparación" en la página 176
- "Resumen de códigos de clase de SQLState"

Resumen de códigos de clase de SQLState

Los dos primeros caracteres de los mensajes de SQLSTATE incluidos en la Tabla 27 en la página 185 están en **negrita** para indicar el código de clase. Estos códigos de clase aparecen *resumidos* in Tabla 25.

Tabla 25. Códigos de clase de los SQLSTATE

Código	Clase
00	Terminación satisfactoria no calificada
01	Aviso
02	No hay datos
07	Error de SQL dinámico
08	Excepción de conexión
09	Excepción de acción desencadenada
0A	Característica no soportada
0F	Símbolo no válido
21	Violación de la cardinalidad

Tabla 25. Códigos de clase de los SQLSTATE (continuación)

Código	Clase
22	Excepción de datos
23	Violación de restricción
24	Estado no válido del cursor
25	Estado no válido de la transacción
26	Identificador no válido de sentencia de SQL
28	Especificación no válida de autorización
2D	Terminación no válida de transacción
2E	Nombre no válido de conexión
34	Nombre no válido de cursor
38	Excepción de función externa
39	Excepción de llamada a función externa
40	Retrotracción de transacción
42	Error de sintaxis o violación de regla de acceso
44	Violación de opción con comprobación
46	DDL de Java
51	Estado no válido de aplicación
54	Límite excedido de SQL o del producto
55	Objeto no encontrado en estado previo necesario
56	Diversos errores de SQL o del producto
57	Recurso no disponible o intervención del operador
58	Recurso erróneo del sistema
59	Error del Administrador de DB2 Everyplace
HY	Generado por el controlador CLI u ODBC de DB2
IM	Generado por el gestor de controladores ODBC

Consulta relacionada:

- “Visión general del soporte de sentencia de SQL de DB2 Everyplace” en la página 137
- “Compatibilidad entre tipos de datos para las operaciones de asignación y comparación” en la página 176
- “Listado de los SQLSTATE” en la página 180
- “Marcadores de parámetro soportados de DB2 Everyplace” en la página 76

Mensajes de SQLSTATE notificados por SQL

La Tabla 26 lista todos los SQLSTATE notificados por el motor SQL de DB2 Everyplace, de las sentencias de SQL.Los SQLSTATE notificados por CLI de DB2 aparecen en las descripciones de las funciones de CLI de DB2, en el “Resumen de las funciones de CLI de DB2” en la página 194.

Tabla 26. Mensajes de SQLSTATE notificados por SQL

SQLSTATE	Descripción	Explicación
01000	Aviso.	Mensaje informativo. (La función devuelve SQL_SUCCESS_WITH_INFO).

Tabla 26. Mensajes de SQLSTATE notificados por SQL (continuación)

SQLSTATE	Descripción	Explicación
01004	Valor truncado.	El valor fue truncado por una función de conversión de tipos de datos o función de ajuste del sistema.
01550	No se creó el índice.	No se creó el índice porque ya existe un índice con la descripción especificada.
02000	No se encontró ninguna fila.	No se encontró ninguna fila durante la ejecución de una sentencia FETCH, DELETE o UPDATE.
07001	Número incorrecto de parámetros.	No se ha enlazado un marcador de parámetro.
07005	Parámetro no válido.	El nombre de sentencia del cursor designa una sentencia preparada que no se puede asociar con un cursor.
07006	Variable no válida.	No se puede utilizar una variable de sistema principal de entrada debido a su tipo de datos.
08002	La conexión ya existe.	Ya existe una conexión.
22001	Es necesario truncar el valor.	Es necesario truncar el valor mediante una función de conversión de tipos de datos o función de ajuste del sistema.
22002	No se ha proporcionado ningún indicador de nulos.	No se puede asignar un valor NULL debido a la falta de espacio de almacenamiento.
22003	Valor numérico fuera de rango.	Un valor numérico no está dentro del rango de su columna destino.
22007	Formato no válido de fecha y hora.	La serie de caracteres utilizada para representar un valor de fecha y hora tiene una sintaxis incorrecta.
22008	Valor de fecha y hora fuera de rango.	La serie de caracteres utilizada para representar un valor de fecha y hora está fuera de rango.
22012	División por cero.	Se ha intentado una división por cero.
22504	Carácter MBCS fragmentado.	Los datos contienen un carácter de múltiples bytes mal formado.
23502	Valor nulo no permitido.	No está permitida la asignación de un valor nulo a una columna definida como NOT NULL.
23505	Los valores no son exclusivos.	La operación no era válida debido a que produciría claves duplicadas.
23513	Valor no válido.	La fila resultante de la sentencia INSERT o UPDATE no se ajusta a la definición de la restricción de comprobación.
23515	Se ha especificado más de una cláusula de clave primaria.	Se ha especificado más de una cláusula de clave primaria.
24000	Estado no válido del cursor.	*StatementHandle* estaba en un estado ejecutado, pero no había ningún conjunto resultante asociado al *StatementHandle*.
24501	Cursor no abierto.	No es válida una operación FETCH debido a que no se ha generado ningún conjunto resultante.
24505	Cursor no posicionado.	No es válida una operación FETCH debido a que el cursor no está situado sobre una fila.
34000	El nombre de cursor no es válido.	El nombre de cursor no es válido.
42501*	El ID de autorización no tiene permitido realizar la operación especificada sobre el objeto identificado	El usuario actual está intentando eliminar un privilegio de un usuario que no existe.

Tabla 26. Mensajes de SQLSTATE notificados por SQL (continuación)

SQLSTATE	Descripción	Explicación
42502*	El ID de autorización no tiene permitido realizar la operación especificada	El usuario actual no tiene una conexión autentificada. Cuando una aplicación (que no tiene la biblioteca de cifrado ni la CryptoPlugin.dll) ejecute un cifrado relacionado con mandatos de SQL (GRANT, REVOKE y CREATE TABLE), se devolverá un error ″42502″. Esto es así para impedir que las aplicaciones caigan.
42505*	Se ha producido un error de autorización de conexión.	Un usuario registrado intenta conectar y no se le puede autentificar.
42506*	Error de autorización de propietario.	No se ha podido autentificar el usuario conectado. (Contraseña incorrecta.)
42601	Error de sintaxis.	Se detectó un error de sintaxis en la sentencia de SQL.
42603	Una constante de tipo serie no tiene un delimitador final.	Una constante de tipo serie o identificador delimitado no tiene un delimitador final.
42610	Utilización no válida de un marcador de parámetro.	La sentencia contiene un marcador de parámetro que no es válido. Vea la Tabla 14 en la página 76 para conocer la utilización válida de los marcadores de parámetros.
42611	Especificación de longitud no válida.	Una especificación de longitud excede el límite.
42614	Una palabra clave duplicada no es válida.	Una palabra clave duplicada no es válida.
42621	La restricción de comprobación no es válida.	La restricción de comprobación no es válida.
42622	Nombre demasiado largo.	El nombre de un identificador es demasiado largo.
42702	Referencia ambigua a un nombre de columna.	Existe más de una posible columna referenciada.
42703	Nombre de columna no definido.	Un nombre de columna no está en las tablas referenciadas.
42704	Objeto no definido.	La tabla no existe.
42710	El objeto designado ya existe.	Ya existe una tabla con el mismo nombre.
42711	Nombre de columna duplicado.	Un mismo nombre de columna está especificado más de una vez.
42802	El número de valores no coincide con el número de columnas.	El número de valores asignados no es el mismo que el número de columnas especificadas o implícitas.
42803	Una referencia de columna contenida en la lista de selección no está especificada en la cláusula GROUP BY.	La lista de selección contiene un nombre de columna y una función de agregación, pero no existe ninguna cláusula GROUP BY.
42818	Tipos de datos incompatibles de los operandos.	Los tipos de datos de los operandos de una operación no son compatibles.
42820	Valor literal fuera de rango.	El valor numérico especificado no está dentro del rango aceptable.
42821	Tipos de datos incompatibles.	Un valor no es compatible con el tipo de datos de una columna destino.
42822	Elemento no válido de ORDER BY.	El elemento de ORDER BY no está en la lista de selección.
42824	Operando no válido de LIKE.	Un operando de LIKE no es una serie de caracteres o el primer operando no es una columna.
42829	FOR UPDATE OF no es válido.	FOR UPDATE OF no es válido, pues la tabla resultante designada por el cursor no se puede modificar.

Tabla 26. Mensajes de SQLSTATE notificados por SQL (continuación)

SQLSTATE	Descripción	Explicación
42830	La clave foránea no se ajusta a la descripción de la clave padre.	La clave foránea no se ajusta a la descripción de la clave padre.
42831	La clave primaria tiene columnas que pueden contener valores nulos.	Las columnas especificadas en la clave primaria no pueden contener nulos.
42832*	Acceso no autorizado a objetos del sistema.	La operación no está permitida para objetos del sistema.
42884	Nombre de función no conocido.	No se ha encontrado ninguna función o procedimiento con el nombre especificado y argumentos compatibles.
42887	Característica no soportada	La característica no está soportada en el release actual.
42894	El valor por omisión (DEFAULT) no es válido.	El valor por omisión (DEFAULT) no es válido.
42902	Referencia duplicada a una tabla de objetos.	La tabla de objetos de la sentencia INSERT también se identifica en una cláusula FROM.
42903	Existe una referencia no válida en una cláusula WHERE o SET.	Una cláusula WHERE o SET contiene una referencia, tal como una función de columna, que no es válida.
42962	No se puede utilizar como clave una columna de tipo LOB.	No se puede utilizar como clave primaria una columna de tipo LOB.
54001	Sentencia demasiado larga.	La sentencia de la consulta es demasiado larga.
54008	Clave demasiado larga.	Demasiadas columnas en una clave primaria, una clave foránea o en el índice.
54010	La longitud del registro de tabla es demasiado larga.	La longitud del registro de la tabla es demasiado larga.
55002	DB2ePLANTABLE no está definido correctamente.	EXPLAIN no se puede ejecutar con una declaración incorrecta de DB2ePLANTABLE.
55009	Archivo de sólo lectura.	El archivo es de sólo lectura. En un entorno de sólo lectura, solamente se pueden ejecutar consultas SELECT.
57001	Tabla no disponible.	REORG no se puede ejecutar en una tabla que está bajo el ámbito de una transacción.
57011	Falta de memoria.	El sistema no puede asignar memoria dinámica.
57014	El proceso se canceló debido a una interrupción.	La ejecución de una consulta se cancela debido a una interrupción del usuario.
58004	Error interno del sistema (continuar).	Se ha producido un error no grave del sistema.
58005	Error interno del sistema (detener).	Se ha producido un error grave del sistema.

Consulta relacionada:

- “Visión general del soporte de sentencia de SQL de DB2 Everyplace” en la página 137
- “Compatibilidad entre tipos de datos para las operaciones de asignación y comparación” en la página 176
- “Listado de los SQLSTATE” en la página 180
- “Marcadores de parámetro soportados de DB2 Everyplace” en la página 76
- “Resumen de códigos de clase de SQLState” en la página 180

Mensajes de SQLSTATE notificados por CLI

Tabla 27. Mensajes de SQLState notificados por CLI

SQLSTATE	Nombre de la función de CLI	Descripción	Explicación
01000	SQLAllocHandle	Aviso.	Mensaje informativo. (La función devuelve SQL_SUCCESS_WITH_INFO).
01000	SQLFreeHandle	Aviso.	Mensaje informativo. (La función devuelve SQL_SUCCESS_WITH_INFO).
01002	SQLDisconnect	Error de desconexión.	Se produjo un error durante la desconexión. Sin embargo, la desconexión se realizó satisfactoriamente. (La función devuelve SQL_SUCCESS_WITH_INFO).
01004	SQLDescribeCol	Datos truncados.	El nombre de columna devuelto en el argumento *ColumnName* era más largo que el valor especificado en el argumento *BufferLength*. El argumento *NameLengthPtr* contiene la longitud del nombre de columna completo. (La función devuelve SQL_SUCCESS_WITH_INFO).
01004	SQLFetch	Datos truncados.	Los datos devueltos para una o más columnas se truncaron. Los valores de tipo serie o los valores numéricos se truncan por la derecha. (Se devuelve SQL_SUCCESS_WITH_INFO si no se ha producido ningún error).
01004	SQLGetData	Datos truncados.	Los datos devueltos para la columna especificada (*ColumnNumber*) se han truncado. Los valores de tipo serie o los valores numéricos se truncan por la derecha. (Se devuelve SQL_SUCCESS_WITH_INFO.)
01S06*	SQLFetchScroll	Se ha intentado recuperar antes de que el conjunto resultante devolviera el primer conjunto de filas.	El conjunto de filas solicitado se solapó con el inicio del conjunto resultante cuando la posición actual estaba más allá de la primera fila y *FetchOrientation* era SQL_PRIOR, o bien *FetchOrientation* era SQL_RELATIVE con un desplazamiento F*etchOffset* negativo cuyo valor absoluto era menor o igual que el valor actual de SQL_ATTR_ROW_ARRAY_SIZE. (La función devuelve SQL_SUCCESS_WITH_INFO).
07005	SQLDescribeCol	La sentencia no devolvió un conjunto resultante.	La sentencia asociada al descriptor de sentencia (*StatementHandle*) no devolvió un conjunto resultante. No había ninguna columna para describir. (Invoque primero `SQLNumResultCols()` para determinar si hay alguna fila en el conjunto resultante).
07006	SQLBindParameter	Conversión no válida.	La conversión desde el tipo de datos identificado por el argumento *ValueType* al tipo de datos identificado por el argumento *ParameterType* no es una conversión válida. (Por ejemplo, la conversión desde SQL_C_DATE a SQL_DOUBLE.)
07006	SQLFetch	Conversión no válida.	El tipo de datos no se ha podido convertir de forma adecuada al tipo de datos especificado por *fCType* en `SQLBindCol()`.

Tabla 27. Mensajes de SQLState notificados por CLI (continuación)

SQLSTATE	Nombre de la función de CLI	Descripción	Explicación
07006	SQLGetData	Conversión no válida.	El tipo de datos no se puede convertir al tipo de datos C especificado por el argumento *TargetType*. La función se invocó anteriormente para el mismo valor de *ColumnNumber*, pero con un valor diferente de *TargetType*.
07009	SQLBindCol	Índice descriptor no válido.	El valor especificado para el argumento *ColumnNumber* excede el número máximo de columnas del conjunto resultante.
07009	SQLDescribeCol	Índice descriptor no válido.	El valor especificado para *ColumnNumber* era menor o igual que 0. El valor especificado para el argumento *ColumnNumber* era mayor que el número de columnas del conjunto resultante.
08001	SQLConnect	No se puede conectar con la fuente de datos.	DB2 CLI no pudo establecer una conexión con la fuente de datos (servidor).
08002	SQLConnect	Conexión en uso.	El descriptor de conexión (*ConnectionHandle*) especificado ya se había utilizado para establecer una conexión con una fuente de datos y la conexión sigue abierta.
08003	SQLAllocHandle	La conexión está cerrada.	El argumento *HandleType* era SQL_HANDLE_STMT, pero la conexión especificada por el argumento *InputHandle* no se abrió. El proceso de conexión debe finalizar satisfactoriamente (y se debe abrir la conexión) para que DB2 CLI asigne un descriptor de contexto de sentencia.
08003	SQLDisconnect	La conexión está cerrada.	La conexión especificada en el argumento *ConnectionHandle* no se abrió.
08004	SQLConnect	El servidor de aplicaciones rechazó el establecimiento de la conexión.	La fuente de datos (servidor) rechazó el establecimiento de la conexión.
08S01	SQLFreeHandle	Error en el enlace de comunicaciones.	El argumento *HandleType* era SQL_HANDLE_DBC y el enlace de comunicaciones entre DB2 CLI y la fuente de datos al que estaba intentando conectarse ha fallado antes de que la función terminara su proceso.
22002	SQLFetch	Almacenamiento intermedio de salida o de indicadores no valido.	El valor de puntero especificado para el argumento *pcbValue* en `SQLBindCol()` era un puntero nulo y el valor de la columna correspondiente es nulo. No existe ninguna manera de notificar SQL_NULL_DATA.
22002	SQLGetData	Almacenamiento intermedio de salida o de indicadores no valido.	El valor de puntero especificado para el argumento *StrLen_or_IndPtr* era un puntero nulo y el valor de la columna es nulo. No existe ninguna manera de notificar SQL_NULL_DATA.
22003	SQLExecDirect	Valor numérico fuera de rango.	Un valor numérico asignado a una columna de tipo numérico provocó el truncamiento de la parte entera del número, durante el proceso de asignación o al calcular un resultado intermedio.
22005	SQLGetData	Error de asignación.	Un valor devuelto era incompatible con el tipo de datos indicado por el argumento *TargetType*.

Tabla 27. Mensajes de SQLState notificados por CLI (continuación)

SQLSTATE	Nombre de la función de CLI	Descripción	Explicación
39001 *	SQLExecute	Una función definida por el usuario ha devuelto un SQLSTATE no válido.	Una función definida por el usuario ha devuelto un SQLSTATE no válido.
40003 **08**S01	SQLBindCol	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
40003 **08**S01	SQLBindParameter	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
40003 **08**S01	SQLDescribeCol	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
40003 **08**S01	SQLFreeStmt	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
40003 **08**S01	SQLGetData	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
40003 **08**S01	SQLNumResultCols	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
40003 **08**S01	SQLRowCount	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
42nnn*	SQLPrepare	Error de sintaxis.	Los SQLSTATE **42**nnn indican que hay varios problemas de sintaxis o de acceso en la sentencia. Los caracteres nnn representan cualquier SQLSTATE con ese código de clase. Ejemplo: **42**nnn hace referencia a cualquier SQLSTATE de la clase **42**.
42xxx	SQLExecDirect	Error de sintaxis o violación de la regla de acceso.	Los SQLSTATE **42**xxx denotan la existencia de diversos problemas de sintaxis o de acceso en la sentencia. xxx representa cualquier SQLSTATE con ese código de clase. Ejemplo: **42**xxx representa cualquier SQLSTATE perteneciente a la clase **42**.
42xxx	SQLNumResultCols	Error de sintaxis.	Los SQLSTATE **42**xxx denotan la existencia de diversos problemas de sintaxis o de acceso en la sentencia. xxx representa cualquier SQLSTATE con ese código de clase. Ejemplo: **42**xxx representa cualquier SQLSTATE perteneciente a la clase **42**.
58004	SQLBindCol	Error inesperado del sistema.	Error no recuperable del sistema.
58004	SQLBindParameter	Error inesperado del sistema.	Error no recuperable del sistema.
58004	SQLConnect	Error inesperado del sistema.	Error no recuperable del sistema.
58004	SQLDescribeCol	Error inesperado del sistema.	Error no recuperable del sistema.
58004	SQLDisconnect	Error inesperado del sistema.	Error no recuperable del sistema.

Tabla 27. Mensajes de SQLState notificados por CLI (continuación)

SQLSTATE	Nombre de la función de CLI	Descripción	Explicación
58004	SQLExecDirect	Error inesperado del sistema.	Error no recuperable del sistema.
58004	SQLFetch	Error inesperado del sistema.	Error no recuperable del sistema.
58004	SQLFreeStmt	Error inesperado del sistema.	Error no recuperable del sistema.
58004	SQLGetData	Error inesperado del sistema.	Error no recuperable del sistema.
58004	SQLPrepare	Error inesperado del sistema.	Error no recuperable del sistema.
58004	SQLNumResultCols	Error inesperado del sistema.	Error no recuperable del sistema.
58004	SQLRowCount	Error inesperado del sistema.	Error no recuperable del sistema.
59101*	SQLExecute	Usuario no definido.	El usuario no está definido en la base de datos de control del Centro de administración de dispositivos portátiles.
59102*	SQLExecute	Contraseña no correcta.	La contraseña del usuario no coincide con la contraseña definida en el Centro de administración de dispositivos portátiles.
59103*	SQLExecute	Grupo no definido.	El grupo no está definido en el Centro de administración de dispositivos portátiles.
59104*	SQLExecute	Aplicación no definida.	La aplicación no está definida en el Centro de administración de dispositivos portátiles.
59105*	SQLExecute	Suscripción no definida.	La suscripción con AgentAdapter no está definida en el Centro de administración de dispositivos portátiles.
59106*	SQLExecute	Suscripción no completada.	La suscripción no dispone de toda la información necesaria para invocar un procedimiento almacenado remoto.
59120*	SQLExecute	Error de conversión XML.	AgentAdapter ha fallado al convertir los datos de entrada del usuario en un documento XML.
59121*	SQLExecute	Se ha producido un error general de AgentAdapter.	Se ha producido un error general de AgentAdapter.
59122*	SQLExecute	Ha fallado la carga de la biblioteca.	No se pueden encontrar en el sistema algunas de las bibliotecas necesarias.
HY000	SQLAllocHandle	Error general.	Se ha producido un error para el que no existe ningún SQLSTATE específico. El mensaje de error devuelto por `SQLGetDiagRec()` en el almacenamiento intermedio **MessageText* describe el error y su causa.
HY000	SQLFreeHandle	Error general.	Se ha producido un error para el que no existe ningún SQLSTATE específico. El mensaje de error devuelto por `SQLGetDiagRec()` en el almacenamiento intermedio **MessageText* describe el error y su causa.

Tabla 27. Mensajes de SQLState notificados por CLI (continuación)

SQLSTATE	Nombre de la función de CLI	Descripción	Explicación
HY001	SQLAllocHandle	Error de asignación de memoria.	DB2 CLI no puede asignar memoria para el descriptor de contexto especificado.
HY001	SQLBindCol	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY001	SQLBindParameter	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY001	SQLConnect	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY001	SQLDescribeCol	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY001	SQLDisconnect	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY001	SQLExecDirect	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY001	SQLFetch	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY001	SQLFreeHandle	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY001	SQLFreeStmt	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY001	SQLGetData	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY001	SQLPrepare	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY001	SQLNumResultCols	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY001	SQLRowCount	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY002	SQLBindCol	Número de columna no válido.	El valor especificado para el argumento *ColumnNumber* es menor que 0. El valor especificado para el argumento *ColumnNumber* superaba el número máximo de columnas soportadas por la fuente de datos.
HY002	SQLDescribeCol	Número de columna no válido.	El valor especificado para el argumento *ColumnNumber* es menor que 1. El valor especificado para el argumento *ColumnNumber* es mayor que el número de columnas del conjunto resultante.
HY002	SQLGetData	Número de columna no válido.	La columna especificada es menor que 0 o mayor que el número de columnas resultantes.
HY003	SQLBindCol	Tipo de programa fuera de rango.	*TargetType* no es un tipo de datos válido ni SQL_C_DEFAULT.
HY003	SQLBindParameter	Tipo de programa fuera de rango.	El valor especificado por el argumento *ParameterNumber* no es un tipo de datos válido ni SQL_C_DEFAULT.
HY003	SQLGetData	Tipo de programa fuera de rango.	*TargetType* no es un tipo de datos válido ni SQL_C_DEFAULT.

Tabla 27. Mensajes de SQLState notificados por CLI (continuación)

SQLSTATE	Nombre de la función de CLI	Descripción	Explicación
HY004	SQLBindParameter	Tipo de datos SQL fuera de rango.	El valor especificado para el argumento *ParameterType* no es un tipo válido de datos SQL.
HY009	SQLBindParameter	Valor no válido de argumento.	El argumento *ParameterValuePtr* es un puntero nulo y el argumento *StrLen_or_IndPtr* es un puntero nulo e *InputOutputType* no es SQL_PARAM_OUTPUT.
HY009	SQLExecDirect	Valor no válido de argumento.	*StatementText* es un puntero nulo.
HY009	SQLNumResultCols	Valor no válido de argumento.	*StatementText* es un puntero nulo.
HY010	SQLDescribeCol	Error de secuencia de función.	Se llama a la función antes de llamar a `SQLPrepare()` o `SQLExecDirect()` para el descriptor de contexto de sentencia (*StatementHandle*).
HY010	SQLExecute	Error de secuencia de función.	El descriptor de contexto de sentencia (*StatementHandle*) especificado no está en estado preparado. Se llama a `SQLExecute()` sin antes llamar a `SQLPrepare()`.
HY010	SQLFetch	Error de secuencia de función.	Se llama a la función antes de llamar a `SQLPrepare()` o `SQLExecDirect()` para el descriptor de contexto de sentencia (*StatementHandle*).
HY010	SQLFreeHandle	Error de secuencia de función.	El argumento *HandleType* es SQL_HANDLE_ENV y el estado de al menos una conexión es asignado o conectado. Antes de llamar a `SQLFreeHandle()` con un *HandleType* de SQL_HANDLE_ENV, se debe llamar para cada conexión a `SQLDisconnect()` y `SQLFreeHandle()` con un *HandleType* de SQL_HANDLE_DBC. El argumento *HandleType* es SQL_HANDLE_DBC y se llama a la función antes de llamar a `SQLDisconnect()` para la conexión. El argumento *HandleType* es SQL_HANDLE_STMT; se llamó a `SQLExecute()` o `SQLExecDirect()` con el descriptor de contexto de sentencia y la función devolvió SQL_NEED_DATA. (DM) Todos los descriptores de contexto auxiliares y otros recursos no se liberaron antes de llamar a `SQLFreeHandle()`.
HY010	SQLGetData	Error de secuencia de función.	Se llama a la función sin antes llamar a `SQLFetch()`.
HY010	SQLNumResultCols	Error de secuencia de función.	Se llama a la función antes de llamar a `SQLPrepare()` o `SQLExecDirect()` para el descriptor de contexto de sentencia (*StatementHandle*).
HY010	SQLRowCount	Error de secuencia de función.	Se llama a la función antes de llamar a `SQLExecute()` o `SQLExecDirect()` para el descriptor de contexto de sentencia (*StatementHandle*).
HY013	SQLAllocHandle	Error inesperado de gestión de la memoria.	El argumento *HandleType* es SQL_HANDLE_DBC, o SQL_HANDLE_STMT; y no se pudo procesar la llamada a la función debido a que no se pudo acceder a los objetos de memoria subyacentes, posiblemente debido a una condición de falta de memoria.
HY013	SQLBindCol	Error inesperado de gestión de la memoria.	DB2 CLI no puede acceder a la memoria necesaria para ejecutar o finalizar la función.

Tabla 27. Mensajes de SQLState notificados por CLI (continuación)

SQLSTATE	Nombre de la función de CLI	Descripción	Explicación
HY013	SQLBindParameter	Error inesperado de gestión de la memoria.	DB2 CLI no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY013	SQLConnect	Error inesperado de gestión de la memoria.	DB2 CLI no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY013	SQLDescribeCol	Error inesperado de gestión de la memoria.	DB2 CLI no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY013	SQLDisconnect	Error inesperado de gestión de la memoria.	DB2 CLI no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY013	SQLExecDirect	Error inesperado de gestión de la memoria.	DB2 CLI no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY013	SQLFetch	Error inesperado de gestión de la memoria.	DB2 CLI no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY013	SQLFreeHandle	Error inesperado de gestión de la memoria.	El argumento *HandleType* es SQL_HANDLE_STMT y no se puede procesar la llamada a la función debido a que no se puede acceder a los objetos de memoria asociados, posiblemente debido a condiciones de falta de memoria.
HY013	SQLGetData	Error inesperado de gestión de la memoria.	DB2 CLI no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY013	SQLNumResultCols	Error inesperado de gestión de la memoria.	DB2 CLI no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY013	SQLNumResultCols	Error inesperado de gestión de la memoria.	DB2 CLI no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY013	SQLRowCount	Error inesperado de gestión de la memoria.	DB2 CLI no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY014	SQLAllocHandle	No hay más descriptores de contexto.	Se ha llegado al límite en el número de descriptores de contexto que se pueden asignar para el tipo de descriptor de contexto indicado por el argumento *HandleType*.
HY014	SQLExecDirect	No hay más descriptores de contexto.	DB2 CLI no puede asignar un descriptor de contexto debido a los recursos internos.
HY014	SQLNumResultCols	No hay más descriptores de contexto.	DB2 CLI no puede asignar un descriptor de contexto debido a los recursos internos.
HY017	SQLFreeHandle	Utilización no válida de un descriptor de contexto de descriptor asignado automáticamente.	El argumento *Handle* se establece en el descriptor de contexto para un descriptor asignado automáticamente o para un descriptor de implementación.
HY024	SQLSetStmtAttr	Valor no válido de atributo.	Dado el valor de *Attribute* especificado, se ha especificado un valor no válido en *ValuePtr*.
HY090	SQLBindCol	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El valor especificado para el argumento *BufferLength* es menor que 1 y el argumento *TargetType* es SQL_C_CHAR, SQL_C_BINARY o SQL_C_DEFAULT.
HY090	SQLBindParameter	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El valor especificado para el argumento *BufferLength* era menor que 0.

Tabla 27. Mensajes de SQLState notificados por CLI (continuación)

SQLSTATE	Nombre de la función de CLI	Descripción	Explicación
HY090	SQLDescribeCol	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	La longitud especificada en el argumento *BufferLength* es menor que 1.
HY090	SQLExecDirect	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El argumento *TextLength* es menor que 1, pero no es igual a SQL_NTS.
HY090	SQLGetData	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El valor del argumento *BufferLength* es menor que 0 y el argumento *TargetType* es SQL_C_CHAR o SQL_C_BINARY, o bien *TargetType* es SQL_C_DEFAULT y el tipo por omisión es SQL_C_CHAR, SQL_C_BINARY o SQL_C_DBCHAR.
HY090	SQLNumResultCols	Longitud no válida de la serie de caracteres o del almacenamiento intermedio	El argumento *TextLength* es menor que 1 pero no igual a SQL_NTS.
HY092	SQLAllocHandle	Tipo de opción fuera de rango.	El argumento *HandleType* no es: SQL_HANDLE_ENV SQL_HANDLE_DBC SQL_HANDLE_STMT
HY092	SQLFreeStmt	Tipo de opción fuera de rango.	El valor especificado para el argumento *Option* no es SQL_DROP ni SQL_RESET_PARAMS.
HY093	SQLBindParameter	Número de parámetros no válido.	El valor especificado para el argumento *ValueType* es menor que 1 o mayor que el número máximo de parámetros soportados por el servidor.
HY094	SQLBindParameter	Valor no válido de escala.	El valor especificado para *ParameterType* es SQL_DECIMAL o SQL_NUMERIC y el valor especificado para *DecimalDigits* es menor que 0 o mayor que el valor del argumento *ParamDef* (precisión).
HY104	SQLBindParameter	Valor no válido de precisión.	El valor especificado para *ParameterType* es SQL_DECIMAL o SQL_NUMERIC y el valor especificado para *ParamDef* es menor que 1.
HY105	SQLBindParameter	Tipo de parámetro no válido.	*InputOutputType* no es SQL_PARAM_INPUT.
HY106	SQLFetchScroll	Tipo de recuperación fuera de rango.	El valor especificado para el argumento *FetchOrientation* no es válido. El valor del atributo de sentencia SQL_CURSOR_TYPE es SQL_CURSOR_FORWARD_ONLY y el valor del argumento *FetchOrientation* no es SQL_FETCH_NEXT.
HY107	SQLFetchScroll	Valor de fila fuera de rango.	El valor especificado con el atributo de sentencia de SQL_ATTR_CURSOR_TYPE es SQL_CURSOR_KEYSET_DRIVEN, pero el valor especificado con el atributo de sentencia de SQL_ATTR_KEYSET_SIZE es mayor que 0 y menor que el valor especificado con el atributo de sentencia de SQL_ATTR_ROW_ARRAY_SIZE.

Tabla 27. Mensajes de SQLState notificados por CLI (continuación)

SQLSTATE	Nombre de la función de CLI	Descripción	Explicación
HY501	SQLConnect	Nombre no válido de la *DataSource*.	El nombre de *DataSource* especificado no es válido.
HYC00	SQLBindCol	Controlador no apropiado.	DB2 CLI reconoce, pero no da soporte al tipo de datos especificado en el argumento *TargetType*.
HYC00	SQLBindParameter	Controlador no apropiado.	DB2 CLI o la fuente de datos no dan soporte a la conversión especificada por la combinación del valor especificado para el argumento *ValueType* y el valor especificado para el argumento *ParameterType*. El valor especificado para el argumento *ParameterType* no está soportado por DB2 CLI o por la fuente de datos.
HYC00	SQLDescribeCol	Controlador no apropiado.	DB2 CLI no reconoce el tipo de datos SQL de la columna *ColumnNumber*.
HYC00	SQLGetData	Controlador no apropiado.	DB2 CLI reconoce, pero no da soporte a, el tipo de datos SQL especificado. DB2 CLI o la fuente de datos no puede convertir el tipo de datos SQL al tipo *TargetType* de los datos de la aplicación.

Consulta relacionada:

- "Visión general del soporte de sentencia de SQL de DB2 Everyplace" en la página 137
- "Compatibilidad entre tipos de datos para las operaciones de asignación y comparación" en la página 176
- "Listado de los SQLSTATE" en la página 180
- "Marcadores de parámetro soportados de DB2 Everyplace" en la página 76
- "Resumen de códigos de clase de SQLState" en la página 180

Mensajes de SQLState notificados por JDBC

Tabla 28. Mensajes de SQLState notificados por JDBC

SQLSTATE	Descripción	Explicación
0100C	Se ha devuelto uno o más conjuntos de resultados ad hoc.	DB2 Everyplace no da soporte a ResultSet.CONCUR_UPDATABLE para la modalidad de simultaneidad de un objeto *ResultSet*. En su lugar se utiliza ResultSet.CONCUR_READ_ONLY.
0641E	Hay una sentencia SELECT en el proceso por lotes.	No se admite una sentencia SELECT en el proceso por lotes.
0643E	No hay ninguna sentencia en el proceso por lotes.	El proceso por lotes no tiene ninguna sentencia.
22005	Error de asignación.	Un tipo de parámetro es incompatible con el tipo de datos destino.
22011	Se ha producido un error de subserie.	Posición ordinal no válida para el primer byte del valor BLOB que ha de extraerse.

Tabla 28. Mensajes de SQLState notificados por JDBC (continuación)

SQLSTATE	Descripción	Explicación
S1010	Error de secuencia de función.	El método get `CallableStatement` se llama sin llamar en primer lugar a `registerOutParameter`.

Consulta relacionada:

- "Visión general del soporte de JDBC de DB2 Everyplace" en la página 287

Funciones de CLI de DB2 soportadas

Este capítulo describe las funciones de la Interfaz de nivel de llamada (CLI de DB2) de DB2 soportadas por DB2 Everyplace.

- "Resumen de las funciones de CLI de DB2" proporciona una breve descripción de la finalidad de cada función y un breve resumen de las diferencias entre las funciones de CLI de DB2 soportadas por DB2 Everyplace y las funciones de CLI de DB2 estándar.
- "Clave para las descripciones de funciones de CLI de DB2" en la página 198 proporciona una explicación de las descripciones de función para cada función CLI.
- "Conversión de datos por funciones de CLI de DB2" en la página 285 contiene una tabla que lista las conversiones de datos soportadas entre tipos de datos C y SQL.

Resumen de las funciones de CLI de DB2

La Tabla 29 resume la finalidad de cada función de CLI de DB2 soportada por DB2 Everyplace y las diferencias entre las funciones de CLI de DB2 soportadas por DB2 Everyplace y las funciones de CLI de DB2 estándar.

Tabla 29. lista de funciones de CLI de DB2

Nombre de la función	Finalidad	Resumen de las diferencias
SQLAllocConnect	Obtiene un descriptor de contexto de conexión.	
SQLAllocEnv	Obtiene un descriptor de entorno.	
SQLAllocHandle	Obtiene un descriptor de contexto.	
SQLAllocStmt	Asigna un descriptor de sentencia.	
SQLBindCol	Asigna almacenamiento para una columna resultante y especifica el tipo de datos.	El tipo de datos destino sólo puede ser un tipo de datos soportado. No se da soporte al localizador de LOB.

Tabla 29. lista de funciones de CLI de DB2 (continuación)

Nombre de la función	Finalidad	Resumen de las diferencias
SQLBindParameter	Asigna almacenamiento para un parámetro de una sentencia de SQL.	No da soporte al enlace con matrices de variables de aplicación o localizadores de LOB. No da soporte a SQLPutData(), por lo que la aplicación debe colocar el valor del parámetro en *ParameterValuePtr* antes de invocar SQLExecute(). El tipo de parámetro sólo puede ser INPUT, pues no se da soporte a los procedimientos almacenados.
SQLColumns	Devuelve la lista de nombres de columna en las tablas especificadas.	No se tienen en cuenta los parámetros *CatalogName*, *NameLength1*, *SchemaName*, *NameLength2*. Las columnas 2, 12 y 15 del conjunto de resultados devuelto son siempre NULL. No se da soporte al código de retorno SQL_STILL_EXECUTING.
SQLConnect	Conecta a un controlador específico mediante un nombre de fuente de datos, un ID de usuario y una contraseña.	
SQLDescribeCol	Describe una columna del conjunto resultante.	La información sobre columnas está limitada por los tipos soportados de datos de columnas.
SQLDisconnect	Cierra la conexión.	
SQLEndTran	Solicita un COMMIT o un ROLLBACK para todas las operaciones de todas las sentencias asociadas a una conexión.	El atributo de conexión SQL_ATTR_AUTOCOMMIT debe establecerse en SQL_AUTOCOMMIT_OFF antes de invocar SQLEndTran().
SQLError	Devuelve información adicional sobre errores o de estado.	
SQLExecDirect	Ejecuta una sentencia.	No se da soporte a los códigos de retorno SQL_STILL_EXECUTING ni SQL_NEED_DATA. No se da soporte a las llamadas de CLI asíncronas.
SQLExecute	Ejecuta una sentencia preparada.	Se deben enlazar todos los parámetros antes de invocar SQLExecute(). No se da soporte a la ejecución asíncrona de las llamadas de SQL.

Tabla 29. lista de funciones de CLI de DB2 (continuación)

Nombre de la función	Finalidad	Resumen de las diferencias
SQLFetch	Devuelve una fila resultante.	El resultado se obtiene fila a fila, no por conjuntos de filas. No se da soporte a los descriptores de sentencias. No se da soporte al código de retorno SQL_STILL_EXECUTING.
SQLFetchScroll	Devuelve un conjunto de filas resultante.	El resultado se obtiene por conjuntos de filas. No se da soporte al código de retorno SQL_STILL_EXECUTING.
SQLForeignKeys	Devuelve información sobre claves foráneas para la tabla especificada.	*PKCatalogName*, *NameLength1*, *PKSchemaName*, *NameLength2*, *FKCatalogName*, *NameLength4*, *FKSchemaName* y *NameLength5* se ignoran. Las columnas 1, 2, 5, 6, 12 y 13 del conjunto resultante devuelto son siempre una serie de longitud cero. Las columnas 10, 11 y 14 del conjunto resultante devuelto son siempre cero. No se da soporte al código de retorno SQL_STILL_EXECUTING.
SQLFreeConnect	Libera el descriptor de contexto de conexión.	
SQLFreeEnv	Libera el descriptor de entorno.	
SQLFreeHandle	Libera recursos de descriptor de contexto.	
SQLFreeStmt	Finaliza el proceso de la sentencia, desecha los resultados pendientes y, opcionalmente, libera todos los recursos asociados al descriptor de la sentencia.	Sólo se da soporte a las opciones SQL_DROP y SQL_RESET_PARAMS.
SQLGetConnectAttr	Devuelve el valor actual de un atributo de conexión.	DB2 Everyplace da soporte a un subconjunto de atributos de conexión soportados por DB2. DB2 Everyplace también da soporte a algunos atributos de conexión no soportados por DB2.
SQLGetCursorName	Devuelve el nombre de cursor que está asociado a un descriptor de sentencia.	El nombre de cursor generado internamente siempre comienza por CUR.

Tabla 29. lista de funciones de CLI de DB2 (continuación)

Nombre de la función	Finalidad	Resumen de las diferencias
SQLGetData	Devuelve parte o la totalidad de una columna de una fila de un conjunto resultante.	El tipo de datos destino sólo puede ser un tipo de datos soportado. No se da soporte al localizador de LOB. No se da soporte al código de retorno SQL_STILL_EXECUTING.
SQLGetDiagRec	Obtiene varios campos de datos de diagnóstico.	Sólo se da soporte a los registros de diagnóstico asociados a un descriptor de contexto de una sentencia o conexión. Sólo se da soporte a registros de diagnóstico individuales.
SQLGetInfo	Devuelve información acerca de una fuente de datos y un controlador específicos.	DB2 Everyplace da soporte a un subconjunto de los tipos de información soportados por DB2.
SQLGetStmtAttr	Devuelve el valor actual de un atributo de una sentencia.	DB2 Everyplace da soporte a un subconjunto de atributos de sentencia soportados por DB2. DB2 Everyplace también da soporte a algunos atributos de sentencia no soportados por DB2.
SQLNumParams	Devuelve el número de marcadores de parámetro de una sentencia de SQL.	No se da soporte al código de retorno SQL_STILL_EXECUTING.
SQLNumResultCols	Devuelve el número de columnas del conjunto resultante.	
SQLPrepare	Prepara una sentencia de SQL para una ejecución posterior.	
SQLPrimaryKeys	Devuelve una lista de nombres de columna que abarcan la clave primaria de una tabla.	No se tienen en cuenta los parámetros *CatalogName*, *NameLength1*, *SchemaName*, *NameLength2*. Las columnas 1, 2 y 6 del conjunto resultante devuelto son siempre una serie de longitud cero. No se da soporte al código de retorno SQL_STILL_EXECUTING.
SQLRowCount	Devuelve el número de filas afectadas por una petición de inserción, actualización o supresión.	

Tabla 29. lista de funciones de CLI de DB2 (continuación)

Nombre de la función	Finalidad	Resumen de las diferencias
SQLSetConnectAttr	Establece opciones relacionadas con una conexión.	DB2 Everyplace da soporte a un subconjunto de atributos de conexión soportados por DB2. DB2 Everyplace también da soporte a algunos atributos de conexión no soportados por DB2.
SQLSetStmtAttr	Establece opciones relacionadas con una sentencia.	DB2 Everyplace da soporte a un subconjunto de atributos de sentencia soportados por DB2. DB2 Everyplace también da soporte a algunos atributos de sentencia no soportados por DB2.
SQLTables	Devuelve la lista de nombres de tabla almacenados en una fuente de datos específica.	No se tienen en cuenta los parámetros *CatalogName*, *NameLength1*, *SchemaName*, *NameLength2*, *TableType*, *NameLength4*. DB2 Everyplace sólo da soporte al tipo "TABLE." No se da soporte al código de retorno SQL_STILL_EXECUTING.

Consulta relacionada:

- "Conversión de datos por funciones de CLI de DB2" en la página 285
- "Clave para las descripciones de funciones de CLI de DB2"

Clave para las descripciones de funciones de CLI de DB2

Cada descripción de función contiene las secciones siguientes:

Finalidad

Esta sección proporciona una breve visión general de lo que realiza la función. También indica si debe invocarse alguna función antes y después de invocar la función que se está describiendo.

Cada función tiene también una tabla que indica qué especificación o norma cumple la función.

Esta tabla indica el soporte de la función. Algunas funciones utilizan un conjunto de opciones que no son aplicables a todas las especificaciones o normas. Las diferencias significativas están indicadas en la sección sobre restricciones de la función.

Sintaxis

Esta sección contiene el prototipo 'C' genérico. El prototipo genérico se utiliza para todos los entornos, incluido Windows.

Todos los argumentos de función que son punteros se definen utilizando la macro `FAR`; esta macro se inhabilita (se establece en un espacio en blanco) para todas las plataformas excepto Windows. En Windows, `FAR` se utiliza para definir argumentos como punteros remotos.

Argumentos
Esta sección lista cada argumento de la función, junto con su tipo de datos, una descripción y la indicación de si es un argumento de entrada o de salida.

Algunas funciones contienen argumentos de entrada o salida, que se denominan argumentos *diferidos* o *enlazados*.

Estos argumentos son punteros que apuntan a almacenamientos intermedios asignados por la aplicación y están asociados con (o enlazados a) un parámetro de una sentencia de SQL o una columna de un conjunto resultante. CLI de DB2 accede a las áreas de datos que especifica la función en un momento posterior. Estas áreas de datos diferidas deben ser válidas todavía cuando CLI de DB2 acceda a ellas.

Uso Esta sección proporciona información sobre cómo utilizar la función y cualquier consideración especial aplicable. Las posibles condiciones de error no se describen aquí, sino que se incluyen en la sección sobre diagnósticos.

Códigos de retorno
Esta sección lista todos los posibles códigos de retorno de la función. Cuando se devuelve SQL_ERROR o SQL_SUCCESS_WITH_INFO, se puede obtener información sobre errores invocando `SQLError()` o `SQLGetDiagRec()`.

Diagnósticos
Esta sección contiene una tabla que lista los SQLSTATE devueltos explícitamente por CLI de DB2 (también pueden obtenerse los SQLSTATE generados por DBMS) e indica la causa del error. Estos valores se obtienen invocando `SQLError()` o `SQLGetDiagRec()` después de que la función devuelva un SQL_ERROR o SQL_SUCCESS_WITH_INFO.

Restricciones
Esta sección indica las diferencias o limitaciones existentes entre CLI y ODBC de DB2 Everyplace que podrían afectar a una aplicación.

Consulte el manual *IBM DB2 Universal Database Call Level Interface Guide and Reference* para obtener más información sobre CLI de DB2, tal como información sobre los códigos de retorno, diagnósticos, ejemplos, configuración del entorno CLI y el acceso a las aplicaciones de ejemplo.

Consulta relacionada:

- "Conversión de datos por funciones de CLI de DB2" en la página 285
- "Resumen de las funciones de CLI de DB2" en la página 194

SQLAllocConnect—Asignar descriptor de conexión

En ODBC Versión 3, `SQLAllocConnect()` se desaprobó y se sustituyó por `SQLAllocHandle()`; para obtener más información, vea "SQLAllocHandle—Asignar descriptor de contexto" en la página 200.

Recomendación: Aunque la versión actual de CLI de DB2 sigue dando soporte a `SQLAllocConnect()`, utilice `SQLAllocHandle()` en sus programas CLI de DB2 para que se ajusten a las normas más recientes.

Migración a la nueva función

Por ejemplo, la sentencia:

```
SQLAllocConnect(henv, hdbc);
```

se escribiría así utilizando la nueva función:

```
SQLAllocHandle(SQL_HANDLE_DBC, henv, hdbc);
```

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194

SQLAllocEnv—Asignar descriptor de entorno

En ODBC versión 3, SQLAllocEnv() se desaprobó y se sustituyó por SQLAllocHandle(); vea "SQLAllocHandle—Asignar descriptor de contexto" para obtener más información.

Recomendación: Aunque la versión actual de CLI de DB2 sigue dando soporte a SQLAllocEnv(), utilice SQLAllocHandle() en sus programas CLI de DB2 para que se ajusten a las normas más recientes.

Migración a la nueva función

Por ejemplo, la sentencia:

```
SQLAllocEnv(henv);
```

se escribiría así utilizando la nueva función:

```
SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, henv);
```

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194

SQLAllocHandle—Asignar descriptor de contexto

Finalidad:

Especificación:	CLI de DB2 5.0	ODBC 3.0	ISO CLI

SQLAllocHandle() asigna descriptores de contexto de entorno, de conexión o de sentencia.

Esta función es una función genérica para asignar descriptores de contexto que sustituye a las funciones de la versión 2 desaprobadas, SQLAllocConnect(), SQLAllocEnv() y SQLAllocStmt().

Sintaxis:

```
SQLRETURN   SQLAllocHandle   (SQLSMALLINT       HandleType,
                              SQLHANDLE         InputHandle,
                              SQLHANDLE         *OutputHandlePtr);
```

Argumentos de la función:

Tabla 30. Argumentos de SQLAllocHandle

Tipo de datos	Argumento	Uso	Descripción
SQLSMALLINT	*HandleType*	entrada	El tipo de descriptor de contexto que debe ser asignado por `SQLAllocHandle()`. Debe ser uno de los valores siguientes: SQL_HANDLE_ENV SQL_HANDLE_DBC SQL_HANDLE_STMT
SQLHANDLE	*InputHandle*	entrada	Descriptor de contexto existente que debe utilizarse como contexto para el nuevo descriptor de contexto que se está asignando. Si *HandleType* es SQL_HANDLE_ENV, este valor es SQL_NULL_HANDLE. Si *HandleType* es SQL_HANDLE_DBC, este valor debe ser un descriptor de entorno; y si es SQL_HANDLE_STMT, debe ser un descriptor de contexto de conexión.
SQLHANDLE	*OutputHandlePtr*	salida	Puntero que apunta a un almacenamiento intermedio en el que devolver el descriptor de contexto a la estructura de datos recién asignada.

Uso:

`SQLAllocHandle()` se utiliza para asignar descriptores de contexto de entorno, de conexión y de sentencia, tal como se describe a continuación.

Una aplicación puede asignar al mismo tiempo varios descriptores de contexto de sentencia.

Si la aplicación invoca `SQLAllocHandle()` con **OutputHandlePtr* establecido en un descriptor de entorno, de conexión o de sentencia que ya existe, CLI de DB2 sobreescribe la información asociada al descriptor de contexto. CLI de DB2 no comprueba si el descriptor de contexto entrado en **OutputHandlePtr* ya está en uso, ni comprueba el contenido anterior de un descriptor de contexto antes de sobreescribirlo.

Para DB2 Everyplace, todos los descriptores de contexto, excepto el descriptor de sentencia, son descriptores de contexto ficticios y no llevan información utilizable.

El descriptor de sentencia proporciona acceso a información sobre una sentencia, tal como mensajes de error, e información de estado para el proceso de sentencias de SQL. Para solicitar un descriptor de sentencia, la aplicación se conecta a una fuente de datos y luego invoca `SQLAllocHandle()` antes de someter sentencias de SQL. En esta llamada, *HandleType* se debe establecer en SQL_HANDLE_STMT y *InputHandle* se debe establecer en el descriptor de conexión devuelto por la llamada a `SQLAllocHandle()` que hizo que se asignara ese descriptor de contexto. CLI de DB2 asigna el descriptor de sentencia, asocia el descriptor de sentencia a la conexión especificada y devuelve en **OutputHandlePtr* el valor del descriptor de contexto asociado. La aplicación pasa el valor de **OutputHandlePtr* en todas las llamadas subsiguientes que necesiten un descriptor de sentencia.

Cuando una aplicación finaliza, se liberan todos los recursos de DB2 Everyplace asignados para la aplicación, por lo que los descriptores de contexto utilizados por la aplicación dejan de ser válidos.

Para DB2 Everyplace, no existe ningún descriptor asociado a un descriptor de sentencia con atributos que puedan ser cambiados por una aplicación.

Cuando se esté utilizando DB2 Everyplace con Visual Basic y la interfaz CLI/ODBC de DB2 Everyplace, debe llamar de manera explícita las funciones de CLI correlacionadas/subyacentes definidas en sqlcli.h. Por ejemplo, una llamada a `SQLAllocHandle()` falla. Pero la llamada a `SQLAllocHandleVer(SQL_HANDLE_STMT, hdbc, hstmt, DB2eVersion)` será satisfactoria.

Códigos de retorno:

- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_INVALID_HANDLE
- SQL_ERROR

Cuando se asigna un descriptor de contexto que no sea un descriptor de contexto de entorno, si `SQLAllocHandle()` devuelve SQL_ERROR, establece *OutputHandlePtr* en SQL_NULL_HENV, SQL_NULL_HDBC o SQL_NULL_HSTMT, en función del valor de *HandleType*, a menos que el argumento de salida sea un puntero nulo. La aplicación puede entonces obtener información adicional a partir de la estructura de datos de diagnóstico asociada al descriptor de contexto en el argumento *InputHandle*.

Diagnósticos:

Tabla 31. SQLSTATE de SQLAllocHandle

SQLSTATE	Descripción	Explicación
01000	Aviso.	Mensaje informativo. (La función devuelve SQL_SUCCESS_WITH_INFO).
08003	La conexión está cerrada.	El argumento *HandleType* es SQL_HANDLE_STMT, pero la conexión especificada por el argumento *InputHandle* no está abierta. El proceso de conexión debe finalizar satisfactoriamente (y se debe abrir la conexión) para que CLI de DB2 asigne un descriptor de sentencia.
HY000	Error general.	Se ha producido un error para el que no existe ningún SQLSTATE específico. El mensaje de error devuelto por `SQLGetDiagRec()` en el almacenamiento intermedio **MessageText* describe el error y su causa.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar memoria para el descriptor de contexto especificado.
HY013	Error inesperado de gestión de la memoria.	El argumento *HandleType* es SQL_HANDLE_DBC, o SQL_HANDLE_STMT; y no se pudo procesar la llamada a la función debido a que no se pudo acceder a los objetos de memoria subyacentes, posiblemente debido a una falta de memoria.
HY014	No hay más descriptores de contexto.	Se ha llegado al límite en el número de descriptores de contexto que se pueden asignar para el tipo de descriptor de contexto indicado por el argumento *HandleType*.

Tabla 31. SQLSTATE de SQLAllocHandle (continuación)

SQLSTATE	Descripción	Explicación
HY092	Tipo de opción fuera de rango.	El argumento *HandleType* no es: SQL_HANDLE_ENV SQL_HANDLE_DBC SQL_HANDLE_STMT

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLExecDirect—Ejecutar una sentencia directamente" en la página 225
- "SQLFreeHandle—Liberar recursos de descriptor de contexto" en la página 242

SQLAllocStmt—Asignar un descriptor de sentencia

En ODBC Versión 3, se desaprobó `SQLAllocStmt()` y se sustituyó por `SQLAllocHandle()`; para obtener más información, vea "SQLAllocHandle—Asignar descriptor de contexto" en la página 200.

Recomendación: Aunque la versión actual de CLI de DB2 sigue dando soporte a `SQLAllocStmt()`, utilice `SQLAllocHandle()` en sus programas CLI de DB2 para que se ajusten a las normas más recientes.

Migración a la nueva función

Por ejemplo, la sentencia:

```
SQLAllocStmt(hdbc, hstmt);
```

se escribiría así utilizando la nueva función:

```
SQLAllocHandle(SQL_HANDLE_STMT, hdbc, hstmt);
```

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194

SQLBindCol—Enlazar una columna a una variable de aplicación

Finalidad:

Especificación:	CLI de DB2 1.1	ODBC 1.0	ISO CLI

`SQLBindCol()` se utiliza para asociar (enlazar) columnas de un conjunto resultante con variables de aplicación, para todos los tipos de datos C. Se transfieren datos del DBMS a la aplicación cuando se invoca `SQLFetch()`. Durante la transferencia de datos puede producirse una conversión de los datos.

`SQLBindCol()` se invoca una vez para cada columna del conjunto resultante que la aplicación necesite recuperar.

En general, SQLPrepare() o SQLExecDirect() se invocan antes que esta función y SQLFetch() se invoca después. Pueden también ser necesarios atributos de columnas antes de invocar SQLBindCol() y se pueden obtener utilizando SQLDescribeCol().

Sintaxis:

```
SQLRETURN   SQLBindCol       (SQLHSTMT
StatementHandle,   /* hstmt */
                SQLUSMALLINT      ColumnNumber,      /* icol */
                SQLSMALLINT       TargetType,        /* fCType */
                SQLPOINTER        TargetValuePtr,    /* rgbValue */
                SQLINTEGER        BufferLength,      /* cbValueMax */
                SQLINTEGER   *FAR StrLen_or_IndPtr); /* pcbValue */
```

Argumentos de la función:

Tabla 32. Argumentos de SQLBindCol

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia
SQLUSMALLINT	*ColumnNumber*	entrada	Número identificador de la columna. Las columnas están numeradas secuencialmente, de izquierda a derecha. Los números de columna comienzan en el 1.
SQLSMALLINT	*TargetType*	entrada	El tipo de datos C del número de columna *ColumnNumber* en el conjunto resultante. Se da soporte a los tipos siguientes: SQL_C_BINARY SQL_C_BIT SQL_C_CHAR SQL_C_DOUBLE SQL_C_FLOAT SQL_C_LONG SQL_C_SHORT SQL_C_TYPE_DATE SQL_C_TYPE_TIME SQL_C_TYPE_TIMESTAMP SQL_C_TINYINT Si se especifica SQL_C_DEFAULT, los datos se transfieren a su tipo de datos C por omisión.
SQLPOINTER	*TargetValuePtr*	entrada/ salida (diferido)	Puntero que apunta al almacenamiento intermedio donde CLI de DB2 debe guardar los datos de columna cuando se obtienen los datos. Si *TargetValuePtr* es nulo, la columna no se enlaza.
SQLINTEGER	*BufferLength*	entrada	Tamaño, en bytes, del almacenamiento intermedio *TargetValuePtr* disponible para almacenar los datos de columna. Si *TargetType* denota una serie binaria o de caracteres o es SQL_C_DEFAULT, *BufferLength* debe ser mayor que 0 o se devuelve un error. En otro caso, este argumento no se tiene en cuenta.
SQLINTEGER *	*StrLen_or_IndPtr*	entrada/salida (diferido)	Puntero que apunta a un valor que indica el número de bytes que CLI de DB2 puede devolver en el almacenamiento intermedio *TargetValuePtr*. SQLFetch() devuelve SQL_NULL_DATA en este argumento si el valor de datos de la columna es nulo. También puede devolverse SQL_NO_LENGTH. Consulte la sección sobre uso para obtener más información.

Para esta función, *TargetValuePtr* y *StrLen_or_Ind* son salidas diferidas, es decir, las posiciones de memoria a las que apuntan estos punteros no se actualizan hasta que se obtiene una fila del conjunto resultante. En consecuencia, las posiciones referenciadas por estos punteros deben permanecer válidas hasta que se invoque `SQLFetch()`. Por ejemplo, si se invoca `SQLBindCol()` dentro de una función local, `SQLFetch()` se debe llamar desde dentro del mismo ámbito de la función o el almacenamiento intermedio *TargetValuePtr* debe estar asignado o declarado como estático o global.

Uso:

La aplicación llama a `SQLBindCol()` una vez para cada columna del conjunto resultante para la que la aplicación necesite recuperar datos. Los conjuntos resultantes se generan llamando a `SQLExecute()` o a `SQLExecDirect()`. Cuando se invoca `SQLFetch()`, los datos de cada una de estas columnas *enlazadas* se colocan en la ubicación asignada (proporcionada por los punteros *TargetValuePtr* y *StrLen_or_Ind*).

Las columnas se identifican mediante un número, que se asigna secuencialmente de izquierda a derecha. Los números de columna comienzan en el 1.

El número de columnas del conjunto resultante se puede determinar invocando `SQLNumResultCols()`.

La aplicación puede consultar los atributos de la columna (tales como el tipo de datos y la longitud) invocando primero `SQLDescribeCol()`. Luego, esta información se puede utilizar para asignar una ubicación de memoria, con el tipo de datos y tamaño apropiados, para indicar una conversión a otro tipo de datos.

Una aplicación puede elegir no enlazar todas las columnas, o incluso no enlazar ninguna columna. Los datos de cualquiera de las columnas también se pueden recuperar utilizando `SQLGetData()` una vez obtenidas las columnas enlazadas correspondientes a la fila actual.

En recuperaciones subsiguientes de datos, la aplicación puede cambiar el enlace de estas columnas o puede invocar `SQLBindCol()` para enlazar columnas previamente desenlazadas. El nuevo enlace no afecta a los datos ya capturados, se utiliza en la recuperación siguiente. Para desenlazar una columna individual, llame a `SQLBindCol()` con el puntero *TargetValuePtr* establecido en NULL. Para desenlazar todas las columnas, la aplicación debe invocar `SQLFreeStmt()`.

La aplicación debe asegurarse de que se asigna suficiente almacenamiento para los datos que deben recuperarse. Si el almacenamiento intermedio debe contener datos de longitud variable, la aplicación debe asignar suficiente almacenamiento para la longitud máxima de la columna enlazada, de lo contrario se puede producir un truncamiento de los datos. Si el almacenamiento intermedio debe contener datos de longitud fija, CLI de DB2 considera que el tamaño del almacenamiento intermedio es la longitud del tipo de datos C. Si se especifica conversión de datos, ello puede afectar al tamaño de almacenamiento necesario.

Si se produce truncamiento de la serie, se devuelve SQL_SUCCESS_WITH_INFO y *StrLen_or_IndPtr* se establece en el tamaño real de *TargetValuePtr* para devolverlo a la aplicación.

Códigos de retorno:

- SQL_SUCCESS

- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 33. SQLSTATE de SQLBindCol

SQLSTATE	Descripción	Explicación
07009	Índice descriptor no válido.	El valor especificado para el argumento *ColumnNumber* excede el número máximo de columnas del conjunto resultante.
40003 **08**S01	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
58004	Error inesperado del sistema.	Error no recuperable del sistema.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY002	Número de columna no válido.	El valor especificado para el argumento *ColumnNumber* es menor que 0. El valor especificado para el argumento *ColumnNumber* excedía el número máximo de columnas soportadas por la fuente de datos.
HY003	Tipo de programa fuera de rango.	*TargetType* no es un tipo de datos válido ni SQL_C_DEFAULT.
HY013	Error inesperado de gestión de la memoria.	CLI de DB2 no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY090	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El valor especificado para el argumento *BufferLength* es menor que 1 y el argumento *TargetType* es SQL_C_CHAR, SQL_C_BINARY o SQL_C_DEFAULT.
HYC00	Controlador no apropiado.	CLI de DB2 reconoce, pero no da soporte al tipo de datos especificado en el argumento *TargetType*.

Durante la recuperación de datos pueden presentarse otros mensajes de diagnóstico referentes a las columnas enlazadas.

Restricciones:

Los almacenamientos intermedios de salida deben estar alineados por palabras (igual). Muchos procesadores, como Motorola 68000, tienen reglas para la alineación por palabras y la aplicación debería alinear de manera apropiada el almacenamiento intermedio para los tipos de datos que no son caracteres.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194

SQLBindParameter—Enlazar un marcador de parámetro a un almacenamiento intermedio

Finalidad:

Especificación:	**CLI de DB2** 2.1	**ODBC** 2.0	

`SQLBindParameter()` se utiliza para asociar (enlazar) marcadores de parámetros de una sentencia de SQL con variables de aplicación, para todos los tipos de datos C. En este caso, se transfieren datos de la aplicación al DBMS cuando se invoca `SQLExecute()` o `SQLExecDirect()`. Durante la transferencia de datos puede producirse una conversión de los datos.

Sintaxis:

```
SQLRETURN  SQL_API SQLBindParameter(
                SQLHSTMT         StatementHandle,  /* hstmt */
                SQLUSMALLINT     ParameterNumber,  /* ipar */
                SQLSMALLINT      InputOutputType,  /* fParamType */
                SQLSMALLINT      ValueType,        /* fCType */
                SQLSMALLINT      ParameterType,    /* fSqlType */
                SQLUINTEGER      ColumnSize,       /* cbColDef */
                SQLSMALLINT      DecimalDigits,    /* ibScale */
                SQLPOINTER       ParameterValuePtr,/* rgbValue */
                SQLINTEGER       BufferLength,      /* cbValueMax */
                SQLINTEGER *FAR  StrLen_or_IndPtr);/* pcbValue */
```

Argumentos de la función:

Tabla 34. Argumentos de SQLBindParameter

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia.
SQLUSMALLINT	*ParameterNumber*	entrada	Número del marcador de parámetro, ordenado secuencialmente de izquierda a derecha y comenzando en 1.
SQLSMALLINT	*InputOutputType*	entrada	El tipo de parámetro. El tipo soportado es: • SQL_PARAM_INPUT: Cuando se ejecuta la sentencia, se envía al servidor el valor de datos real del parámetro; el almacenamiento intermedio *ParameterValuePtr* debe contener valores válidos de datos de entrada; el almacenamiento intermedio *StrLen_or_IndPtr* debe contener el correspondiente valor de longitud o SQL_NTS, o SQL_NULL_DATA. DB2 Everyplace no da soporte a `SQLPutData()`, por lo que no se debe colocar el valor del parámetro en el almacenamiento intermedio *ParameterValuePtr*. • SQL_PARAM_INPUT_OUTPUT: El marcador de parámetro está asociado a un parámetro de entrada/salida del procedimiento almacenado invocado. Una vez ejecutada la sentencia, se envían al servidor los valores de datos reales para el parámetro. El almacenamiento intermedio *ParameterValuePtr* debe contener valores válidos de datos de entrada; el almacenamiento intermedio *StrLen_or_IndPtr* debe contener el correspondiente valor de longitud o SQL_NTS, SQL_NULL_DATA. • SQL_PARAM_OUTPUT: El marcador de parámetro está asociado a un parámetro de salida del procedimiento almacenado invocado o al valor de retorno del procedimiento almacenado. Una vez ejecutada la sentencia, los datos del parámetro de salida se devuelven al almacenamiento intermedio de la aplicación especificado mediante *ParameterValuePtr* y *StrLen_or_IndPtr*, a menos que ambos sean punteros NULL, en cuyo caso se descartan los datos de salida. Si un parámetro de salida no tiene un valor de retorno, *StrLen_or_IndPtr* se establece en SQL_NULL_DATA.

Tabla 34. Argumentos de SQLBindParameter (continuación)

Tipo de datos	Argumento	Uso	Descripción
SQLSMALLINT	*ValueType*	entrada	Tipo de datos C del parámetro. Se da soporte a los tipos siguientes: • SQL_C_BINARY • SQL_C_BIT • SQL_C_CHAR • SQL_C_DOUBLE • SQL_C_FLOAT • SQL_C_LONG • SQL_C_SHORT • SQL_C_TYPE_DATE • SQL_C_TYPE_TIME • SQL_C_TYPE_TIMESTAMP • SQL_C_TINYINT Si se especifica SQL_C_DEFAULT, los datos se transfieren desde su tipo de datos C por omisión al tipo indicado en *ParameterType*.
SQLSMALLINT	*ParameterType*	entrada	Tipo de datos SQL del parámetro. Los tipos soportados son: • SQL_BLOB • SQL_CHAR • SQL_DECIMAL • SQL_INTEGER • SQL_SMALLINT • SQL_TYPE_DATE • SQL_TYPE_TIME • SQL_TYPE_TIMESTAMP • SQL_VARCHAR
SQLUINTEGER	*ColumnSize*	entrada	Precisión del marcador de parámetro correspondiente. • Si *ParameterType* denota una serie binaria o una serie de caracteres de un solo byte (tal como SQL_CHAR, SQL_BLOB), esta es la longitud máxima en bytes del marcador de parámetro. • En otro caso, este argumento no se tiene en cuenta.
SQLSMALLINT	*DecimalDigits*	entrada	Escala del correspondiente parámetro si *ParameterType* es SQL_DECIMAL.
SQLPOINTER	*ParameterValuePtr*	entrada (diferido), salida (diferido), o ambos	• En la entrada (*InputOutputType* establecido en SQL_PARAM_INPUT o SQL_PARAM_INPUT_OUTPUT): Durante la ejecución, si *StrLen_or_IndPtr* no contiene SQL_NULL_DATA, entonces *ParameterValuePtr* apunta a un almacenamiento intermedio que contiene los datos reales del parámetro. • En la salida (InputOutputType establecido en SQL_PARAM_OUTPUT o SQL_PARAM_INPUT_OUTPUT): *ParameterValuePtr* apunta al almacenamiento intermedio donde se almacena el valor del parámetro de salida del procedimiento almacenado. • Si *ParameterValuePtr* es nulo, denota la desvinculación del parámetro.
SQLINTEGER	*BufferLength*	entrada	Para datos binarios y de tipo de carácter, *BufferLength* especifica la longitud del almacenamiento intermedio *ParameterValuePtr*. Para datos que no sean binarios ni de tipo carácter, este argumento no se tiene en cuenta y se considera que el almacenamiento intermedio *ParameterValuePtr* tiene la longitud asociada al tipo de datos C. Para parámetros de salida, *BufferLength* se utiliza para determinar si se truncan los datos.

Tabla 34. Argumentos de SQLBindParameter (continuación)

Tipo de datos	Argumento	Uso	Descripción
SQLINTEGER *	*StrLen_or_IndPtr*	entrada (diferido), salida (diferido), o ambos	• Si es un parámetro de entrada o de entrada/salida: Es el puntero que apunta a la ubicación que contiene (cuando se ejecuta la sentencia) la longitud del valor de marcador de parámetro almacenado en *ParameterValuePtr*. Para especificar un valor nulo para un marcador de parámetro, esta ubicación de almacenamiento debe contener SQL_NULL_DATA. Si *ValueType* es SQL_C_CHAR, esta ubicación de almacenamiento debe contener la longitud exacta de los datos almacenados en *ParameterValuePtr* o SQL_NTS si el contenido de *ParameterValuePtr* termina en nulo. Si contiene la longitud exacta, no se permiten caracteres nulos en los datos almacenados en ParameterValuePtr. Si *ValueType* indica datos de tipo carácter, (explícitamente o implícitamente utilizando SQL_C_DEFAULT) y este puntero se establece en NULL, la aplicación debe proporcionar una serie terminada en nulo en *ParameterValuePtr*. Esto también implica que este marcador de parámetro nunca tiene un valor nulo. • Si se trata de un parámetro de salida (*InputOutputType* se establece en SQL_PARAM_OUTPUT): Debe ser un parámetro de salida o un valor de retorno de una llamada (CALL) a un procedimiento almacenado y debe apuntar a uno de los siguientes valores, después de la ejecución del procedimiento almacenado: – Número de bytes disponibles a devolver en *ParameterValuePtr*, excluyendo el carácter de terminación en nulo. – SQL_NULL_DATA

Uso:

Un marcador de parámetro se representa mediante un símbolo ? en una sentencia de SQL y sirve para indicar una posición en la sentencia donde se colocará un valor proporcionado por la aplicación cuando se ejecute la sentencia. Este valor se puede obtener a partir de una variable de aplicación. Se utiliza `SQLBindParameter()` para enlazar el área de almacenamiento de la aplicación con el marcador de parámetro.

La aplicación debe asociar una variable a cada marcador de parámetro de la sentencia de SQL antes de ejecutarse ésta. Para esta función, *ParameterValuePtr* y *StrLen_or_IndPtr* son argumentos diferidos. Las posiciones de almacenamiento deben ser válidas y contener valores de datos de entrada cuando se ejecute la sentencia. Esto significa que la llamada a `SQLExecDirect()` o a `SQLExecute()` se debe mantener en el mismo ámbito de procedimiento que las llamadas a `SQLBindParameter()`, o bien estas posiciones de almacenamiento se deben asignar dinámicamente o declarar de forma estática o global.

Se hace referencia a los marcadores de parámetros por un número (*ColumnNumber*) y están numerados secuencialmente de izquierda a derecha, comenzando en el 1.

Todos los parámetros enlazados por esta función permanecen en vigor hasta que se invoca una de las funciones siguientes:

- Se invoca `SQLFreeStmt()` con la opción SQL_RESET_PARAMS
- Se invoca `SQLFreeHandle()` con *HandleType* establecido en SQL_HANDLE_STMT
- Se invoca de nuevo `SQLBindParameter()` para el mismo número de parámetro (*ParameterNumber*)

Una vez ejecutada la sentencia de SQL y procesados los resultados, la aplicación puede volver a utilizar el descriptor de sentencia para ejecutar otra sentencia de SQL. Si las especificaciones para el marcador de parámetro son diferentes (número de parámetros, longitud o tipo), se debe invocar `SQLFreeStmt()` con SQL_RESET_PARAMS para restaurar o borrar los enlaces de parámetros.

El tipo de datos del almacenamiento intermedio C indicado por *ValueType* debe ser compatible con el tipo de datos SQL indicado por *ParameterType*, de lo contrario se produce un error.

Debido a que los datos contenidos en las variables referenciadas por *ParameterValuePtr* y *StrLen_or_IndPtr* no se verifican hasta que se ejecuta la sentencia, los errores de contenido o formato de los datos no se detectan ni notifican hasta que se llama a `SQLExecute()` o a `SQLExecDirect()`.

Para esta función, *ParameterValuePtr* y *StrLen_or_IndPtr* son argumentos diferidos. Cuando *InputOutputType* está establecido en SQL_PARAM_INPUT, las posiciones de almacenamiento deben ser válidas y contener valores de datos de entrada cuando se ejecute la sentencia. Esto significa que la llamada a `SQLExecDirect()` o a `SQLExecute()` se debe mantener en el mismo ámbito de procedimiento que las llamadas a `SQLBindParameter()`, o bien estas posiciones de almacenamiento se deben asignar dinámicamente o declarar de forma estática o global.

DB2 Everyplace da soporte a SQL_PARAM_INPUT, SQL_PARAM_INPUT_OUTPUT y SQL_PARAM_OUTPUT. DB2 Everyplace no da soporte a `SQLPutData()`, por lo que no se debe colocar el valor del parámetro en el almacenamiento intermedio *ParameterValuePtr*.

Para datos C binarios y de tipo carácter, el argumento *BufferLength* especifica la longitud del almacenamiento intermedio *ParameterValuePtr*. Para todos los demás tipos de datos C, el argumento *BufferLength* no se tiene en cuenta.

Códigos de retorno:
- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 35. SQLSTATE de SQLBindParameter

SQLSTATE	Descripción	Explicación
07006	Conversión no válida.	La conversión desde el valor de datos identificado por el argumento *ValueType* al tipo de datos identificado por el argumento *ParameterType* no es una conversión válida. (Por ejemplo, la conversión desde SQL_C_DATE a SQL_DOUBLE.)
40003 **08**S01	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
58004	Error inesperado del sistema.	Error no recuperable del sistema.
HY001	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY003	Tipo de programa fuera de rango.	El valor especificado por el argumento *ParameterNumber* no es un tipo de datos válido ni SQL_C_DEFAULT.
HY004	Tipo de datos SQL fuera de rango.	El valor especificado para el argumento *ParameterType* no es un tipo válido de datos SQL.

Tabla 35. SQLSTATE de SQLBindParameter (continuación)

SQLSTATE	Descripción	Explicación
HY009	Valor no válido de argumento.	El argumento *ParameterValuePtr* es un puntero nulo y el argumento *StrLen_or_IndPtr* es un puntero nulo e *InputOutputType* no es SQL_PARAM_OUTPUT.
HY013	Error inesperado de gestión de la memoria.	DB2 CLI no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY090	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El valor especificado para el argumento *BufferLength* era menor que 0.
HY093	Número de parámetros no válido.	El valor especificado para el argumento *ValueType* es menor que uno o mayor que el número máximo de parámetros soportado por el servidor.
HY094	Valor no válido de escala.	El valor especificado para *ParameterType* es SQL_DECIMAL o SQL_NUMERIC y el valor especificado para *DecimalDigits* es menor que 0 o mayor que el valor del argumento *ParamDef* (precisión).
HY104	Valor no válido de precisión.	El valor especificado para *ParameterType* es SQL_DECIMAL o SQL_NUMERIC y el valor especificado para *ParamDef* es menor que uno.
HY105	Tipo de parámetro no válido.	*InputOutputType* no es SQL_PARAM_INPUT.
HYC00	Controlador no apropiado.	CLI de DB2 o la fuente de datos no da soporte a la conversión especificada por la combinación del valor especificado para el argumento *ValueType* y el valor especificado para el argumento *ParameterType*. El valor especificado para el argumento *ParameterType* no está soportado por CLI de DB2 o por la fuente de datos.

Consulta relacionada:

- "Compatibilidad entre tipos de datos para las operaciones de asignación y comparación" en la página 176
- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLExecDirect—Ejecutar una sentencia directamente" en la página 225
- "SQLExecute—Ejecutar una sentencia" en la página 227

SQLConnect—Conectar con una fuente de datos

Finalidad:

Especificación:	CLI de DB2 2.1	ODBC 1.0	ISO CLI

`SQLConnect()` establece una conexión con la base de datos destino.

Antes de invocar esta función se debe asignar un descriptor de conexión utilizando `SQLAllocHandle()`.

Esta función se debe invocar antes de asignar un descriptor de sentencia utilizando `SQLAllocHandle()`.

Sintaxis:

```
SQLRETURN   SQLConnect       (
                SQLHDBC              ConnectionHandle,   /* hdbc */
                SQLCHAR      *FAR ServerName,        /* szDSN */
                SQLSMALLINT       NameLength1,       /* cbDSN */
                SQLCHAR      *FAR UserName,          /* szUID */
                SQLSMALLINT       NameLength2,       /* cbUID */
                SQLCHAR      *FAR Authentication,    /* szAuthStr */
                SQLSMALLINT       NameLength3);      /* cbAuthStr */
```

Argumentos de la función:

Tabla 36. Argumentos de SQLConnect

Tipo de datos	Argumento	Uso	Descripción
SQLHDBC	*ConnectionHandle*	entrada	Descriptor de contexto de conexión.
SQLCHAR *	*ServerName*	entrada	Ubicación y nombre de la base de datos. El nombre es opcional. DB2 Everyplace no tiene en cuenta el nombre.
SQLSMALLINT	*NameLength1*	entrada	Longitud del contenido del argumento *ServerName*.
SQLCHAR *	*UserName*	entrada	Nombre de autorización (identificador de usuario). Esta serie se utiliza con cifrado; en otro caso, DB2 Everyplace la ignora.
SQLSMALLINT	*NameLength2*	entrada	Longitud del contenido del argumento *UserName*.
SQLCHAR *	*Authentication*	entrada	Serie de caracteres de autenticación (contraseña). Esta serie se utiliza con cifrado; en otro caso, DB2 Everyplace la ignora.
SQLSMALLINT	*NameLength3*	entrada	Longitud del contenido del argumento *Authentication*.

Notas:

Un usuario *no registrado* (alguien que no existe en la tabla DB2eSYSUSERS) recibirá el mensaje de aviso, 42704 (objeto no definido) al intentar conectar con una base de datos DB2 Everyplace cifrada durante una llamada a la función SQLGetDiagRec() de CLI. Un usuario *registrado* no recibirá este aviso. En cambio, tanto un usuario registrado como un usuario no registrado pueden conectar con la base de datos durante la llamada a la función SQLConnect() y no recibirán un mensaje de aviso.

Uso:

`SQLConnect()` se puede utilizar para conectar con fuentes de datos en ubicaciones distintas.

Para acceder a una fuente de datos en un dispositivo local, se establece el argumento *ServerName* en un nombre de fuente de datos. DB2 Everyplace ignora el nombre de la fuente de datos y se accede a la fuente de datos local.

Para aplicaciones que utilizan dispositivos de almacenamiento secundario, el argumento *ServerName* acepta una serie que apunte a la ubicación de una *DataSource* que existe localmente o que existe en un dispositivo de almacenamiento secundario soportado tales como IBM Microdrive, Sony Memory Stick, Compact Flash, SD Memory Card o MultiMediaCard. El formato de la serie *ServerName* es el siguiente:

ServerName=Device:/Path/DataSource

Dispositivo
Es el nombre del dispositivo donde se almacena la fuente de datos (*DataSource*). El carácter reservado # se utiliza para acceder a cualquier dispositivo de almacenamiento Compact Flash (CF) de Tipo II (en dispositivos Palm OS con soporte para CF). El almacenamiento secundario se direcciona utilizando el carácter reservado #. `#0` y `#1` especifican la ranura de almacenamiento secundario que se debe acceder. # es equivalente a `#0`. Por ejemplo:

```
ServerName=#:/storage/
```

DB2 Everyplace se conecta a *DataSource* en el directorio `storage` de IBM Microdrive en la primera ranura CF.

Vía Es la vía de acceso de la *DataSource* ubicada en el *Dispositivo*. Si se especifica *Vía* sin especificar *`Dispositivo`*`:/`, se utiliza la vía de acceso del sistema de archivos local referida a la ubicación de la aplicación. No se debe grabar ningún archivo en el directorio raíz de un volumen. Algunos tipos de soportes no dan soporte a los archivos en el directorio raíz. Por ejemplo:

```
ServerName=dir1/dir2/DATA1
```

Nota: En DB2 Everyplace no existe límite en la longitud de la vía de acceso.

Si la aplicación se encuentra en `/myapp` en el sistema de archivos local, DB2 Everyplace se conecta a la *DataSource* ubicada en `/myapp/dir1/dir2/`. El nombre DATA1 de la *DataSource* no se tiene en cuenta.

DataSource
Opcional: Es el nombre de la fuente de datos con el que se establece una conexión. DB2 Everyplace no tiene en cuenta este nombre.

Para acceder a un procedimiento almacenado remoto utilizando el adaptador de Consulta remota y Procedimiento almacenado, se utiliza el argumento *ServerName* para identificar la ubicación y el nombre de la base de datos. Para aplicaciones que utilizan el adaptador de Consulta remota y Procedimiento almacenado para acceder a bases de datos remotas, el argumento *ServerName* acepta el formato URL:

```
http://IPAddress:portNumber/path?DB=DataSource
```

Se necesitan *IPAddress* y *Authentication*.

Si utiliza almacenamiento de objetos de Windows CE en lugar de un sistema de archivos convencional, deberá:

- establecer el parámetro de vía de acceso de la función `SQLConnect` de CLI en `"@:\"`

o bien

- en CLP, ejecutar `'connect to @:\'`

En el almacenamiento de objetos de Windows CE no existe el concepto de "directorio". Cuando se utiliza el almacenamiento de objetos, el usuario no puede especificar el directorio, o vía de acceso, en que se crean las tablas. Todas las tablas del almacenamiento de objetos se crean en el mismo espacio de nombres. Debido a esta limitación, no se pueden establecer varias conexiones simultáneas con el almacenamiento de objetos. A efectos de serialización de conexiones, el archivo de bloqueo se crea en la vía de acceso raíz del sistema de archivos.

Cuando se utiliza almacenamiento de objetos, los archivos DB2 Everyplace no se pueden suprimir de forma manual, a diferencia de la utilización de un sistema de archivos convencional.

Ejemplos:

Para conectar con la fuente de datos situada localmente en `c:\dir1\dir2\`. Se ignora el nombre de fuente de datos `DS1`:

```
ServerName=c:/dir1/dir2/DS1
```

Para conectar localmente con la fuente de datos situada en `/dir1/dir2/` utilizando la notación de sistema de archivos propia de UNIX:

```
ServerName=/dir1/dir2/
```

Para conectar con la fuente de datos situada localmente en el directorio `dir1\`, relativo a la vía de acceso de la aplicación. Si la aplicación está situada en `c:\myapp\`, se accede a la fuente de datos de `c:\myapp\dir1\`:

```
ServerName=dir1\
```

Para conectar con la fuente de datos situada en el directorio `/dir1/` de la memoria de almacenamiento en la ranura 1 del almacenamiento intermedio:

```
ServerName=#1:/dir1/
```

Conectar con el DB2 Everyplace Sync Server `192.168.0.1` en el puerto `8080` y con la base de datos `mysample` utilizando el adaptador de consulta remota y procedimiento almacenado.

```
ServerName=
 http://192.168.0.1:8080/db2e/servlet/com.ibm.mobileservices.adapter
                .agent.AgentServlet?DB=mysample
```

Conectar con la fuente de datos utilizando el almacenamiento de Windows CE.

```
ServerName=@:\
```

Serialización de conexiones:

Consulte el apartado “Serialización de conexiones” en la página 67 para obtener información sobre la serialización de la conexión.

Autentificación de conexiones:

El cifrado de una base de datos requiere una autentificación rudimentaria del usuario. DB2 Everyplace utiliza el UserName y la Authentication para autentificar el usuario durante la conexión.

La autentificación funciona del modo siguiente: Si la tabla de catálogos de DB2eSYSUSERS no existe en la base de datos con la que conecta SQLConnect, se ignora la información de UserName y Authentication. DB2 Everyplace distingue entre usuarios *registrados* y *no registrados*. Un usuario registrado es un usuario que está relacionado en la tabla DB2eSYSUSERS y que se ha añadido mediante la sentencia GRANT SQL. Durante la conexión, si existe una tabla DB2eSYSUSERS y el UserName pertenece a un usuario registrado, se intenta la autentificación. Si la contraseña suministrada en el parámetro Authentication no es correcta, se devuelve un error (42505). Si el UserName no está registrado, la función SQLConnect resultará satisfactoria. Sin embargo, una llamada posterior a SQLGetDiagRec devolverá el aviso 42704 (objeto no definido). Esto permite que las aplicaciones distingan entre el caso de un usuario registrado que se conecta satisfactoriamente y

un usuario no registrado que se conecta satisfactoriamente. Para obtener más información, vea el “Visión general del cifrado local de los datos” en la página 81, 356 y “GRANT” en la página 156.

Códigos de retorno:

- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 37. SQLSTATE de SQLConnect

SQLSTATE	Descripción	Explicación
08001	No se puede conectar con la fuente de datos.	CLI de DB2 no puede establecer una conexión con la fuente de datos (servidor).
08002	Conexión en uso.	El descriptor de conexión (*ConnectionHandle*) especificado ya se ha utilizado para establecer una conexión con una fuente de datos y la conexión todavía está abierta.
08004	El servidor de aplicaciones rechazó el establecimiento de la conexión.	La fuente de datos (servidor) rechazó el establecimiento de la conexión.
58004	Error inesperado del sistema.	Error no recuperable del sistema.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY013	Error inesperado de gestión de la memoria.	CLI de DB2 no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY501	Nombre no válido de la *DataSource*.	El nombre de *DataSource* especificado no es válido.
HYT00	Ha transcurrido el tiempo de espera de conexión.	El tiempo de espera ha transcurrido antes de que la aplicación pudiera conectar con la fuente de datos. El período de tiempo de espera se puede establecer mediante el atributo SQL_ATTR_LOGIN_TIMEOUT de *SQLSetConnectAttr()*. Se devuelve este error cuando hay otra aplicación que está utilizando la base de datos.

Restricciones:

Se debe invocar `SQLConnect()` para poder ejecutar cualquier sentencia de SQL.

Conceptos afines:

- “Serialización de conexiones” en la página 67
- “SQLAllocHandle—Asignar descriptor de contexto” en la página 200
- “SQLDisconnect—Desconectar de una fuente de datos” en la página 222

Consulta relacionada:

- “Clave para las descripciones de funciones de CLI de DB2” en la página 198
- “Resumen de las funciones de CLI de DB2” en la página 194

- "SQLAllocHandle—Asignar descriptor de contexto" en la página 200
- "SQLDisconnect—Desconectar de una fuente de datos" en la página 222

SQLColumns - Obtener información de columna para una tabla

Finalidad:

Especificación:	CLI de DB2 2.1	ODBC 1.0	

SQLColumns() devuelve una lista de columnas en las tablas especificadas. La información se devuelve en un conjunto de resultados de SQL, que se puede recuperar utilizando las mismas funciones que se utilizan para buscar un conjunto de resultados generado por una consulta.

Sintaxis:

```
SQLRETURN  SQLColumns       (
               SQLHSTMT          StatementHandle,   /* hstmt */
               SQLCHAR      FAR  *CatalogName,      /* szCatalogName */
               SQLSMALLINT       NameLength1,       /* cbCatalogName */
               SQLCHAR      FAR  *SchemaName,       /* szSchemaName */
               SQLSMALLINT       NameLength2,       /* cbSchemaName */
               SQLCHAR      FAR  *TableName,        /* szTableName */
               SQLSMALLINT       NameLength3,       /* cbTableName */
               SQLCHAR      FAR  *ColumnName,       /* szColumnName */
               SQLSMALLINT       NameLength4);      /* cbColumnName */
```

Argumentos de la función:

Tabla 38. Argumentos de SQLColumns

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	Entrada	Descriptor de contexto de sentencia.
SQLCHAR	*CatalogName*	Entrada	Es posible que el almacenamiento intermedio contenga un *pattern-value* para calificar el conjunto de resultados. *Catalog* es la primera parte de un nombre de tabla de 3 partes. DB2 Everyplace ignora este argumento.
SQLSMALLINT	*NameLength1*	Entrada	Longitud de *CatalogName*. DB2 Everyplace ignora este argumento.
SQLCHAR	*SchemaName*	Entrada	Es posible que el almacenamiento intermedio contenga un *pattern-value* para calificar el conjunto de resultados por medio de un nombre de esquema. DB2 Everyplace ignora este argumento.
SQLSMALLINT	*NameLength2*	Entrada	Longitud de *SchemaName*. DB2 Everyplace ignora este argumento.
SQLCHAR	*TableName*	Entrada	Es posible que el almacenamiento intermedio contenga un *pattern-value* para calificar el conjunto de resultados por medio de un nombre de tabla.
SQLSMALLINT	*NameLength3*	Entrada	Longitud de *TableName*.
SQLCHAR	*ColumnName*	Entrada	Es posible que el almacenamiento intermedio contenga un *pattern-value* para calificar el conjunto de resultados por medio de un nombre de columna.
SQLSMALLINT	*NameLength4*	Entrada	Longitud de *ColumnName*.

Uso:

Esta función se llama para recuperar información sobre las columnas de una tabla o de un conjunto de tablas. Es posible que una aplicación habitual desee llamar a esta función después de una llamada a SQLTables() para determinar las columnas de una tabla. La aplicación debería utilizar las series de caracteres devueltas en la TABLE_NAME del conjunto de resultados de SQLTables() como entrada para esta función.

SQLColumns() devuelve un conjunto de resultados estándar, ordenado mediante TABLE_NAME y ORDINAL_POSITION. 217 lista las columnas en el conjunto de resultados.

Los argumentos *TableName* y *ColumnName* aceptan patrones de búsqueda.

Esta función no devuelve información sobre las columnas de un conjunto de resultados.En su lugar debería utilizarse SQLDescribeCol() o SQLColAttribute().

Las llamadas a SQLColumns() deben utilizarse con moderación, ya que en muchos casos se asignan a una consulta compleja y por tanto cara en el catálogo del sistema. Los resultados deben guardarse en vez de repetir las llamadas.

Las columnas VARCHAR del conjunto resultante de las funciones de catálogo se han declarado con un atributo de longitud máxima 128 para que sea coherente los límites de SQL92. Dado que los nombres de DB2 son inferiores a 128, la aplicación puede optar por poner siempre a un lado 128 caracteres (más el terminador nulo) para el almacenamiento intermedio de salida o, alternativamente llamar a SQLGetInfo() con SQL_MAX_TABLE_NAME_LEN y SQL_MAX_COLUMN_NAME_LEN para determinar respectivamente las longitudes reales de las columnas TABLE_NAME y COLUMN_NAME a las que da soporte el DBMS conectado.

Aunque en futuros releases se pueden añadir nuevas columnas y modificarse los nombres de las columnas existentes, la posición de las columnas actuales no cambiará.

Columnas devueltas por SQLColumns:

Columna 1 TABLE_CAT (VARCHAR(128))
Siempre es nula (NULL).

Columna 2 TABLE_SCHEM (VARCHAR(128))
Siempre es nula (NULL).

Columna 3 TABLE_NAME (VARCHAR(128) no nulo)
Nombre de la tabla.

Columna 4 COLUMN_NAME (VARCHAR(128) no nulo)
Identificador de columna. Nombre de la columna de la tabla especificada, vista, alias o sinónimo.

Columna 5 DATA_TYPE (SMALLINT no NULL)
Tipo de datos SQL de la columna que se identifica por medio de COLUMN_NAME. Este es uno de los valores de la columna Tipo de datos de SQL simbólico de la sección “Tipos de datos por omisión y simbólicos de SQL” en la página 177.

Columna 6 TYPE_NAME (VARCHAR(128) no NULL)
Serie de caracteres que representa el nombre del tipo de datos que corresponde a DATA_TYPE.

Columna 7 COLUMN_SIZE (INTEGER)

Si el valor de la columna DATA_TYPE denota un carácter o serie binaria, esta columna contendrá la longitud máxima de caracteres para la columna.

Para los tipos de datos DATE, TIME o TIMESTAMP, es el número total de caracteres necesarios para visualizar el valor cuando se convierte en carácter.

Para los tipos de datos numéricos, es el número total de dígitos que se permite en la columna.

Consulte también, "Atributos de tipos de datos" en la página 177.

Columna 8 BUFFER_LENGTH (INTEGER)

El número máximo de bytes para que el almacenamiento intermedio C asociado almacene datos de esta columna en el caso de que SQL_C_DEFAULT se haya especificado en las llamadas `SQLBindCol()`, `SQLGetData()` y `SQLBindParameter()`. Esta longitud no incluye ningún terminador nulo. Para los tipos de datos numéricos exactos, incluye el decimal y el signo. Consulte también, "Atributos de tipos de datos" en la página 177

Columna 9 DECIMAL_DIGITS (SMALLINT)

La escala de la columna. NULL se devuelve para los tipos de datos en los que la escala no resulta aplicable. Consulte también, "Atributos de tipos de datos" en la página 177

Columna 10 NUM_PREC_RADIX (SMALLINT)

10 o NULL.

Si DATA_TYPE es un tipo de datos numérico exacto, esta columna contiene el valor 10 y COLUMN_SIZE contiene el número de dígitos decimales que se permiten para la columna.

Para los tipos de datos numéricos, el DBMS devuelve un NUM_PREC_RADIX de 10.

NULL se devuelve para los tipos de datos en los que el radix no resulta aplicable.

Columna 11 NULLABLE (SMALLINT no NULL)

SQL_NO_NULLS si la columna no acepta valores nulos (NULL).

SQL_NULLABLE si la columna acepta valores nulos (NULL).

Columna 12 REMARKS (VARCHAR(254))

Siempre es nula (NULL).

Columna 13 COLUMN_DEF (VARCHAR(254))

El valor por omisión de la columna. Si el valor por omisión es un literal numérico, esta columna contendrá la representación de caracteres del literal numérico que no tengan comillas simples encerrándolos. Si el valor por omisión es una serie de caracteres, esta columna es dicha serie encerrada entre comillas simples. Si el valor por omisión es una seudo-literal, como por ejemplo para las columnas DATE, TIME y TIMESTAMP, esta columna contendrá la palabra clave de la seudo-literal (por ejemplo CURRENT DATE) sin comillas.

Si NULL se ha especificado como valor por omisión, esta columna devuelve la palabra NULL, sin comillas. Si no se ha especificado ningún valor por omisión, esta columna es NULL.

Columna 14 SQL_DATA_TYPE (SMALLINT no NULL)
Esta columna es igual que la columna DATA_TYPE.

Columna 15 SQL_DATETIME_SUB (SMALLINT)
Esta columna siempre es nula (NULL).

Columna 16 CHAR_OCTET_LENGTH (INTEGER)
Contiene la longitud máxima en octetos para una columna de tipo de datos de carácter. Para conjuntos de caracteres de un único byte, es igual que COLUMN_SIZE. Para todos los demás tipos de datos es NULL.

Columna 17 ORDINAL_POSITION (INTEGER no NULL)
La posición ordinal de la columna de la tabla. La primera columna de la tabla es el número 1.

Columna 18 IS_NULLABLE (VARCHAR(254))

Contiene la serie 'NO' si se sable que la columna no sea anulable; y 'YES' en caso contrario.

Este conjunto de resultados resulta idéntico a la especificación de conjunto de resultados *Columns()* de X/Open CLI, el cual es una versión ampliada del conjunto de resultados *SQLColumns()* especificado en ODBC V2. El conjunto de resultados *SQLColumns()* de ODBC incluye cada una de las columnas en la misma posición.

Nota: Este conjunto de resultados resulta idéntico a la especificación de conjunto de resultados `Columns()` de X/Open CLI, el cual es una versión ampliada del conjunto de resultados `SQLColumns()` especificado en ODBC V2. El conjunto de resultados `SQLColumns()` de ODBC incluye cada una de las columnas en la misma posición.

Códigos de retorno:
- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 39. SQLColumns SQLSTATEs

SQLSTATE	Descripción	Explicación
24000	Estado no válido del cursor.	Ya había un cursor abierto en el descriptor de contexto de sentencia.
40003 **08**S01	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
HY001	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY014	No hay más descriptores de contexto.	DB2 CLI no pudo asignar un descriptor debido a los recursos internos.
HY090	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El valor de uno de los argumentos de longitud de nombre era menor que 0, pero no igual a SQL_NTS.

Restricciones:

Ninguna.

Consulta relacionada:

- "SQLTables - Obtener información de tabla" en la página 283

SQLDescribeCol—Devolver un conjunto de atributos de una columna

Finalidad:

Especificación:	CLI de DB2 1.1	ODBC 1.0	ISO CLI

`SQLDescribeCol()` devuelve un conjunto de información descriptiva de uso habitual (nombre, tipo, precisión, escala, posibilidad de contener nulos) referente a la columna indicada del conjunto resultante generado por una consulta.

Se debe invocar `SQLPrepare()` o `SQLExecDirect()` antes de invocar esta función.

Generalmente esta función se invoca antes de una función de enlace de columna (`SQLBindCol()`) para determinar los atributos de una columna antes de enlazarla a una variable de aplicación.

Sintaxis:

```
SQLRETURN   SQLDescribeCol   (
                SQLHSTMT          StatementHandle,  /* hstmt */
                SQLUSMALLINT      ColumnNumber,      /* icol */
                SQLCHAR      *FAR ColumnName,        /* szColName */
                SQLSMALLINT       BufferLength,      /* cbColNameMax */
                SQLSMALLINT  *FAR NameLengthPtr,     /* pcbColName */
                SQLSMALLINT  *FAR DataTypePtr,       /* pfSqlType */
                SQLUINTEGER  *FAR ColumnSizePtr,     /* pcbColDef */
                SQLSMALLINT  *FAR DecimalDigitsPtr,  /* pibScale */
                SQLSMALLINT  *FAR NullablePtr);      /* pfNullable */
```

Argumentos de la función:

Tabla 40. Argumentos de SQLDescribeCol

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia.
SQLUSMALLINT	*ColumnNumber*	entrada	Número de la columna que se debe describir. Las columnas están numeradas secuencialmente, de izquierda a derecha, comenzando en 1.
SQLCHAR *	*ColumnName*	salida	Puntero que apunta a un almacenamiento intermedio de nombres de columna. Este puntero se establece en NULL si no se puede determinar el nombre de la columna.
SQLSMALLINT	*BufferLength*	entrada	Tamaño del almacenamiento intermedio de *ColumnName*.
SQLSMALLINT *	*NameLengthPtr*	salida	Bytes disponibles para devolver para el argumento *ColumnName*. El nombre de columna (*ColumnName*) se trunca a *BufferLength - 1* bytes si *NameLengthPtr* es mayor o igual que *BufferLength*.
SQLSMALLINT *	*DataTypePtr*	salida	Tipo básico de datos SQL de la columna.
SQLUINTEGER *	*ColumnSizePtr*	salida	Precisión de la columna tal como está definida en la base de datos.
SQLSMALLINT *	*DecimalDigitsPtr*	salida	Escala de la columna tal como está definida en la base de datos (sólo es aplicable a SQL_DECIMAL).
SQLSMALLINT *	*NullablePtr*	salida	Indica si la columna puede contener nulos. Cualquiera: SQL_NO_NULLS SQL_NULLABLE

Uso:

Las columnas se identifican mediante un número, que se asigna secuencialmente de izquierda a derecha, y se pueden describir en cualquier orden. Los números de columna comienzan en el 1.

Si se especifica un puntero nulo para cualquiera de los argumentos de puntero, CLI de DB2 considera que la información no es necesaria para la aplicación y no se devuelve nada.

Códigos de retorno:

- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Si `SQLDescribeCol()` devuelve SQL_ERROR o SQL_SUCCESS_WITH_INFO, se puede obtener uno de los SQLSTATE siguientes al invocar la función `SQLError()`.

Tabla 41. SQLSTATE de SQLDescribeCol

SQLSTATE	Descripción	Explicación
01004	Datos truncados.	El nombre de columna devuelto en el argumento *ColumnName* es más largo que el valor especificado en el argumento *BufferLength*. El argumento *NameLengthPtr* contiene la longitud del nombre de columna completo. (La función devuelve SQL_SUCCESS_WITH_INFO.)
07005	La sentencia no devolvió un conjunto resultante.	La sentencia asociada al descriptor de sentencia (*StatementHandle*) no devolvió un conjunto resultante. No había ninguna columna para describir. (Invoque primero `SQLNumResultCols()` para determinar si hay alguna fila en el conjunto resultante.)
07009	Índice descriptor no válido.	El valor especificado para *ColumnNumber* es menor o igual que 0. El valor especificado para el argumento *ColumnNumber* es mayor que el número de columnas del conjunto resultante.
40003 **08**S01	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
58004	Error inesperado del sistema.	Error no recuperable del sistema.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY002	Número de columna no válido.	El valor especificado para el argumento *ColumnNumber* es menor que 1 o el valor especificado para el argumento *ColumnNumber* es mayor que el número de columnas del conjunto resultante.
HY090	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	La longitud especificada en el argumento *BufferLength* es menor que 1.
HY010	Error de secuencia de función.	Se llama a la función antes de llamar a `SQLPrepare()` o `SQLExecDirect()` para el descriptor de sentencia (*StatementHandle*).
HY013	Error inesperado de gestión de la memoria.	CLI de DB2 no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HYC00	Controlador no apropiado.	CLI de DB2 no reconoce el tipo de datos SQL de la columna *ColumnNumber*.

Restricciones:

DB2 Everyplace sólo da soporte a los siguientes tipos de datos definidos por ODBC:

- SQL_BLOB
- SQL_CHAR
- SQL_DECIMAL
- SQL_INTEGER

- SQL_SMALLINT
- SQL_TYPE_DATE
- SQL_TYPE_TIME
- SQL_TYPE_TIMESTAMP
- SQL_VARCHAR

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLExecDirect—Ejecutar una sentencia directamente" en la página 225
- "SQLNumResultCols—Obtener número de columnas resultantes" en la página 265

SQLDisconnect—Desconectar de una fuente de datos

Finalidad:

Especificación:	CLI de DB2 1.1	ODBC 1.0	ISO CLI

`SQLDisconnect()` cierra la conexión asociada al descriptor de conexión de la base de datos.

Después de invocar esta función, invoque `SQLConnect()` para conectar con otra base de datos o bien invoque `SQLFreeHandle()`.

Sintaxis:

```
SQLRETURN   SQLDisconnect    (SQLHDBC          ConnectionHandle;) /* hdbc */
```

Argumentos de la función:

Tabla 42. Argumentos de SQLDisconnect

Tipo de datos	Argumento	Uso	Descripción
SQLHDBC	*ConnectionHandle*	entrada	Descriptor de contexto de conexión.

Uso:

Si una aplicación invoca `SQLDisconnect()` antes de haber liberado todos los descriptores de contexto de sentencia asociados a la conexión, CLI de DB2 los libera después de desconectarse satisfactoriamente de la base de datos.

Si se devuelve SQL_SUCCESS_WITH_INFO, significa que desconexión de la base de datos se ha realizado satisfactoriamente pero existe información adicional sobre errores o específica de la implementación. Por ejemplo, se encontró un problema durante el proceso posterior a la desconexión de la conexión o no existe ninguna conexión actual debido a un suceso que se produjo independientemente de la aplicación (por ejemplo, un error de comunicaciones).

Después de invocar `SQLDisconnect()` satisfactoriamente, la aplicación puede reutilizar el argumento *ConnectionHandle* para hacer otra petición `SQLConnect()` o `SQLDriverConnect()`.

Códigos de retorno:

- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 43. SQLSTATE de SQLDisconnect

SQLSTATE	Descripción	Explicación
01002	Error de desconexión.	Se produjo un error durante la desconexión. Sin embargo, la desconexión se realizó satisfactoriamente. (La función devuelve SQL_SUCCESS_WITH_INFO).
08003	La conexión está cerrada.	La conexión especificada en el argumento *ConnectionHandle* no está abierta.
58004	Error inesperado del sistema.	Error no recuperable del sistema.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY013	Error inesperado de gestión de la memoria.	CLI de DB2 no puede acceder a la memoria necesaria para ejecutar o finalizar la función.

Restricciones:

Ninguna.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLAllocHandle—Asignar descriptor de contexto" en la página 200
- "SQLConnect—Conectar con una fuente de datos" en la página 211
- "SQLFreeHandle—Liberar recursos de descriptor de contexto" en la página 242

SQLEndTran—Solicitar COMMIT o ROLLBACK

Finalidad:

Especificación:	CLI de DB2	ODBC	ISO CLI

`SQLEndTran()` solicita una operación de COMMIT o ROLLBACK para todas las operaciones activas de todas las sentencias asociadas a una conexión.

Sintaxis:

```
SQLRETURN   SQLEndTran      (SQLSMALLINT       HandleType,
                             SQLHANDLE         Handle,
                             SQLSMALLINT       Completion Type);
```

Argumentos de la función:

Tabla 44. Argumentos de SQLEndTran

Tipo de datos	Argumento	Uso	Descripción
SQLSMALLINT	*HandleType*	entrada	Tipo de descriptor de contexto
SQLHANDLE	*Handle*	entrada	Descriptor de contexto de conexión.
SQLSMALLINT	*CompletionType*	entrada	Cómo completar las operaciones activas asociadas a una conexión

Uso:

HandleType
Identificador de tipo de descriptor de contexto. Sólo se permite SQL_HANDLE_DBC (descriptor de conexión).

Handle Descriptor de contexto, del tipo indicado por *HandleType*.

CompletionType
Uno de los dos valores siguientes:
- SQL_COMMIT
- SQL_ROLLBACK

En modalidad de comprometer de forma manual, `SQLEndTran()` debe invocarse antes de invocar `SQLDisconnect()`. Si *no* se llama a `SQLEndTran()` antes que a `SQLDisconnect()`, se retrotraen las operaciones que han actualizado la base de datos (desde que se inició la última transacción).

Cuando se ejecuta una operación de ROLLBACK, se borran todos los descriptores de contexto de sentencia.

Si la aplicación se interrumpe o termina prematuramente durante su utilización en la modalidad manual, se pierden las actualizaciones realizadas a partir del último COMMIT. `SQLEndTran()` debe invocarse antes de invocar SQLDisconnect().

Códigos de retorno:
- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 45. SQLSTATEs de SQLEndTran

SQLSTATE	Descripción	Explicación
58004	Error inesperado del sistema.	Error no recuperable del sistema.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY013	Error inesperado de gestión de la memoria.	CLI de DB2 no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY014	No hay más descriptores de contexto.	CLI de DB2 no puede asignar un descriptor de contexto debido a recursos internos.

Restricciones:

Ninguna.

Consulta relacionada:
- “Clave para las descripciones de funciones de CLI de DB2” en la página 198
- “Resumen de las funciones de CLI de DB2” en la página 194
- “SQLSetConnectAttr—Establecer opciones referentes a una conexión” en la página 272

SQLError—Recuperar información sobre errores

En ODBC Versión 3, SQLError() se desaprobó y se sustituyó por SQLGetDiagRec() y SQLGetDiagField(); para obtener más información, vea "SQLGetDiagRec—Obtener varios valores de campos del registro de diagnósticos" en la página 254.

Recomendación: Aunque la versión actual de CLI de DB2 sigue dando soporte a SQLError(), utilice SQLGetDiagRec() en sus programas CLI de DB2 para que se ajusten a las normas más recientes.

Migración a la nueva función

Por ejemplo, para obtener información de diagnóstico correspondiente a un descriptor de sentencia determinado, la sentencia:

```
SQLError(henv, hdbc, hstmt, szSqlState, pfNativeError, szErrorMsg,
       cbErrorMsgMax, pcbErrorMsg);
```

se escribiría así utilizando la nueva función:

```
SQLGetDiagRec(SQL_HANDLE_STMT, hstmt, 1, szSqlState, pfNativeError,
       szErrorMsg, cbErrorMsgMax, pcbErrorMsg);
```

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194

SQLExecDirect—Ejecutar una sentencia directamente

Finalidad:

Especificación:	CLI de DB2 1.1	ODBC 1.0	ISO CLI

SQLExecDirect() ejecuta directamente la sentencia de SQL especificada. La sentencia se puede ejecutar una sola vez.

Sintaxis:

```
SQLRETURN   SQLExecDirect    (SQLHSTMT          StatementHandle,   /* hstmt */
                              SQLCHAR      *FAR StatementText,    /* szSqlStr */
                              SQLINTEGER        TextLength);      /* cbSqlStr */
```

Argumentos de la función:

Tabla 46. Argumentos de SQLExecDirect

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia.
SQLCHAR *	*StatementText*	entrada	Serie de la sentencia de SQL.
SQLINTEGER	*TextLength*	entrada	Longitud del contenido del argumento *StatementText*. La longitud debe establecerse en un valor igual a la longitud exacta de la sentencia o, si la sentencia termina con nulo, debe establecerse en SQL_NTS.

Uso:

La serie de la sentencia de SQL no puede contener marcadores de parámetros.

Códigos de retorno:

- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE
- SQL_NO_DATA_FOUND

Se devuelve SQL_NO_DATA_FOUND si la sentencia de SQL es un UPDATE de búsqueda o un DELETE de búsqueda y no hay ninguna fila que cumpla la condición de búsqueda.

Diagnósticos:

Tabla 47. SQLSTATE de SQLExecDirect

SQLSTATE	Descripción	Explicación
22003	Valor numérico fuera de rango.	Un valor numérico asignado a una columna de tipo numérico provocó el truncamiento de la parte entera del número, durante el proceso de asignación o al calcular un resultado intermedio.
42xxx	Error de sintaxis o violación de la regla de acceso.	Los SQLSTATE **42**xxx indican la existencia de diversos problemas de sintaxis o de acceso en la sentencia. xxx representa cualquier SQLSTATE con ese código de clase. Ejemplo: **42**xxx representa cualquier SQLSTATE perteneciente a la clase **42**.
58004	Error inesperado del sistema.	Error no recuperable del sistema.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY009	Valor no válido de argumento.	*StatementText* es un puntero nulo.
HY013	Error inesperado de gestión de la memoria.	CLI de DB2 no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY014	No hay más descriptores de contexto.	CLI de DB2 no puede asignar un descriptor de contexto debido a recursos internos.
HY090	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El argumento *TextLength* es menor que 1 pero no igual a SQL_NTS.

Restricciones:

Ninguna.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLBindCol—Enlazar una columna a una variable de aplicación" en la página 203

SQLExecute—Ejecutar una sentencia

Finalidad:

Especificación:	CLI de DB2 1.1	ODBC 1.0	ISO CLI

`SQLExecute()` ejecuta, una o más veces, una sentencia que se ha preparado satisfactoriamente con `SQLPrepare()`. La sentencia se ejecuta utilizando el valor actual de las variables de aplicación que se enlazaron a marcadores de parámetros mediante `SQLBindParameter()`.

Sintaxis:

```
SQLRETURN   SQLExecute      (SQLHSTMT         StatementHandle);  /* hstmt */
```

Argumentos de la función:

Tabla 48. Argumentos de SQLExecute

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia.

Uso:

La serie de la sentencia de SQL puede contener marcadores de parámetros. Un marcador de parámetro se representa mediante un carácter ? y sirve para indicar una posición en la sentencia donde se colocará un valor proporcionado por la aplicación cuando se invoque `SQLExecute()`. Este valor se puede obtener a partir de una variable de aplicación. Se utiliza `SQLBindParameter()` para enlazar el área de almacenamiento de la aplicación con el marcador de parámetro.

Se deben enlazar todos los parámetros antes de invocar `SQLExecute()`.

Después de procesar los resultados de la llamada a `SQLExecute()`, la aplicación puede ejecutar de nuevo la sentencia con los mismos u otros valores de parámetros.

Las sentencias ejecutadas por `SQLExecDirect()` no se pueden ejecutar de nuevo invocando `SQLExecute()`; se debe primero invocar `SQLPrepare()`.

Si se genera un conjunto resultante, `SQLFetch()` recupera la siguiente fila de datos en variables enlazadas. También se pueden recuperar datos llamando a `SQLGetData()` para cualquier columna que no esté enlazada.

Códigos de retorno:
- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE
- SQL_NO_DATA_FOUND

Se devuelve SQL_NO_DATA_FOUND si la sentencia de SQL es un UPDATE de búsqueda o un DELETE de búsqueda y no hay ninguna fila que cumpla la condición de búsqueda.

Diagnósticos:

Los SQLSTATE para SQLExecute() incluyen todos los correspondientes a SQLExecDirect() (consulte la Tabla 47 en la página 226), excepto **HY**009 y **HY**090, y también incluyen los SQLSTATE de la Tabla 49.

Tabla 49. SQLSTATE de SQLExecute

SQLSTATE	Descripción	Explicación
HY010	Error de secuencia de función.	El descriptor de contexto de sentencia (*StatementHandle*) especificado no está en estado preparado. Se llama a SQLExecute() sin antes llamar a SQLPrepare().
08004	El servidor de aplicaciones rechazó la conexión.	El nombre de usuario o la contraseña utilizados para conectar con la fuente de datos no son correctos.
08S01	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
39001	Una función definida por el usuario ha devuelto un SQLSTATE no válido.	Una función definida por el usuario ha devuelto un SQLSTATE que no es válido.
59101	Usuario no definido.	El usuario no está definido en la base de datos de control del Centro de administración de dispositivos portátiles.
59102	Contraseña no correcta.	La contraseña del usuario no coincide con la contraseña definida en el Centro de administración de dispositivos portátiles.
59103	Grupo no definido.	El grupo no está definido en el Centro de administración de dispositivos portátiles.
59104	Aplicación no definida.	La aplicación no está definida en el Centro de administración de dispositivos portátiles.
59105	Suscripción no definida.	La suscripción con ″AgentAdapter″ no está definida en el Centro de administración de dispositivos portátiles.
59106	Suscripción no completada.	La suscripción no dispone de toda la información necesaria para invocar un procedimiento almacenado remoto.
59120	Error de conversión XML.	AgentAdapter ha fallado al convertir los datos de entrada del usuario en un documento XML.
59121	Se ha producido un error general de AgentAdapter.	Se ha producido un error general de AgentAdapter.
59122	Ha fallado la carga de la biblioteca.	No se encuentran en el sistema algunas de las bibliotecas necesarias.
HY501	Nombre no válido de la fuente de datos.	El nombre de fuente de datos especificado no es válido.

Restricciones:

Ninguna.

Consulta relacionada:

- “Clave para las descripciones de funciones de CLI de DB2” en la página 198

- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLBindParameter—Enlazar un marcador de parámetro a un almacenamiento intermedio" en la página 207
- "SQLBindCol—Enlazar una columna a una variable de aplicación" en la página 203
- "SQLExecDirect—Ejecutar una sentencia directamente" en la página 225
- "SQLPrepare—Preparar una sentencia" en la página 266
- "SQLFetch—Recuperar la fila siguiente"

SQLFetch—Recuperar la fila siguiente

Finalidad:

Especificación:	**CLI de DB2** 1.1	**ODBC** 1.0	

`SQLFetch()` avanza el cursor hasta la siguiente fila del conjunto resultante y recupera las columnas enlazadas.

Las columnas pueden estar asignadas al almacenamiento de la aplicación.

Cuando se llama a `SQLFetch()`, se realiza la transferencia de datos apropiada, junto con la conversión de datos que se haya indicado al enlazar la columna. Las columnas también se pueden obtener individualmente después de la búsqueda, invocando `SQLGetData()`.

`SQLFetch()` sólo se puede invocar después de generar un conjunto resultante (utilizando el mismo descriptor de sentencia) mediante la ejecución de una consulta.

Sintaxis:

```
SQLRETURN   SQLFetch         (SQLHSTMT         StatementHandle);  /* hstmt */
```

Argumentos de la función:

Tabla 50. Argumentos de SQLFetch

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia.

Uso:

`SQLFetch()` sólo se puede invocar después de generar un conjunto resultante para el mismo descriptor de sentencia. Antes de invocar `SQLFetch()` por primera vez, el cursor se sitúa en el inicio del conjunto resultante.

El número de variables de aplicación enlazadas mediante `SQLBindCol()` no debe superar el número de columnas del conjunto resultante, de lo contrario `SQLFetch()` falla.

Si no se ha invocado `SQLBindCol()` para enlazar ninguna columna, `SQLFetch()` no devuelve ningún dato a la aplicación, sino que simplemente avanza el cursor. En este caso, se podría invocar `SQLGetData()` para obtener todas las columnas individualmente. Los datos de las columnas no enlazadas se descartan cuando `SQLFetch()` avanza el cursor hasta la fila siguiente.

Las columnas pueden estar asignadas al almacenamiento de la aplicación. Se utiliza SQLBindCol() para enlazar el almacenamiento de la aplicación a la columna. Durante la recuperación, se transfieren datos desde la base de datos a la aplicación. También se define la longitud de los datos disponibles para devolver.

Si el almacenamiento intermedio de enlace no es lo suficiente grande para albergar los datos devueltos por SQLFetch(), se truncan los datos. Si se truncan datos de tipo carácter, se devuelve SQL_SUCCESS_WITH_INFO, y se genera un estado SQL que informa del truncamiento. El argumento diferido de salida *pcbValue* de SQLBindCol() contiene la longitud real de los datos de columna recuperados del servidor. La aplicación debe comparar la longitud real de la salida con la longitud del almacenamiento intermedio de entrada (argumentos *pcbValue* y *cbValueMax* de SQLBindCol()) para determinar qué columnas de caracteres se truncaron.

El truncamiento de tipos de datos numéricos se notifica en forma de aviso si el truncamiento afecta a dígitos situados a la derecha de la coma decimal. Si el truncamiento se produce a la izquierda de la coma decimal, se devuelve un error (consulte la sección sobre diagnósticos).

Una vez recuperadas todas las filas del conjunto resultante, o cuando no sean necesarias las filas restantes, invoque SQLFreeStmt() para cerrar el cursor y rechazar los datos restantes y los recursos asociados.

DB2 Everyplace recupera una fila como máximo cada vez, en lugar de utilizar un conjunto de filas. DB2 Everyplace no da soporte a los descriptores de sentencias.

SQLFetch() determina si la aplicación ha especificado almacenamientos intermedios separados de longitudes y de indicadores. En este caso, si los datos no son nulos, SQLFetch() establece el almacenamiento intermedio de indicadores en 0 y devuelve la longitud en el almacenamiento intermedio de longitudes. Si los datos son nulos, SQLFetch() establece el almacenamiento intermedio de indicadores en SQL_NULL_DATA y no modifica el almacenamiento intermedio de longitudes.

Posicionamiento del cursor:

Cuando se crea el conjunto resultante, el cursor está situado antes del comienzo del conjunto resultante. SQLFetch() recupera la fila siguiente.

Códigos de retorno:
- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE
- SQL_NO_DATA_FOUND

Se devuelve SQL_NO_DATA_FOUND si no hay ninguna fila en el conjunto resultante o si invocaciones anteriores de SQLFetch() han recuperado todas las filas del conjunto resultante.

Si se han recuperado todas las filas, el cursor está situado después del final del conjunto resultante.

Diagnósticos:

Tabla 51. SQLSTATE de SQLFetch

SQLSTATE	Descripción	Explicación
01004	Datos truncados.	Los datos devueltos para una o más columnas están truncados. Los valores de tipo serie o los valores numéricos se truncan por la derecha. (Se devuelve SQL_SUCCESS_WITH_INFO si no se ha producido ningún error).
07006	Conversión no válida.	Los datos no se pudieron convertir de forma adecuada al tipo de datos especificado por *fCType* en `SQLBindCol()`.
22002	Almacenamiento intermedio de salida o de indicadores no valido.	El valor de puntero especificado para el argumento *pcbValue* en `SQLBindCol()` es un puntero nulo y el valor de la columna correspondiente es nulo. No existe ninguna manera de notificar SQL_NULL_DATA.
58004	Error inesperado del sistema.	Error no recuperable del sistema.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY010	Error de secuencia de función.	Se llama a la función antes de llamar a `SQLPrepare()` o `SQLExecDirect()` para el descriptor de sentencia (*StatementHandle*).
HY013	Error inesperado de gestión de la memoria.	CLI de DB2 no puede acceder a la memoria necesaria para ejecutar o finalizar la función.

Restricciones:

Ninguna.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLBindCol—Enlazar una columna a una variable de aplicación" en la página 203
- "SQLExecDirect—Ejecutar una sentencia directamente" en la página 225
- "SQLGetData—Obtener datos de una columna" en la página 250

SQLFetchScroll—Recuperar conjunto de filas y devolver datos para todas las columnas enlazadas

Finalidad:

Especificación:	**CLI de DB2** 5.0	**ODBC** 3.0	

`SQLFetchScroll()` recupera el conjunto especificado de filas de datos del conjunto resultante y devuelve datos para todas las columnas enlazadas. Los conjuntos de filas se pueden especificar en una posición absoluta o relativa.

Sintaxis:

```
SQLRETURN   SQLFetchScroll  (
                  SQLHSTMT          StatementHandle,
                  SQLSMALLINT       FetchOrientation,
                  SQLINTEGER        FetchOffset);
```

Argumentos de la función:

Tabla 52. Argumentos de SQLFetchScroll

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia
SQLSMALLINT	*FetchOrientation*	entrada	Tipo de recuperación: • SQL_FETCH_NEXT • SQL_FETCH_PRIOR • SQL_FETCH_FIRST • SQL_FETCH_LAST • SQL_FETCH_ABSOLUTE • SQL_FETCH_RELATIVE
SQLINTEGER	*FetchOffset*	entrada	Número de la fila para recuperar. La interpretación de este argumento depende del valor del argumento *FetchOrientation*.

Uso:

`SQLFetchScroll()` devuelve un conjunto de filas especificado del conjunto resultante. Los conjuntos de filas se pueden especificar mediante una posición absoluta o relativa. `SQLFetchScroll()` se puede invocar sólo mientras exista un conjunto resultante, es decir después de que una llamada genere un conjunto resultante y antes de que se cierre el cursor para ese conjunto resultante. Si existe alguna columna enlazada, la función devuelve los datos de esas columnas. Si la aplicación ha especificado un puntero que apunta a una matriz de estado de filas o a un almacenamiento intermedio en el que devolver el número de filas recuperadas, `SQLFetchScroll()` devuelve también esta información. Las llamadas a `SQLFetchScroll()` se pueden intercalar con llamadas a `SQLFetch()`.

Posicionamiento del cursor:

Cuando se crea el conjunto resultante, el cursor está situado antes del comienzo del conjunto resultante. `SQLFetchScroll()` sitúa el cursor de bloques de acuerdo con los valores de los argumentos *FetchOrientation* y *FetchOffset*, tal y como se muestra en la lista siguiente. Las normas exactas para determinar el inicio del nuevo conjunto de filas se muestran en el apartado siguiente.

FetchOrientation	Significado
SQL_FETCH_NEXT	Devolver el conjunto de filas siguiente. Esto equivale a invocar `SQLFetch()`. `SQLFetchScroll()` no tiene en cuenta el valor de *FetchOffset*.
SQL_FETCH_PRIOR	Devolver el conjunto de filas anterior. `SQLFetchScroll()` no tiene en cuenta el valor de *FetchOffset*.
SQL_FETCH_RELATIVE	Devolver el conjunto de filas *FetchOffset* a partir del inicio del conjunto de filas actual.
SQL_FETCH_ABSOLUTE	Devolver el conjunto de filas que comienza en la fila *FetchOffset*.
SQL_FETCH_FIRST	Devolver el primer conjunto de filas del conjunto resultante. `SQLFetchScroll()` no tiene en cuenta el valor de *FetchOffset*.
SQL_FETCH_LAST	Devolver el último conjunto de filas completo del

conjunto resultante. SQLFetchScroll() no tiene en cuenta el valor de *FetchOffset*.

El atributo de sentencia de SQL_ATTR_ROW_ARRAY_SIZE especifica el número de filas del conjunto de filas. Si el conjunto de filas que se recupera mediante SQLFetchScroll() se solapa con el final del conjunto resultante, SQLFetchScroll() devuelve un conjunto de filas parcial. Es decir, si S + R - 1 es mayor que L, siendo S la fila inicial del conjunto de filas que se recupera, R es el tamaño del conjunto de filas y L es la última fila del conjunto resultante, entonces sólo son válidas las primeras L - S + 1 filas del conjunto de filas. Las filas restantes están vacías y su estado es SQL_ROW_NOROW.

Una vez que SQLFetchScroll() vuelve, el cursor del conjunto de filas se sitúa en la primera fila del conjunto resultante.

Reglas del posicionamiento del cursor:

Las secciones siguientes describen las reglas exactas para cada valor de *FetchOrientation*. Estas reglas utilizan la notación siguiente:

Antes del inicio
El cursor de bloques está situado antes del inicio del conjunto resultante. Si la primera fila del nuevo conjunto de filas está delante del inicio del conjunto resultante, SQLFetchScroll() devuelve SQL_NO_DATA.

Después del final
El cursor de bloques está situado después del final del conjunto resultante. Si la primera fila del nuevo conjunto de filas está detrás del final del conjunto resultante, SQLFetchScroll() devuelve SQL_NO_DATA.

CurrRowsetStart
Es el número de la primera fila del conjunto de filas actual.

LastResultRow
Es el número de la última fila del conjunto resultante.

RowsetSize
Es el tamaño del conjunto de filas.

FetchOffset
Es el valor del argumento *FetchOffset*.

Reglas para SQL_FETCH_NEXT:

Tabla 53. Reglas para SQL_FETCH_NEXT:

Condición	Primera fila del nuevo conjunto de filas
Antes del inicio	1
CurrRowsetStart + RowsetSize <= LastResultRow	CurrRowsetStart + RowsetSize
CurrRowsetStart + RowsetSize > LastResultRow	Después del final
Después del final	Después del final

Reglas para SQL_FETCH_PRIOR:

Tabla 54. Reglas para SQL_FETCH_PRIOR:

Condición	Primera fila del nuevo conjunto de filas
Antes del inicio	Antes del inicio
CurrRowsetStart = 1	Antes del inicio
1 < CurrRowsetStart <= RowsetSize	1[a]
CurrRowsetStart > RowsetSize	CurrRowsetStart - RowsetSize
Después del final Y LastResultRow < RowsetSize	1[a]
Después del final Y LastResultRow >= RowsetSize	LastResult - RowRowsetSize + 1

a `SQLFetchScroll()` devuelve SQLSTATE 01S06 (Intento de recuperación antes de que el conjunto resultante devolviera el primer conjunto de filas) y SQL_SUCCESS_WITH_INFO.

Reglas para SQL_FETCH_RELATIVE:

Tabla 55. Reglas para SQL_FETCH_RELATIVE:

Condición	Primera fila del nuevo conjunto de filas
(Antes del inicio Y *FetchOffset* > 0) O BIEN (Después del final Y *FetchOffset* < 0)	--[a]
Antes del inicio Y *FetchOffset* <= 0	Antes del inicio
CurrRowsetStart = 1 AND *FetchOffset* < 0	Antes del inicio
CurrRowsetStart > 1 AND CurrRowsetStart + *FetchOffset* < 1 AND |*FetchOffset*| > RowsetSize	Antes del inicio
CurrRowsetStart > 1 Y CurrRowsetStart + *FetchOffset* < 1 Y |*FetchOffset*| <= RowsetSize	1[b]
1 <= CurrRowsetStart + *FetchOffset* <= LastResultRow	CurrRowsetStart + *FetchOffset*
CurrRowsetStart + *FetchOffset* > LastResultRow	Después del final
Después del final Y *FetchOffset* >= 0	Después del final

a `SQLFetchScroll()` devuelve el mismo conjunto de filas que si se la llamara con *FetchOrientation* establecido en SQL_FETCH_ABSOLUTE. Para obtener más información, consulte la sección SQL_FETCH_ABSOLUTE.

b `SQLFetchScroll()` devuelve SQLSTATE 01S06 (Intento de recuperación antes de que el conjunto resultante devolviera el primer conjunto de filas) y SQL_SUCCESS_WITH_INFO.

Reglas para SQL_FETCH_ABSOLUTE:

Tabla 56. Reglas para SQL_FETCH_ABSOLUTE:

Condición	Primera fila del nuevo conjunto de filas
FetchOffset < 0 Y |*FetchOffset*| <= LastResultRow	LastResultRow + *FetchOffset* + 1
FetchOffset < 0 Y |*FetchOffset*| > LastResultRow Y |*FetchOffset*| > RowsetSize	Antes del inicio
FetchOffset < 0 Y |*FetchOffset* | > LastResultRow Y |*FetchOffset*| <= RowsetSize	1[a]

Tabla 56. Reglas para SQL_FETCH_ABSOLUTE: (continuación)

Condición	Primera fila del nuevo conjunto de filas
FetchOffset = 0	Antes del inicio
1 <= *FetchOffset* <= LastResultRow	*FetchOffset*
FetchOffset > LastResultRow	Después del final

a `SQLFetchScroll()` devuelve SQLSTATE 01S06 (Intento de recuperación antes de que el conjunto resultante devolviera el primer conjunto de filas) y SQL_SUCCESS_WITH_INFO.

Reglas para SQL_FETCH_FIRST::

Tabla 57. Reglas para SQL_FETCH_FIRST:

Condición	Primera fila del nuevo conjunto de filas
Cualquiera	1

Reglas para SQL_FETCH_LAST:

Tabla 58. Reglas para SQL_FETCH_LAST:

Condición	Primera fila del nuevo conjunto de filas
RowsetSize <= LastResultRow	LastResultRow - RowsetSize + 1
RowsetSize > LastResultRow	1

Devolución de datos de columnas enlazadas:

`SQLFetchScroll()` devuelve datos de columnas enlazadas de la misma manera que `SQLFetch()`. Para obtener más información, consulte la sección "SQLFetch—Recuperar la fila siguiente" en la página 229.

Si no existe ninguna columna enlazada, `SQLFetchScroll()` no devuelve datos, pero sitúa el cursor de bloques en la posición especificada. En este caso, y al igual que ocurre con `SQLFetch()`, puede utilizar `SQLGetData()` para recuperar la información.

Direcciones de almacenamientos intermedios:

`SQLFetchScroll()` utiliza el mismo método que `SQLFetch()` para determinar la dirección de los almacenamientos intermedios de datos y de longitudes/indicadores. Para obtener más información, consulte el apartado "SQLBindCol—Enlazar una columna a una variable de aplicación" en la página 203.

Matriz de estado de filas:

La matriz de estado de filas se utiliza para devolver el estado de cada una de las filas del conjunto de filas. La dirección de esta matriz se especifica con el atributo de sentencia de SQL_ATTR_ROW_STATUS_PTR. La matriz es asignada por la aplicación y debe tener tantos elementos como especifique el atributo de sentencia de SQL_ATTR_ROW_ARRAY_SIZE. Sus valores son establecidos por `SQLFetch()` y

SQLFetchScroll(). Si el valor del atributo de sentencia de SQL_ATTR_ROW_STATUS_PTR es un puntero nulo, estas funciones no devuelven el estado de las filas.

El contenido del almacenamiento intermedio de la matriz de estado de filas es indefinido si SQLFetch() o SQLFetchScroll() no devuelve SQL_SUCCESS ni SQL_SUCCESS_WITH_INFO.

La matriz de estado de filas devuelve los valores siguientes.

Valor de la matriz de estado de filas	Descripción
SQL_ROW_SUCCESS	La fila se recupera satisfactoriamente.
SQL_ROW_SUCCESS_WITH_INFO	La fila se recupera satisfactoriamente. No obstante, se devuelve un mensaje de aviso referente a la fila.
SQL_ROW_ERROR	Se produjo un error durante la recuperación de la fila.
SQL_ROW_NOROW	El conjunto de filas se solapó con el final del conjunto resultante y no se devolvió ninguna fila que corresponda a este elemento de la matriz de estado de filas.

Almacenamiento intermedio de filas recuperadas:

El almacenamiento intermedio de filas recuperadas se utiliza para devolver el número de filas recuperadas, incluidas aquéllas para las que no se devolvió ningún dato debido a un error producido al recuperar las filas. Es el número de filas para las que el valor en la matriz de estado de filas no es SQL_ROW_NOROW. La dirección de este almacenamiento intermedio se especifica con el atributo de sentencia de SQL_ATTR_ROWS_FETCHED_PTR. El almacenamiento intermedio es asignado por la aplicación. Sus valores son establecidos por SQLFetch() y SQLFetchScroll(). Si el valor del atributo de sentencia de SQL_ATTR_ROWS_FETCHED_PTR es un puntero nulo, estas funciones no devuelven el número de filas recuperadas. Para determinar el número de la fila actual en el conjunto resultante, una aplicación puede invocar SQLGetStmtAttr() con el atributo SQL_ATTR_ROW_NUMBER.

El contenido del almacenamiento intermedio de filas recuperadas es indefinido si SQLFetch() o SQLFetchScroll() no devuelve SQL_SUCCESS ni SQL_SUCCESS_WITH_INFO, excepto cuando se devuelve SQL_NO_DATA, en cuyo caso el valor contenido en el almacenamiento intermedio de filas recuperadas se establece en 0.

Códigos de retorno:
- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_NO_DATA
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 59. SQLSTATE de SQLFetchScroll

SQLSTATE	Descripción	Explicación
01000	Aviso	Mensaje informativo. (La función devuelve SQL_SUCCESS_WITH_INFO).
01004	Datos truncados.	Los datos devueltos para una o más columnas están truncados. Los valores de tipo serie o los valores numéricos se truncan por la derecha. (Se devuelve SQL_SUCCESS_WITH_INFO si no se ha producido ningún error).
01S06	Se ha intentado recuperar antes de que el conjunto resultante devolviera el primer conjunto de filas.	El conjunto de filas solicitado se solapó con el inicio del conjunto resultante cuando la posición actual estaba más allá de la primera fila, y FetchOrientation era SQL_PRIOR o bien FetchOrientation era SQL_RELATIVE con un desplazamiento (FetchOffset) negativo cuyo valor absoluto era menor o igual que el valor actual de SQL_ATTR_ROW_ARRAY_SIZE. (La función devuelve SQL_SUCCESS_WITH_INFO).
07006	Conversión no válida.	Los datos no se pudieron convertir de forma adecuada al tipo de datos especificado por *fCType* en `SQLBindCol()`.
22002	Almacenamiento intermedio de salida o de indicadores no valido.	El valor de puntero especificado para el argumento *pcbValue* en `SQLBindCol()` es un puntero nulo y el valor de la columna correspondiente es nulo. No existe ninguna manera de notificar SQL_NULL_DATA.
22003	Valor numérico fuera de rango.	La devolución del valor numérico para una o más columnas enlazadas hubiera provocado el truncamiento de la parte entera del número.
24000	Estado no válido del cursor.	*StatementHandle* está en estado de ejecutado, pero no hay ningún conjunto resultante asociado al *StatementHandle*.
HY000	Error general.	Se ha producido un error para el que no existe ningún SQLSTATE específico. El mensaje de error devuelto por `SQLGetDiagRec()` en el almacenamiento intermedio **MessageText* describe el error y su causa.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY010	Error de secuencia de función.	Se llama a la función antes de llamar a `SQLPrepare()` o `SQLExecDirect()` para el descriptor de sentencia (*StatementHandle*).
HY106	Tipo de recuperación fuera de rango.	El valor especificado para el argumento *FetchOrientation* no es válido. El valor del atributo de sentencia de SQL_CURSOR_TYPE es SQL_CURSOR_FORWARD_ONLY y el valor del argumento *FetchOrientation* no es SQL_FETCH_NEXT.

Tabla 59. SQLSTATE de SQLFetchScroll (continuación)

SQLSTATE	Descripción	Explicación
HY107	Valor de fila fuera de rango.	El valor especificado con el atributo de sentencia de SQL_ATTR_CURSOR_TYPE es SQL_CURSOR_KEYSET_DRIVEN, pero el valor especificado con el atributo de sentencia de SQL_ATTR_KEYSET_SIZE es mayor que 0 y menor que el valor especificado con el atributo de sentencia de SQL_ATTR_ROW_ARRAY_SIZE.

Restricciones:

Ninguna.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLBindCol—Enlazar una columna a una variable de aplicación" en la página 203
- "SQLDescribeCol—Devolver un conjunto de atributos de una columna" en la página 220
- "SQLExecDirect—Ejecutar una sentencia directamente" en la página 225
- "SQLFetch—Recuperar la fila siguiente" en la página 229
- "SQLExecute—Ejecutar una sentencia" en la página 227
- "SQLNumResultCols—Obtener número de columnas resultantes" en la página 265
- "SQLSetStmtAttr—Establecer opciones referentes a una sentencia" en la página 276

SQLForeignKeys—Obtener la lista de columnas de clave foránea

Finalidad:

Especificación:	CLI de DB2 2.1	ODBC 1.0	

`SQLForeignKeys()` devuelve información sobre claves foráneas para la tabla especificada. La información se devuelve en un conjunto resultante de SQL, que se puede procesar utilizando las mismas funciones que se utilizan para recuperar un conjunto resultante generado por una consulta. *PKCatalogName*, *NameLength1*, *PKSchemaName*, *NameLength2*, *FKCatalogName*, *NameLength4*, *FKSchemaName* y *NameLength5* se ignoran. Las columnas 1, 2, 5, 6, 12 y 13 del conjunto resultante devuelto son siempre una serie de longitud cero. Las columnas 10, 11 y 14 del conjunto resultante devuelto son siempre cero.

Sintaxis:

```
SQLRETURN   SQLForeignKeys  (
                  SQLHSTMT         StatementHandle,  /* hstmt */
                  SQLCHAR      *FAR PKCatalogName,   /* szPkCatalogName */
                  SQLSMALLINT      NameLength1,      /* cbPkCatalogName */
                  SQLCHAR      *FAR PKSchemaName,    /* szPkSchemaName */
                  SQLSMALLINT      NameLength2,      /* cbPkSchemaName */
                  SQLCHAR      *FAR PKTableName,     /* szPkTableName */
```

```
SQLSMALLINT       NameLength3,      /* cbPkTableName */
SQLCHAR      *FAR FKCatalogName,    /* szFkCatalogName */
SQLSMALLINT       NameLength4       /* cbFkCatalogName */
SQLCHAR      *FAR FKSchemaName,     /* szFkSchemaName */
SQLSMALLINT       NameLength5,      /* cbFkSchemaName */
SQLCHAR      *FAR FKTableName,      /* szFkTableName */
SQLSMALLINT       NameLength6);     /* cbFkTableName */
```

Argumentos de la función:

Tabla 60. Argumentos de SQLForeignKeys

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia.
SQLCHAR*	*PKCatalogName*	entrada	Calificador de catálogo de la tabla de claves primarias. DB2 Everyplace no tiene en cuenta este campo.
SQLSMALLINT	*NameLength1*	entrada	Longitud de *PKCatalogName*. DB2 Everyplace no tiene en cuenta este campo.
SQLCHAR*	*PKSchemaName*	entrada	Calificador de esquema de la tabla de claves primarias. DB2 Everyplace no tiene en cuenta este campo.
SQLSMALLINT	*NameLength2*	entrada	Longitud de *PKSchemaName*. DB2 Everyplace no tiene en cuenta este campo.
SQLCHAR*	*PKTableName*	entrada	Nombre de la tabla donde reside la clave primaria.
SQLSMALLINT	*NameLength3*	entrada	Longitud de *PKTableName*.
SQLCHAR*	*FKCatalogName*	entrada	Calificador de catálogo de la tabla donde reside la clave foránea. DB2 Everyplace no tiene en cuenta este campo.
SQLSMALLINT	*NameLength4*	entrada	Longitud de *FKCatalogName*. DB2 Everyplace no tiene en cuenta este campo.
SQLCHAR*	*FKSchemaName*	entrada	Calificador de esquema de la tabla donde reside la clave foránea. DB2 Everyplace no tiene en cuenta este campo.
SQLSMALLINT	*NameLength5*	entrada	Longitud de *FKSchemaName*. DB2 Everyplace no tiene en cuenta este campo.
SQLCHAR*	*FKTableName*	entrada	Nombre de la tabla donde reside la clave foránea.
SQLSMALLINT	*NameLength6*	entrada	Longitud de *FKTableName*.

Uso:

Si *PKTableName* contiene un nombre de tabla y *FKTableName* es una serie de caracteres vacía, `SQLForeignKeys()` devuelve un conjunto resultante que contiene la clave primaria de la tabla especificada y todas las claves foráneas (en otras tablas) que hacen referencia a la clave primaria.

Si *FKTableName* contiene un nombre de tabla y *PKTableName* es una serie de caracteres vacía, `SQLForeignKeys()` devuelve un conjunto resultante que contiene todas las claves foráneas de la tabla especificada y las claves primarias (en otras tablas) a las que hacen referencia.

Si tanto *PKTableName* como *FKTableName* contienen nombres de tablas, `SQLForeignKeys()` devuelve las claves foráneas de la tabla especificada en *FKTableName* que hacen referencia a la clave primaria de la tabla especificada en *PKTableName*. El resultado debe ser una sola clave como máximo.

Si se solicitan las claves foráneas asociadas a una clave primaria, el conjunto resultante se ordena de acuerdo con FKTABLE_NAME y ORDINAL_POSITION. Si se solicitan las claves primarias asociadas a una clave foránea, el conjunto resultante se ordena de acuerdo con PKTABLE_NAME y ORDINAL_POSITION.

Las columnas VARCHAR del conjunto resultante de las funciones de catálogo se han declarado con un atributo de longitud máxima 128 para que sea coherente los límites de SQL92.

Aunque en futuros releases de SQL se pueden añadir nuevas columnas y modificarse los nombres de las columnas existentes, la posición de las columnas actuales no cambiará.

El conjunto resultante contiene las columnas siguientes::

Columna 1 PKTABLE_CAT (VARCHAR(128))
Siempre es una serie de longitud cero.

Columna 2 PKTABLE_SCHEM (VARCHAR(128))
Siempre es una serie de longitud cero.

Columna 3 PKTABLE_NAME (VARCHAR(128) no nulo)
Nombre de la tabla donde reside la clave primaria.

Columna 4 PKCOLUMN_NAME (VARCHAR(128) no nulo)
Nombre de la columna de clave primaria.

Columna 5 FKTABLE_CAT (VARCHAR(128))
Siempre es una serie de longitud cero.

Columna 6 FKTABLE_SCHEM (VARCHAR(128))
Siempre es una serie de longitud cero.

Columna 7 FKTABLE_NAME (VARCHAR(128) no nulo)
Nombre de la tabla donde reside la clave foránea.

Columna 8 FKCOLUMN_NAME (VARCHAR(128) no nulo)
Nombre de la columna de clave foránea.

Columna 9 ORDINAL_POSITION (SMALLINT no nulo)
Posición ordinal de la columna en la clave, comenzando en 1.

Columna 10 UPDATE_RULE (SMALLINT)
Esto es siempre cero.

Columna 11 DELETE_RULE (SMALLINT)
Esto es siempre cero.

Columna 12 FK_NAME (VARCHAR(128))
Siempre es una serie de longitud cero.

Columna 13 PK_NAME (VARCHAR(128))
Siempre es una serie de longitud cero.

Columna 14 DEFERRABILITY (SMALLINT)
Esto es siempre cero.

Los nombres de columna utilizados por CLI de DB2 siguen el estilo de especificación de X/Open para CLI CAE. El tipo, contenido y orden de las columnas son idénticos a los definidos para el conjunto resultante `SQLForeignKeys()` en ODBC.

Códigos de retorno:
- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 61. SQLSTATE de SQLForeign

SQLSTATE	Descripción	Explicación
24000	Estado no válido del cursor.	Ya hay un cursor abierto para el descriptor de contexto de sentencia.
40003 **08**S01	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY009	Valor no válido de argumento.	Los argumentos *PKTableName* y *FKTableName* eran ambos punteros nulos.
HY010	Error de secuencia de función.	Se llama a la función mientras se produce una operación de datos en ejecución (`SQLPrepare()` o `SQLExecDirect()`).
HY014	No hay más descriptores de contexto.	CLI de DB2 no puede asignar un descriptor de contexto debido a recursos internos.
HY090	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El valor de uno de los argumentos de longitud de nombre es menor que 0, pero no igual a SQL_NTS

Restricciones:

Ninguna.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLPrimaryKeys—Obtener columnas de clave primaria de una tabla" en la página 268

SQLFreeConnect—Liberar descriptor de conexión

En ODBC versión 3, `SQLFreeConnect()` se desaprobó y se sustituyó por `SQLFreeHandle()`; vea "SQLFreeHandle—Liberar recursos de descriptor de contexto" en la página 242 para obtener más información.

Recomendación: Aunque la versión actual de CLI de DB2 sigue dando soporte a `SQLFreeConnect()`, utilice `SQLFreeHandle()` en sus programas CLI de DB2 para que se ajusten a las normas más recientes.

Migración a la nueva función

Por ejemplo, la sentencia:

```
SQLFreeConnect(hdbc);
```

se escribiría así utilizando la nueva función:

```
SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
```

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194

SQLFreeEnv—Liberar descriptor de entorno

En ODBC versión 3, SQLFreeEnv() se desaprobó y se sustituyó por SQLFreeHandle(); vea "SQLFreeHandle—Liberar recursos de descriptor de contexto" para obtener más información.

Recomendación: Aunque la versión actual de CLI de DB2 sigue dando soporte a SQLFreeEnv(), utilice SQLFreeHandle() en sus programas CLI de DB2 para que se ajusten a las normas más recientes.

Migración a la nueva función

Por ejemplo, la sentencia:

```
SQLFreeEnv(henv);
```

se escribiría así utilizando la nueva función:

```
SQLFreeHandle(SQL_HANDLE_ENV, henv);
```

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194

SQLFreeHandle—Liberar recursos de descriptor de contexto

Finalidad:

Especificación:	CLI de DB2 5.0	ODBC 3.0	ISO CLI

SQLFreeHandle() libera recursos asociados a un descriptor de entorno, conexión o sentencia específico.

Esta función es una función genérica para liberar recursos. Sustituye a SQLFreeConnect (para liberar un descriptor de conexión) y a SQLFreeEnv() (para liberar un descriptor de entorno). SQLFreeHandle() también sustituye a SQLFreeStmt() (con la opción SQL_DROP) para liberar un descriptor de sentencia.

Sintaxis:

```
SQLRETURN   SQLFreeHandle    (SQLSMALLINT      HandleType,
                              SQLHANDLE        Handle);
```

Argumentos de la función:

Tabla 62. Argumentos de SQLFreeHandle

Tipo de datos	Argumento	Uso	Descripción
SQLSMALLINT	*HandleType*	entrada	El tipo de descriptor de contexto que debe ser liberado por SQLFreeHandle(). Debe ser uno de los valores siguientes: SQL_HANDLE_ENV SQL_HANDLE_DBC SQL_HANDLE_STMT Si *HandleType* no es uno de los valores anteriores, SQLFreeHandle() devuelve SQL_INVALID_HANDLE.
SQLHANDLE	*Handle*	entrada	El nombre del descriptor de contexto que se debe liberar.

Uso:

`SQLFreeHandle()` se utiliza para liberar descriptores de contexto para entornos, conexiones y sentencias.

Una vez liberado un descriptor de contexto, no debe ser utilizado por una aplicación; CLI de DB2 no comprueba la validez de un descriptor de contexto en una llamada a función.

Liberación de un descriptor de entorno:

Antes de llamar a `SQLFreeHandle()` con un *HandleType* de SQL_HANDLE_ENV, una aplicación debe llamar a `SQLFreeHandle()` con un *HandleType* de SQL_HANDLE_DBC para todas las conexiones asignadas bajo el entorno. De lo contrario, la llamada a `SQLFreeHandle()` devuelve SQL_ERROR y el entorno y las conexiones activas que haya siguen siendo válidos.

Liberación de un descriptor de conexión:

Antes de invocar `SQLFreeHandle()` con *HandleType* establecido en SQL_HANDLE_DBC, la aplicación debe invocar `SQLDisconnect()` para la conexión. De lo contrario, la llamada a `SQLFreeHandle()` devuelve SQL_ERROR y la conexión sigue siendo válida.

Liberación de un descriptor de sentencia:

La llamada a `SQLFreeHandle()` con *HandleType* establecido en SQL_HANDLE_STMT libera todos los recursos que fueron asignados por una llamada a `SQLAllocHandle()` con *HandleType* establecido en SQL_HANDLE_STMT. Cuando una aplicación invoca `SQLFreeHandle()` para liberar una sentencia que tiene resultados pendientes, éstos se eliminan. Si hay resultados pendientes cuando se invoca `SQLFreeHandle()`, se descartan los conjuntos resultantes.

`SQLDisconnect()` descarta automáticamente las sentencias que estén abiertas en la conexión.

Códigos de retorno:

- SQL_SUCCESS
- SQL_ERROR
- SQL_INVALID_HANDLE

Si `SQLFreeHandle()` devuelve SQL_ERROR, el descriptor de contexto es válido todavía.

Diagnósticos:

Tabla 63. SQLSTATE de SQLFreeHandle

SQLSTATE	Descripción	Explicación
01000	Aviso.	Mensaje informativo. (La función devuelve SQL_SUCCESS_WITH_INFO).
08S01	Error en el enlace de comunicaciones.	El argumento *HandleType* es SQL_HANDLE_DBC, y el enlace de comunicaciones entre CLI de DB2 y la fuente de datos con el que estaba intentando conectar ha fallado antes de que la función terminara su proceso.

Tabla 63. SQLSTATE de SQLFreeHandle (continuación)

SQLSTATE	Descripción	Explicación
HY000	Error general.	Se ha producido un error para el que no existe ningún SQLSTATE específico. El mensaje de error devuelto por `SQLGetDiagRec()` en el almacenamiento intermedio **MessageText* describe el error y su causa.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY010	Error de secuencia de función.	El argumento *HandleType* es SQL_HANDLE_ENV y el estado de al menos una conexión es asignado o conectado. Antes de llamar a `SQLFreeHandle()` con un *HandleType* de SQL_HANDLE_ENV, se debe llamar para cada conexión a `SQLDisconnect()` y `SQLFreeHandle()` con un *HandleType* de SQL_HANDLE_DBC. El argumento *HandleType* es SQL_HANDLE_DBC, y se llama a la función antes de llamar a `SQLDisconnect()` para la conexión. El argumento *HandleType* es SQL_HANDLE_STMT; se llamó a `SQLExecute()` o `SQLExecDirect()` con el descriptor de contexto de sentencia, y la función devolvió SQL_NEED_DATA. (DM) Todos los descriptores de contexto auxiliares y otros recursos no se liberaron antes de llamar a `SQLFreeHandle()`.
HY013	Error inesperado de gestión de la memoria.	El argumento *HandleType* es SQL_HANDLE_STMT y no se pudo procesar la llamada a la función debido a que no se pudo acceder a los objetos de memoria asociados, posiblemente debido a una falta de memoria.
HY017	Utilización no válida de un descriptor de contexto de descriptor asignado automáticamente.	El argumento *Handle* se establece en el descriptor de contexto para un descriptor asignado automáticamente o para un descriptor de implementación.

Restricciones:

Ninguna.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLAllocHandle—Asignar descriptor de contexto" en la página 200

SQLFreeStmt—Liberar (o restaurar) un descriptor de sentencia

Finalidad:

Especificación:	**CLI de DB2** 1.1	**ODBC** 1.0	**ISO CLI**

`SQLFreeStmt()` finaliza el proceso para la sentencia referenciada por el descriptor de sentencia. Utilice esta función para:

- Desasociar (restaurar) parámetros respecto a variables de aplicación.
- Descarta el descriptor de sentencia y liberar los recursos de CLI de DB2 asociados al descriptor de sentencia.

`SQLFreeStmt()` se invoca después de ejecutar una sentencia de SQL y procesar los resultados.

Sintaxis:

```
SQLRETURN   SQLFreeStmt      (SQLHSTMT          StatementHandle,   /* hstmt */
                              SQLUSMALLINT      Option); /* fOption */
```

Argumentos de la función:

Tabla 64. Argumentos de SQLFreeStmt

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia.
SQLUSMALLINT	*Option*	entrada	Opción que especifica la manera de liberar el descriptor de sentencia. La opción debe tener uno de los valores siguientes: SQL_DROP o SQL_RESET_PARAMS.

Uso:

`SQLFreeStmt()` se puede invocar con las opciones siguientes:

SQL_DROP
Se liberan los recursos de CLI de DB2 asociados al descriptor de sentencia de entrada y se invalida el descriptor de contexto. Se descartan todos los resultados pendientes.

Esta opción se sustituye por una llamada a `SQLFreeHandle()` con el *HandleType* establecido en SQL_HANDLE_STMT.

Recomendación: Aunque la versión actual de CLI de DB2 sigue dando soporte a esta opción, utilice `SQLFreeHandle()` en sus programas CLI de DB2 para que se ajusten a las normas más recientes.

SQL_RESET_PARAMS
Libera todos los almacenamientos intermedios de parámetros establecidos por `SQLBindParameter()` para *StatementHandle*.

Como método alternativo, puede descartar el descriptor de sentencia y asignar uno nuevo.

Códigos de retorno:

- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

No se devuelve SQL_SUCCESS_WITH_INFO si *Option* está establecido en SQL_DROP, pues no habría ningún descriptor de sentencia para utilizar cuando se invoca `SQLError()`.

Diagnósticos:

Tabla 65. SQLSTATE de SQLFreeStmt

SQLSTATE	Descripción	Explicación
40003 **08**S01	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
58004	Error inesperado del sistema.	Error no recuperable del sistema.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY092	Tipo de opción fuera de rango.	El valor especificado para el argumento *Option* no es SQL_DROP ni SQL_RESET_PARAMS.

Restricciones:

Ninguna.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLAllocHandle—Asignar descriptor de contexto" en la página 200
- "SQLBindCol—Enlazar una columna a una variable de aplicación" en la página 203

SQLGetConnectAttr—Obtener el valor actual de un atributo de conexión

Finalidad:

Especificación:	CLI de DB2 5.0	ODBC 3.0	ISO CLI

SQLGetConnectAttr() devuelve el valor actual de un atributo de una conexión.

Sintaxis:

```
SQLRETURN   SQLGetConnectAttr(SQLHDBC          ConnectionHandle,
                              SQLINTEGER       Attribute,
                              SQLPOINTER       ValuePtr,
                              SQLINTEGER       BufferLength,
                              SQLINTEGER       *StringLengthPtr);
```

Argumentos de la función:

Tabla 66. Argumentos de SQLGetConnectAttr

Tipo de datos	Argumento	Uso	Descripción
SQLHDBC	*ConnectionHandle*	entrada	Descriptor de contexto de conexión.
SQLINTEGER	*Attribute*	entrada	Atributo que se debe recuperar.
SQLPOINTER	*ValuePtr*	salida	Puntero a la memoria en la que devolver el valor actual del atributo especificado por *Attribute*.

Tabla 66. Argumentos de SQLGetConnectAttr (continuación)

Tipo de datos	Argumento	Uso	Descripción
SQLINTEGER	*BufferLength*	entrada	• Si *ValuePtr* apunta a una serie de caracteres, este argumento debe ser igual a la longitud de **ValuePtr*. • Si *ValuePtr* es un puntero, pero no apunta a una serie, *BufferLength* debe tener el valor *SQL_IS_POINTER*. • Si el valor de **ValuePtr* es una serie de Unicode, el argumento de *BufferLength* debe ser un número par.
SQLINTEGER *	*StringLengthPtr*	salida	Puntero a un almacenamiento intermedio en el que se devolverá el número total de bytes (excluyendo el carácter de terminación en nulo) disponibles para devolverlos en **ValuePtr*. • Si *ValuePtr* es un puntero nulo, no se devuelve ninguna longitud. • Si el valor del atributo es una serie de caracteres, y si el número de bytes disponibles para devolverlos es mayor que *BufferLength* menos la longitud de un carácter de terminación en nulo, los datos de **ValuePtr* se truncan en *BufferLength* menos la longitud del carácter de terminación en nulo y la CLI de DB2 termina en nulo.

Uso:

Una llamada a `SQLGetConnectAttr`() devuelve en **ValuePtr* el valor del atributo de conexión especificado en *Attribute*. En DB2 Everyplace, este valor es un valor de 32 bits y no se utilizan los argumentos *BufferLength* ni *StringLengthPtr*.

Los atributos de conexión siguientes pueden recuperarse por medio de `SQLGetConnectAttr`(). Para una descripción de los atributos, consulte SQLSetConnectAttr--Establecer opciones relacionadas con una conexión.

- SQL_ATTR_AUTOCOMMIT (DB2 CLI/ODBC)
- SQL_ATTR_CONNECTION_DEAD (DB2 CLI/ODBC)
- SQL_ATTR_LOGIN_TIMEOUT (DB2 CLI/ODBC)
- SQL_ATTR_FILENAME_FORMAT (DB2 Everyplace)

En función del atributo, una aplicación no necesita establecer una conexión antes de llamar a `SQLGetConnectAttr`().

Códigos de retorno:

- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_NO_DATA
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 67. SQLGetConnectAttr SQLSTATEs

SQLSTATE	Descripción	Explicación
01000	Aviso.	Mensaje informativo. (La función devuelve SQL_SUCCESS_WITH_INFO).
01004	Datos truncados.	Los datos devueltos en **ValuePtr* se truncan hasta *BufferLength* menos la longitud de un carácter de terminación en nulo. La longitud del valor de la serie no truncada se devuelve en **StringLengthPtr*. (La función devuelve SQL_SUCCESS_WITH_INFO).
08003	La conexión está cerrada.	Se ha especificado un valor de *Attribute* que necesitaba una conexión abierta.
HY000	Error general.	Se ha producido un error para el que no existía ningún SQLSTATE específico. El mensaje de error devuelto por `SQLGetDiagRec`() en el almacenamiento intermedio **MessageText* describe el error y su causa.
HY001	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función. Es probable que la memoria a nivel de proceso se haya agotado para el proceso de la aplicación. Consulte la configuración del sistema operativo para obtener más información sobre las limitaciones de memoria a nivel de proceso.
HY090	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El valor especificado para el argumento *BufferLength* era menor que 0.
HY092	Tipo de opción fuera de rango.	El valor especificado para el argumento *Attribute* no era válido.
HYC00	Controlador no apropiado.	El valor especificado para el argumento Attribute era un atributo de sentencia o conexión válida para la versión del controlador de CLI de DB2, pero la fuente de datos no lo soporta.

Restricciones:

Ninguna.

Consulta relacionada:

- "SQLSetConnectAttr—Establecer opciones referentes a una conexión" en la página 272

SQLGetCursorName—Obtener nombre de cursor

Finalidad:

Especificación:	CLI de DB2 1.1	ODBC 1.0	ISO CLI

SQLGetCursorName() devuelve el nombre de cursor que está asociado al descriptor de sentencia de entrada. Si se establece explícitamente un nombre de cursor mediante una llamada a SQLSetCursorName(), se devuelve este nombre; en otro caso, se devuelve un nombre generado implícitamente.

Sintaxis:

```
SQLRETURN   SQLGetCursorName (
                SQLHSTMT          StatementHandle,  /* hstmt */
                SQLCHAR      *FAR CursorName,        /* szCursor */
                SQLSMALLINT       BufferLength,      /* cbCursorMax */
                SQLSMALLINT  *FAR NameLengthPtr);    /* pcbCursor */
```

Argumentos de la función:

Tabla 68. Argumentos de SQLGetCursorName

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia
SQLCHAR *	*CursorName*	salida	Nombre del cursor
SQLSMALLINT	*BufferLength*	entrada	Longitud del almacenamiento intermedio *CursorName*
SQLSMALLINT *	*NameLengthPtr*	salida	Número de bytes disponibles para devolver para *CursorName*

Uso:

SQLGetCursorName() devuelve el nombre de cursor que se estableció explícitamente mediante SQLSetCursorName() o, si no se definió ningún nombre, devuelve el nombre de cursor generado internamente por CLI de DB2.

Si se establece explícitamente un nombre de cursor mediante SQLSetCursorName(), se devuelve este nombre hasta que se descarte la sentencia, o hasta que se defina explícitamente otro nombre.

Los nombres de cursor generados internamente comienzan siempre con SQLCUR o SQL_CUR. Los nombres de cursor tienen siempre 18 caracteres o menos y siempre son exclusivos dentro de una conexión.

Códigos de retorno:
- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 69. SQLSTATE de SQLGetCursorName

SQLSTATE	Descripción	Explicación
01004	Datos truncados.	El nombre de cursor devuelto en *CursorName* es más largo que el valor contenido en *BufferLength*, y se trunca hasta que su longitud sea de *BufferLength* - 1 bytes. El argumento *NameLengthPtr* contiene la longitud del nombre de cursor completo que está disponible para devolver. La función devuelve SQL_SUCCESS_WITH_INFO.
40003 **08**S01	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
58004	Error inesperado del sistema.	Error no recuperable del sistema.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY010	Error de secuencia de función.	Se llama a la función mientras se produce una operación de datos en ejecución (`SQLParamData()` o `SQLPutData()`).
HY013	Error inesperado de gestión de la memoria.	CLI de DB2 no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY090	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El valor especificado para el argumento *BufferLength* era menor que 0.

Restricciones:

Ninguna.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLExecDirect—Ejecutar una sentencia directamente" en la página 225

SQLGetData—Obtener datos de una columna

Finalidad:

Especificación:	CLI de DB2 1.1	ODBC 1.0	ISO CLI

`SQLGetData()` recupera datos de una columna individual en la fila actual del conjunto resultante. Esta función es una alternativa al uso de `SQLBindCol()`, que se utiliza para transferir datos directamente a variables de aplicación en cada llamada a `SQLFetch()`.

Se debe invocar `SQLFetch()` antes de `SQLGetData()`.

Después de llamar a `SQLGetData()` para cada columna que sea necesario, se llama a `SQLFetch()` para recuperar la fila siguiente.

Sintaxis:

```
SQLRETURN   SQLGetData       (
                SQLHSTMT          StatementHandle,  /* hstmt */
                SQLUSMALLINT      ColumnNumber,      /* icol */
                SQLSMALLINT       TargetType,        /* fCType */
                SQLPOINTER        TargetValuePtr,    /* rgbValue */
                SQLINTEGER        BufferLength,      /* cbValueMax */
                SQLINTEGER   *FAR StrLen_or_IndPtr); /* pcbValue */
```

Argumentos de la función:

Tabla 70. Argumentos de SQLGetData

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia.
SQLUSMALLINT	*ColumnNumber*	entrada	Número de columna para el que se solicita la obtención de datos. Las columnas del conjunto resultante están numeradas secuencialmente. Los números de columna comienzan en el 1.
SQLSMALLINT	*TargetType*	entrada	El tipo de datos C de la columna identificada por *ColumnNumber*. Se da soporte a los tipos siguientes: SQL_C_BINARY SQL_C_BIT SQL_C_CHAR SQL_C_DOUBLE SQL_C_FLOAT SQL_C_LONG SQL_C_SHORT SQL_C_TYPE_DATE SQL_C_TYPE_TIME SQL_C_TYPE_TIMESTAMP SQL_C_TINYINT Si se especifica SQL_C_DEFAULT, los datos se convierten a su tipo de datos C por omisión.
SQLPOINTER	*TargetValuePtr*	salida	Puntero al almacenamiento intermedio donde deben almacenarse los datos de la columna recuperada. El almacenamiento intermedio de salida debe estar alineado por palabras (igual). Muchos procesadores, tales como Motorola 68000, tienen reglas para la alineación por palabras, y la aplicación debe alinear debidamente el almacenamiento intermedio para los datos que no son de tipo carácter.

Tabla 70. Argumentos de SQLGetData (continuación)

Tipo de datos	Argumento	Uso	Descripción
SQLINTEGER	*BufferLength*	entrada	Tamaño máximo del almacenamiento intermedio al que apunta *TargetValuePtr*. Si *TargetType* denota una serie binaria o de caracteres, BufferLength debe ser mayor que 0 o se devuelve un error. En otro caso, este argumento no se tiene en cuenta.
SQLINTEGER *	*StrLen_or_IndPtr*	salida	Puntero al valor que indica el número de bytes que CLI de DB2 tiene disponibles para devolverlos en el almacenamiento intermedio *TargetValuePtr*. Si se produce el truncamiento de los datos, esto contiene el número total de bytes necesarios para recuperar la columna entera. Para los tipos de datos binarios y de caracteres, la aplicación puede elegir, alternativamente, la modalidad de recuperación gradual a fin de recuperar grandes cantidades de datos por partes. En esta modalidad, el argumento *StrLen_or_IndPtr* contiene el número de bytes que *quedan* en la columna. El valor es SQL_NULL_DATA si el valor de datos de la columna es nulo. Si este puntero es nulo (NULL) y `SQLFetch()` obtuvo una columna que contiene datos nulos, esta función falla porque no tiene forma de notificar este hecho. Si `SQLFetch()` recuperó una columna que contiene datos binarios, el puntero que apunta a *StrLen_or_IndPtr* no debe ser nulo (NULL) o la función fallará porque no tiene otra forma de notificar a la aplicación la longitud de los datos recuperados en el almacenamiento intermedio *TargetValuePtr*.

Uso:

`SQLGetData()` se puede utilizar con `SQLBindCol()` para el mismo conjunto resultante si se utiliza `SQLFetch()`. Los pasos generales son:

1. `SQLFetch()` avanza el cursor hasta la primera fila, recupera la primera fila y transfiere datos para columnas enlazadas.
2. `SQLGetData()` transfiere datos para la columna especificada.
3. `SQLGetData()` repite el paso 2 para cada columna necesaria.
4. `SQLFetch()` avanza el cursor hasta la fila siguiente, recupera la fila siguiente y transfiere datos para columnas enlazadas.

5. Se repiten los pasos 2, 3 y 4 para cada fila del conjunto resultante, o hasta que el conjunto resultante ya no sea necesario.

Para desechar los datos de la columna antes de terminar el proceso de recuperación, la aplicación puede invocar `SQLGetData()` con *ColumnNumber* establecido en la posición de columna deseada. Para desechar datos que no se han recuperado para la fila completa, la aplicación debe llamar a `SQLFetch()` para avanzar hasta la fila siguiente; o, si la aplicación no desea más datos del conjunto resultante, llama a `SQLFreeStmt()`.

El argumento de entrada *TargetType* determina el tipo de conversión de datos (si la hay) que es necesaria antes de colocar los datos de la columna en el área de almacenamiento indicada por *TargetValuePtr*.

El valor devuelto en *TargetValuePtr* termina siempre en nulo, a menos que los datos de columna a recuperar sean binarios.

El truncamiento de tipos de datos numéricos se notifica en forma de aviso si el truncamiento afecta a dígitos situados a la derecha de la coma decimal. Si el truncamiento se produce a la izquierda de la coma decimal, se devuelve un error.

Códigos de retorno:
- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE
- SQL_NO_DATA_FOUND

Se devuelve SQL_SUCCESS si `SQLGetData()` recupera una serie de longitud cero. En este caso, *StrLen_or_IndPtr* contendrá 0 y *TargetValuePtr* contendrá un terminador nulo.

Si la llamada anterior a `SQLFetch()` ha fallado, no llame a `SQLGetData()` porque el resultado es indefinido

Diagnósticos:

Tabla 71. SQLSTATE de SQLGetData

SQLSTATE	Descripción	Explicación
01004	Datos truncados.	Los datos devueltos para la columna especificada (*ColumnNumber*) están truncados. Los valores de tipo serie o los valores numéricos se truncan por la derecha. Se devuelve SQL_SUCCESS_WITH_INFO.
07006	Conversión no válida.	El valor de datos no se puede convertir al tipo de datos C especificado por el argumento *TargetType*. Se ha llamado anteriormente a la función para el mismo valor de *ColumnNumber*, pero con un valor diferente de *TargetType*.
22002	Almacenamiento intermedio de salida o de indicadores no valido.	El valor de puntero especificado para el argumento *StrLen_or_IndPtr* era un puntero nulo y el valor de la columna es nulo. No existe ninguna manera de notificar SQL_NULL_DATA.

Tabla 71. SQLSTATE de SQLGetData (continuación)

SQLSTATE	Descripción	Explicación
22005	Error de asignación.	Un valor devuelto es incompatible con el tipo de datos indicado por el argumento *TargetType*.
40003 **08**S01	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
58004	Error inesperado del sistema.	Error no recuperable del sistema.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY002	Número de columna no válido.	La columna especificada es menor que 0 o mayor que el número de columnas resultantes.
HY003	Tipo de programa fuera de rango.	*TargetType* no es un tipo de datos válido ni SQL_C_DEFAULT.
HY010	Error de secuencia de función.	Se llama a la función sin antes llamar a `SQLFetch()`.
HY013	Error inesperado de gestión de la memoria.	CLI de DB2 no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY090	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El valor del argumento *BufferLength* es menor que 0 y el argumento *TargetType* es SQL_C_CHAR o SQL_C_BINARY, o bien *TargetType* es SQL_C_DEFAULT y el tipo por omisión es SQL_C_CHAR, SQL_C_BINARY o SQL_C_DBCHAR.
HYC00	Controlador no apropiado.	CLI de DB2 reconoce, pero no da soporte a, el tipo de datos SQL especificado. CLI de DB2 o la fuente de datos no puede convertir el tipo de datos SQL al tipo *TargetType* de los datos de la aplicación

Restricciones:

Ninguna.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLBindCol—Enlazar una columna a una variable de aplicación" en la página 203

SQLGetDiagRec—Obtener varios valores de campos del registro de diagnósticos

Finalidad:

Especificación:	**CLI de DB2** 5.0	**ODBC** 3.0	

SQLGetDiagRec() devuelve el valor actual del campo SQLSTATE de un registro de diagnóstico que contiene información sobre errores, avisos y estados.

Antes de invocar esta función se debe asignar un descriptor de conexión utilizando SQLAllocHandle().

Sintaxis:

```
SQLRETURN   SQLGetDiagRec    (SQLSMALLINT      HandleType,
                              SQLHANDLE        Handle,
                              SQLSMALLINT      RecNumber,
                              SQLCHAR          *SQLState,
                              SQLINTEGER       *NativeErrorPtr,
                              SQLCHAR          *MessageText,
                              SQLSMALLINT      BufferLength,
                              SQLSMALLINT      *TextLengthPtr);
```

Argumentos de la función:

Tabla 72. Argumentos de SQLGetDiagRec

Tipo de datos	Argumento	Uso	Descripción
SQLSMALLINT	*HandleType*	entrada	Identificador del tipo de descriptor de contexto que describe el tipo de descriptor de contexto para el cual se desea información de diagnóstico. Puede ser SQL_HANDLE_STMT o SQL_HANDLE_DBC.
SQLHANDLE	*Handle*	entrada	Descriptor de contexto correspondiente a la estructura de datos de diagnóstico; su tipo está indicado por *HandleType*.
SQLSMALLINT	*RecNumber*	entrada	Indica el registro de estado en el que la aplicación busca información. Su valor debe ser 1.
SQLCHAR	*SQLState*	salida	Puntero que apunta a un almacenamiento intermedio en el que devolver un código SQLSTATE, de cinco caracteres, perteneciente al registro de diagnóstico *RecNumber*. Los primeros dos caracteres indican la clase; los tres siguientes indican la subclase.
SQLINTEGER	*NativeErrorPtr*	salida	Puntero que apunta a un almacenamiento intermedio en el que devolver el código de error nativo, que es específico de la fuente de datos.
SQLCHAR	*MessageText*	salida	Puntero que apunta a un almacenamiento intermedio en el que devolver el texto del mensaje de error. Los campos devueltos por SQLGetDiagRec() están contenidos en una serie de texto.
SQLINTEGER	*BufferLength*	entrada	Longitud (en bytes) del almacenamiento intermedio *MessageText*.
SQLSMALLINT	*TextLengthPtr*	salida	Puntero que apunta a un almacenamiento intermedio que contendrá el número total de bytes (excepto los bytes necesarios para el carácter de terminación en nulo) para devolver en *MessageText*. Si el número de bytes para devolver es mayor que *BufferLength*, el texto del mensaje de error contenido en *MessageText* se trunca hasta que su longitud sea *BufferLength* menos la longitud del carácter de terminación en nulo.

Uso:

Habitualmente una aplicación invoca SQLGetDiagRec() cuando una llamada anterior a una función de CLI de DB2 devuelve un código de retorno distinto de SQL_SUCCESS.

SQLGetDiagRec() devuelve una serie de caracteres que contiene varios campos del registro de la estructura de datos de diagnóstico.

El funcionamiento de SQLGetDiagRec() se ha ampliado en la Versión 8.1 de DB2 Everyplace. Ahora se pueden devolver los SQLSTATE siguientes: 57011, HY024, HY092, HY000, HY012. Para obtener más información sobre estos SQLSTATE, consulte el “Listado de los SQLSTATE” en la página 180.

SQLGetDiagRec() recupera sólo la información de diagnóstico más reciente asociada al descriptor de contexto especificado en el argumento *Handle*. Si la aplicación

invoca otra función, excepto SQLGetDiagRec(), se pierde la información de diagnóstico obtenida en llamadas anteriores para el mismo descriptor de contexto.

Argumento *HandleType*

Cada tipo de descriptor de contexto tiene información de diagnóstico asociada a él. El argumento *HandleType* indica el tipo del descriptor de contexto especificado en *Handle*. DB2 Everyplace soporta descriptores de contexto de sentencia y de conexión.

Códigos de retorno:
- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

SQLGetDiagRec() no notifica valores de error por sí mismo. Utiliza los valores de retorno siguientes para notificar el resultado de su propia ejecución:

SQL_SUCCESS
La función devolvió satisfactoriamente información de diagnóstico.

SQL_SUCCESS_WITH_INFO
El almacenamiento intermedio *MessageText* es demasiado pequeño para contener el mensaje de diagnóstico solicitado. No se genera ningún registro de diagnóstico. Para determinar que se ha producido un truncamiento, la aplicación debe comparar *BufferLength* con el número real de bytes disponibles, que está escrito en *StringLengthPtr*.

SQL_INVALID_HANDLE
El descriptor de contexto indicado por *HandleType* y *Handle* no es un descriptor de contexto válido.

SQL_ERROR
Se ha producido una de las situaciones siguientes:
- *RecNumber* es negativo o 0.
- *BufferLength* es menor que cero.

SQL_NO_DATA
RecNumber es mayor que el número de registros de diagnóstico que existían para el descriptor de contexto especificado en *Handle*. La función también devuelve SQL_NO_DATA para cualquier valor positivo de *RecNumber* si no hay ningún registro de diagnóstico para *Handle*.

Restricciones:

Ninguna.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194

SQLGetInfo—Obtener información general

Finalidad:

Especificación:	CLI de DB2 1.1	ODBC 1.0	ISO CLI

SQLGetInfo() devuelve información general (que incluye conversiones de datos soportadas) sobre el DBMS al que está conectada la aplicación.

Sintaxis:

```
SQLRETURN   SQLGetInfo      (
              SQLHDBC              ConnectionHandle,   /* hdbc */
              SQLUSMALLINT         InfoType,           /* fInfoType */
              SQLPOINTER        InfoValuePtr,    /* rgbInfoValue */
              SQLSMALLINT       BufferLength,       /* cbInfoValueMax */
              SQLSMALLINT     *FAR StringLengthPtr,       /* pcbInfoValue */
```

Argumentos de la función:

Tabla 73. Argumentos de SQLGetInfo

Tipo de datos	Argumento	Uso	Descripción
SQLHDBC	*ConnectionHandle*	entrada	Descriptor de contexto de conexión de base de datos
SQLUSMALLINT	*InfoType*	salida	El tipo de información deseado. El argumento debe ser uno de los valores de la primera columna de las tablas en Tipos de Datos y Conversión de Datos.
SQLPOINTER	*InfoValuePtr*	salida (también entrada)	Puntero al almacenamiento intermedio donde la función almacena la información necesaria. Se pueden recuperar 5 tipos de información: Valor entero de 16 bits Valor entero de 32 bits Valor binario de 32 bits Máscara de 32 bits Serie de caracteres terminada en nulo
SQLSMALLINT	*BufferLength*	entrada	Tamaño máximo del almacenamiento intermedio al que apunta *InfoValuePtr*.

Tabla 73. Argumentos de SQLGetInfo (continuación)

Tipo de datos	Argumento	Uso	Descripción
SQLSMALLINT *	*StrLen_or_IndPtr*	salida	Puntero a la ubicación donde esta función devuelve el número total de bytes disponibles para devolver la información deseada. Si la salida de la función es una serie de caracteres, este tamaño no incluye el carácter de terminación en nulo. Si el valor contenido en la ubicación apuntada por *StringLengthPtr* es mayor que el tamaño del almacenamiento intermedio *InfoValuePtr* tal como está especificado en *BufferLength*, los datos de salida se truncarán hasta que su longitud sea *BufferLength* - 1 bytes, y la función devolverá SQL_SUCCESS_WITH_INFO.

Uso:

Consulte en la sección Información que devuelve SQLGetInfo una lista de los valores posibles de *InfoType* y una descripción de la información que `SQLGetInfo()` devuelve para dicho valor.

DB2 CLI devuelve un valor para cada *InfoType* de esta tabla. Si no se aplica el *InfoType* o no está soportado, el resultado dependerá del tipo de retorno:

- Si el tipo de retorno es una serie de caracteres que contiene 'Y' o 'N', se devuelve "N".
- Si el tipo de retorno es una serie de caracteres que contiene un valor que no es simplemente 'Y' o 'N', se devuelve una serie vacía.
- Si el tipo de retorno es un entero de 16 bits, se devuelve 0 (cero).
- Si el tipo de retorno es un entero de 32 bits, se devuelve 0 (cero).
- Si el tipo de retorno es una máscara de 32 bits, se devuelve 0 (cero).

Información que devuelve SQLGetInfo

SQL_DBMS_NAME (serie)
El nombre del producto DBMS al que se está accediendo. Por ejemplo: "DB2 Everyplace".

SQL_DBMS_VER (serie)
La versión del producto DBMS de DB2 Everyplace. La información devuelta es una serie con formato: `DB2 Everyplace Vm.v.r Build aaaa-mm-dd`, donde `m` es la versión principal, `v` es la versión secundaria, `r` es el release y `aaaa-mm-dd` es la fecha de creación expresada en formato ISO.

Por ejemplo:

```
'DB2 Everyplace V8.1.2 Build 2003-04-01'
es DB2 Everyplace Versión 8.1.2 creado el 1 de Abril de 2003
```

Nota: Las aplicaciones necesitan un almacenamiento intermedio que pueda contener como mínimo 39 caracteres (BUFSIZE). Por ejemplo:

```
rc = SQLGetInfo(hdbc, SQL_DBMS_VER, buf, BUFSIZE, &len);
```

SQL_IDENTIFIER_QUOTE_CHAR (serie)
Indica el carácter utilizado para rodear un identificador delimitado.

SQL_MAX_BINARY_LITERAL_LEN (entero de 32 bits sin signo)
Valor entero de 32 bits sin signo que especifica la longitud máxima de un literal hexadecimal en una sentencia SQL.

SQL_MAX_CHAR_LITERAL_LEN (entero de 32 bits sin signo)
La longitud máxima de un literal de carácter en una sentencia SQL (en bytes).

SQL_MAX_COLUMN_NAME_LEN (entero de 16 bits)
La longitud máxima de un nombre de columna (en bytes).

SQL_MAX_COLUMNS_IN_GROUP_BY (entero de 16 bits)
Indica el número máximo de columnas al que da soporte un servidor en una cláusula GROUP BY. Cero si no hay límite.

SQL_MAX_COLUMNS_IN_INDEX (entero de 16 bits)
Indica el número máximo de columnas al que da soporte el servidor en un índice. Cero si no hay límite.

SQL_MAX_COLUMNS_IN_ORDER_BY (entero de 16 bits)
Indica el número máximo de columnas al que da soporte el servidor en una cláusula ORDER BY. Cero si no hay límite.

SQL_MAX_COLUMNS_IN_SELECT (entero de 16 bits)
Indica el número máximo de columnas al que da soporte el servidor en una cláusula de selección. Cero si no hay límite.

SQL_MAX_CONCURRENT_ACTIVITIES (entero de 16 bits)
El número máximo de entornos activos al que puede dar soporte el controlador CLI de DB2 Everyplace. Si no hay un límite especificado o el límite es desconocido, este valor se establece en cero.

SQL_MAX_DRIVER_CONNECTIONS (entero de 16 bits)
El número máximo de conexiones activas soportadas por aplicación.

SQL_MAX_INDEX_SIZE (entero de 32 bits sin signo)
Indica el tamaño máximo en bytes al que da soporte el servidor para las columnas combinadas en un índice. Cero si no hay límite.

SQL_MAX_ROW_SIZE (entero de 32 bits sin signo)
Especifica la longitud máxima en bytes al que da soporte el servidor en una única fila de una tabla base. Cero si no hay límite.

SQL_MAX_STATEMENT_LEN (entero de 32 bits sin signo)
Indica la longitud máxima de una serie de sentencia SQL en bytes, incluyendo el número de espacios en blanco de la sentencia.

SQL_MAX_TABLE_NAME_LEN (entero de 16 bits)
La longitud máxima de un nombre de tabla (en bytes).

SQL_MAX_TABLES_IN_SELECT (entero de 16 bits)
Indica el número máximo de nombres de tabla que se permiten en una cláusula FROM de una especificación de consulta.

SQL_MAX_USER_NAME_LEN (entero de 16 bits)
Indica el tamaño máximo que se permite para un identificador de usuario (en bytes).

SQL_SEARCH_PATTERN_ESCAPE (serie)
Se utiliza para especificar lo que el controlador soporta como carácter de escape para las funciones de catálogo, como por ejemplo (`SQLTables()`, `SQLColumns()`).

SQL_TXN_CAPABLE (entero de 16 bits)
Indica si las transacciones puede contener DDL o DML o ambos.

- SQL_TC_NONE = transacciones no soportadas.
- SQL_TC_DML = las transacciones sólo pueden contener sentencias DML (SELECT, INSERT, UPDATE, DELETE, etc.) Las sentencias DDL (CREATE TABLE, DROP INDEX, etc.) que se encuentren en una transacción ocasionarán un error.
- SQL_TC_DDL_COMMIT = las transacciones sólo pueden contener sentencias DML. Las sentencias DDL que se encuentren en una transacción harán que se confirme la transacción.
- SQL_TC_DDL_IGNORE = las transacciones sólo pueden contener sentencias DML. Las sentencias DDL que se encuentren en una transacción se ignorarán.
- SQL_TC_ALL = las transacciones pueden contener sentencias DDL y DML en cualquier orden.

SQL_USER_NAME (serie)
El nombre de usuario utilizado en una base de datos concreta. Este es el identificador especificado en la llamada `SQLConnect()`.

Códigos de retorno:

- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Restricciones:

Ninguna.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194

SQLGetStmtAttr—Obtener el valor actual de un atributo de sentencia

Finalidad:

Especificación:	CLI de DB2 5.0	ODBC 3.0	ISO CLI

`SQLGetStmtAttr()` devuelve el valor actual de un atributo de una sentencia.

Sintaxis:

```
SQLRETURN   SQLGetStmtAttr      (
                SQLHSTMT            StatementHandle,
                SQLINTEGER          Attribute,
```

```
SQLPOINTER          ValuePtr,
SQLINTEGER          BufferLength,
SQLINTEGER         *StringLengthPtr);
```

Argumentos de la función:

Tabla 74. Argumentos de SQLGetStmtAttr

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia.
SQLINTEGER	*Attribute*	entrada	Atributo que se debe recuperar.
SQLPOINTER	*ValuePtr*	salida	Puntero a un almacenamiento intermedio en el que devolver el valor del atributo especificado en *Attribute*.
SQLINTEGER	*BufferLength*	entrada	Si *Attribute* es un atributo definido por ODBC y *ValuePtr* apunta a una serie de caracteres o almacenamiento intermedio binario, este argumento debe ser igual a la longitud de **ValuePtr*. Si *Attribute* es un atributo definido por ODBC y **ValuePtr* es un entero, se ignora *BufferLength*. Si *Attribute* es un atributo de la CLI de DB2, la aplicación indica la naturaleza del atributo estableciendo el argumento *BufferLength*. *BufferLength* puede tener los valores siguientes: • Si **ValuePtr* es un puntero a una serie de caracteres, *BufferLength* es la longitud de la serie o SQL_NTS. • Si **ValuePtr* es un puntero a un almacenamiento intermedio binario, la aplicación coloca el resultado de la macro SQL_LEN_BINARY_ATTR(longitud) en *BufferLength*. • Si **ValuePtr* es un puntero a un valor distinto de una serie de caracteres o serie binaria, *BufferLength* debe tener el valor SQL_IS_POINTER. • Si **ValuePtr* contiene un tipo de datos de longitud fija, *BufferLength* es SQL_IS_INTEGER o SQL_IS_UINTEGER, según corresponda.

Tabla 74. Argumentos de SQLGetStmtAttr (continuación)

Tipo de datos	Argumento	Uso	Descripción
SQLSMALLINT	**StringLengthPtr*	salida	Puntero a un almacenamiento intermedio en el que se devolverá el número total de bytes (excluyendo el carácter de terminación en nulo) disponibles para devolverlos en **ValuePtr*. Si se trata de un puntero nulo, no se devuelve ninguna longitud. Si el valor del atributo es una serie de caracteres, y si el número de bytes disponibles para devolverlos es mayor o igual que *BufferLength*, los datos **ValuePtr* se truncan hasta *BufferLength* menos la longitud de un carácter de terminación en nulo, y la CLI de DB2 los termina en nulo.

Uso:

Una llamada a `SQLGetStmtAttr()` devuelve en **ValuePtr* el valor del atributo de sentencia especificado en *Attribute*. En DB2 Everyplace, este valor es un valor de 32 bits y no se utilizan los argumentos *BufferLength* ni *StringLengthPtr*.

`SQLGetStmtAttr()` puede recuperar los atributos de sentencia siguientes. Para ver una descripción de los atributos, consulte "SQLSetStmtAttr—Establecer opciones referentes a una sentencia" en la página 276.

- SQL_ATTR_CURSOR_SCROLLABLE (DB2 CLI/ODBC)
- SQL_ATTR_CURSOR_SENSITIVITY (DB2 CLI/ODBC)
- SQL_ATTR_CURSOR_TYPE (DB2 CLI/ODBC)
- SQL_ATTR_ROW_ARRAY_SIZE (DB2 CLI/ODBC)
- SQL_ATTR_ROW_BIND_TYPE (DB2 CLI/ODBC)
- SQL_ATTR_ROW_NUMBER (DB2 CLI/ODBC)
- SQL_ATTR_DELETE_MODE (DB2 Everyplace)
- SQL_ATTR_DIRTYBIT_SET_MODE (DB2 Everyplace)
- SQL_ATTR_READ_MODE (DB2 Everyplace)
- SQL_ATTR_REORG_MODE (DB2 Everyplace)

Códigos de retorno:
- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 75. SQLSTATE de SQLGetStmtAttr

SQLSTATE	Descripción	Explicación
01000	Aviso.	Mensaje informativo. (La función devuelve SQL_SUCCESS_WITH_INFO).

Tabla 75. SQLSTATE de SQLGetStmtAttr (continuación)

SQLSTATE	Descripción	Explicación
01004	Datos truncados.	Los datos devueltos en **ValuePtr* se truncan hasta *BufferLength* menos la longitud de un carácter de terminación en nulo. La longitud del valor de la serie no truncada se devuelve en *StringLengthPtr*. (La función devuelve SQL_SUCCESS_WITH_INFO).
24000	Estado no válido del cursor.	El argumento *Attribute* es SQL_ATTR_ROW_NUMBER y el cursor no está abierto, o el cursor está posicionado delante del comienzo del conjunto resultante o detrás del final del mismo.
HY000	Error general.	Se ha producido un error para el que no existe ningún SQLSTATE específico. El mensaje de error devuelto por `SQLGetDiagRec()` en el almacenamiento intermedio **MessageText* describe el error y su causa.
HY001	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY010	Error de secuencia de función.	Se llama a una función de ejecución asíncrona para *StatementHandle* y se sigue ejecutando cuando se llama a esta función. Se llama a `SQLExecute()` o `SQLExecDirect()` para *StatementHandle* y se devuelve SQL_NEED_DATA. Se llama a esta función antes de que se envíen datos para todos los parámetros o columnas de datos en ejecución.
HY013	Error inesperado de gestión de la memoria.	DB2 CLI no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY090	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El valor especificado para el argumento *BufferLength* es menor que 0.
HY092	Tipo de opción fuera de rango.	El valor especificado para el argumento *Attribute* no es válido para esta versión de la CLI de DB2.
HY109	Posición no válida del cursor.	El argumento *Attribute* es SQL_ATTR_ROW_NUMBER y la fila se había suprimido o no se había podido recuperar.
HYC00	Controlador no apropiado.	El valor especificado para el argumento *Attribute* es un atributo válido para esta versión de la CLI de DB2, pero la fuente de datos no lo soporta.

Restricciones:

Ninguna.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLSetConnectAttr—Establecer opciones referentes a una conexión" en la página 272

• "SQLSetStmtAttr—Establecer opciones referentes a una sentencia" en la página 276

SQLNumParams - Obtener número de parámetros en una sentencia SQL

Finalidad:

Especificación:	CLI de DB2 2.1	ODBC 1.0	

SQLNumParams() devuelve el número de marcadores de parámetro de una sentencia de SQL.

Sintaxis:

```
SQLRETURN   SQLNumParams     (SQLHSTMT          StatementHandle,
                              SQLSMALLINT  FAR  *ParameterCountPtr);
```

Argumentos de la función:

Tabla 76. Argumentos de SQLNumParams

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	Entrada	Descriptor de contexto de sentencia.
SQLSMALLINT	*ParameterCountPtr*	Salida	Número de parámetros de la sentencia.

Uso:

Esta función sólo puede llamarse después de que se haya preparado la sentencia asociada con *StatementHandle*. Si la sentencia no contiene ningún marcador de parámetro, *ParameterCountPtr* se establece en 0.

Una aplicación puede llamar a esta función para determinar el número de llamadas SQLBindParameter() necesarias para la sentencia SQL asociada con el descriptor de sentencia.

Códigos de retorno:
- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 77. SQLNumParams SQLSTATEs

SQLSTATE	Descripción	Explicación
HY001	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY010	Error de secuencia de función.	Esta función se llamó antes de que se llamara SQLPrepare() para la *StatementHandle* especificada.
HY013	Error inesperado de gestión de la memoria.	DB2 CLI no pudo acceder a la memoria necesaria para ejecutar o finalizar la función.

Restricciones:

Ninguna.

Consulta relacionada:

- "SQLBindParameter—Enlazar un marcador de parámetro a un almacenamiento intermedio" en la página 207
- "SQLPrepare—Preparar una sentencia" en la página 266

SQLNumResultCols—Obtener número de columnas resultantes

Finalidad:

Especificación:	**CLI de DB2** 1.1	**ODBC** 1.0	

`SQLNumResultCols()` devuelve el número de columnas del conjunto resultante asociado al descriptor de sentencia de entrada.

Se debe invocar `SQLPrepare()` o `SQLExecDirect()` antes de invocar esta función.

Después de llamar a esta función, puede llamar a `SQLColAttribute()` o una de las funciones de enlace de columnas.

Sintaxis:

```
SQLRETURN   SQLNumResultCols (SQLHSTMT          StatementHandle,   /* hstmt */
                              SQLSMALLINT  FAR  *ColumnCountPtr);  /* pccol */
```

Argumentos de la función:

Tabla 78. Argumentos de SQLNumResultCols

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia.
SQLSMALLINT *	*ColumnCountPtr*	salida	Número de columnas del conjunto resultante.

Uso:

Esta función establece en 0 el argumento de salida si la última sentencia o función ejecutada para el descriptor de sentencia de entrada no generó un conjunto resultante.

Códigos de retorno:

- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 79. SQLSTATE de SQLNumResultCols

SQLSTATE	Descripción	Explicación
40003 **08**S01	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
58004	Error inesperado del sistema.	Error no recuperable del sistema.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY010	Error de secuencia de función.	Se llama a la función antes de llamar a `SQLPrepare()` o `SQLExecDirect()` para el descriptor de sentencia (*StatementHandle*).
HY013	Error inesperado de gestión de la memoria.	CLI de DB2 no puede acceder a la memoria necesaria para ejecutar o finalizar la función.

Restricciones:

Ninguna.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLBindCol—Enlazar una columna a una variable de aplicación" en la página 203
- "SQLDescribeCol—Devolver un conjunto de atributos de una columna" en la página 220
- "SQLExecDirect—Ejecutar una sentencia directamente" en la página 225
- "SQLGetData—Obtener datos de una columna" en la página 250

SQLPrepare—Preparar una sentencia

Finalidad:

Especificación:	CLI de DB2 1.1	ODBC 1.0	ISO CLI

`SQLPrepare()` asocia una sentencia de SQL con el descriptor de sentencia de entrada y envía la sentencia al DBMS para que se prepare. La aplicación puede hacer referencia a esta sentencia preparada pasando el descriptor de sentencia a otras funciones.

Si previamente se utiliza el descriptor de contexto de sentencia con una sentencia de consulta (o cualquier función que devuelva un conjunto resultante), se debe llamar a `SQLFreeStmt()` antes de llamar a `SQLPrepare()`.

Sintaxis:

```
SQLRETURN   SQLPrepare       (SQLHSTMT          StatementHandle,   /* hstmt */
                              SQLCHAR       FAR *StatementText,   /* szSqlStr */
                              SQLINTEGER         TextLength);     /* cbSqlStr */
```

Argumentos de la función:

Tabla 80. Argumentos de SQLPrepare

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia.
SQLCHAR	*StatementText*	entrada	Serie de la sentencia de SQL
SQLINTEGER	*TextLength*	entrada	Longitud del contenido del argumento *StatementText*. Se debe establecer en la longitud exacta de la sentencia de SQL contenida en *szSqlstr*, o en SQL_NTS si el texto de la sentencia termina en nulo.

Uso:

Después de preparar una sentencia utilizando `SQLPrepare()`, la aplicación puede solicitar información sobre el formato del conjunto resultante (si la sentencia es una consulta), llamando a una de estas funciones:
- `SQLNumResultCols()`
- `SQLDescribeCol()`

La serie de la sentencia de SQL puede contener marcadores de parámetros. Un marcador de parámetro se representa mediante un símbolo ? y se utiliza para indicar una posición en la sentencia donde se colocará un valor proporcionado por la aplicación cuando se llame a `SQLExecute()`. La función de enlace de parámetros, `SQLBindParameter()`, enlaza (asocia) valores de la aplicación con cada marcador de parámetro e indica si se debe realizar alguna conversión de datos al transferir los datos.

Se deben enlazar todos los parámetros antes de invocar `SQLExecute()`. Si desea obtener más información, consulte el apartado "SQLExecute—Ejecutar una sentencia" en la página 227.

Consulte la sección sobre la sentencia PREPARE del manual *DB2 Universal Database Consulta de SQL* para obtener información sobre reglas referentes a marcadores de parámetros.

Después de procesar los resultados de la llamada a `SQLExecute()`, la aplicación puede ejecutar de nuevo la sentencia con valores de parámetros nuevos (o los mismos).

Códigos de retorno:
- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 81. SQLSTATE de SQLPrepare

SQLSTATE	Descripción	Explicación
42nnn	Error de sintaxis.	Los SQLSTATE **42**nnn indican que hay varios problemas de sintaxis o de acceso en la sentencia. Los caracteres nnn representan cualquier SQLSTATE con ese código de clase. Ejemplo: **42**nnn representa cualquier SQLSTATE perteneciente a la clase **42**.
58004	Error inesperado del sistema.	Error no recuperable del sistema.
HY001	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY009	Valor no válido de argumento.	*StatementText* es un puntero nulo.
HY013	Error inesperado de gestión de la memoria.	DB2 CLI no puede acceder a la memoria necesaria para ejecutar o finalizar la función.
HY014	No hay más descriptores de contexto.	DB2 CLI no puede asignar un descriptor de contexto debido a los recursos internos.
HY090	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El argumento *TextLength* es menor que uno pero no igual a SQL_NTS.

Restricciones:

Ninguna.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLBindParameter—Enlazar un marcador de parámetro a un almacenamiento intermedio" en la página 207
- "SQLDescribeCol—Devolver un conjunto de atributos de una columna" en la página 220
- "SQLExecDirect—Ejecutar una sentencia directamente" en la página 225
- "SQLExecute—Ejecutar una sentencia" en la página 227
- "SQLNumResultCols—Obtener número de columnas resultantes" en la página 265

SQLPrimaryKeys—Obtener columnas de clave primaria de una tabla

Finalidad:

Especificación:	**CLI de DB2** 2.1	**ODBC** 1.0	

`SQLPrimaryKeys()` devuelve una lista de nombres de columna que abarcan la clave primaria de una tabla. La información se devuelve en un conjunto resultante de SQL, que se puede recuperar utilizando las mismas funciones que se utilizan para

procesar un conjunto resultante generado por una consulta. No se tienen en cuenta los parámetros *CatalogName*, *NameLength1*, *SchemaName* y *NameLength2*. Las columnas 1, 2 y 6 del conjunto resultante devuelto son siempre una serie de longitud cero.

Sintaxis:

```
SQLRETURN   SQLPrimaryKeys  (
                  SQLHSTMT          StatementHandle,  /* hstmt */
                  SQLCHAR      FAR  *CatalogName,     /* szCatalogName */
                  SQLSMALLINT       NameLength1,      /* cbCatalogName */
                  SQLCHAR      FAR  *SchemaName,      /* szSchemaName */
                  SQLSMALLINT       NameLength2,      /* cbSchemaName */
                  SQLCHAR      FAR  *TableName,       /* szTableName */
                  SQLSMALLINT       NameLength3);     /* cbTableName */
```

Argumentos de la función:

Tabla 82. Argumentos de SQLPrimaryKeys

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia.
SQLCHAR*	*CatalogName*	entrada	Calificador de catálogo de un nombre de tabla que consta de tres partes. DB2 Everyplace no tiene en cuenta este campo.
SQLSMALLINT	*NameLength1*	entrada	Longitud de *CatalogName*. DB2 Everyplace no tiene en cuenta este campo.
SQLCHAR*	*SchemaName*	entrada	Calificador de esquema del nombre de tabla. DB2 Everyplace no tiene en cuenta este campo.
SQLSMALLINT	*NameLength2*	entrada	Longitud de *SchemaName*. DB2 Everyplace no tiene en cuenta este campo.
SQLCHAR*	*TableName*	entrada	Nombre de tabla.
SQLSMALLINT	*NameLength3*	entrada	Longitud de *TableName*.

Uso:

`SQLPrimaryKeys()` devuelve las columnas de clave primaria de una tabla individual. No se pueden utilizar patrones de búsqueda para especificar el nombre de tabla.

Si la tabla especificada no contiene una clave primaria, se devuelve un conjunto resultante vacío.

Las llamadas a `SQLPrimaryKeys()` se correlacionan, en muchos casos, formando consultas complejas y caras sobre el catálogo del sistema.

Aunque en futuros releases se pueden añadir nuevas columnas y modificarse los nombres de las columnas existentes, la posición de las columnas actuales no cambia.

El conjunto resultante contiene esta columnas ordenadas por TABLE_NAME y ORDINAL_POSITION:

Columna 1 TABLE_CAT (VARCHAR(128))
Siempre es una serie de longitud cero.

Columna 2 TABLE_SCHEM (VARCHAR(128))
Siempre es una serie de longitud cero.

Columna 3 TABLE_NAME (VARCHAR(128) no nulo)
Nombre de la tabla especificada.

Columna 4 COLUMN_NAME (VARCHAR(128) no nulo)
Nombre de la columna de clave primaria.

Columna 5 ORDINAL_POSITION (SMALLINT no nulo)
Número de secuencia de columna en la clave primaria, comenzando en el uno.

Columna 6 PK_NAME (VARCHAR(128))
Siempre es una serie de longitud cero.

Los nombres de columna utilizados por DB2 CLI/ODBC siguen el estilo de especificación de X/Open CLI CAE.

Códigos de retorno:
- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 83. SQLSTATE de SQLPrimaryKey

SQLSTATE	Descripción	Explicación
24000	Estado no válido del cursor.	Ya hay un cursor abierto para el descriptor de contexto de sentencia.
40003 **08**S01	Error en el enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos se interrumpió antes de finalizar la función.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY010	Error de secuencia de función.	Se llama a la función mientras se produce una operación de datos en ejecución (`SQLPrepare()` o `SQLExecDirect()`).
HY014	No hay más descriptores de contexto.	CLI de DB2 no puede asignar un descriptor de contexto debido a recursos internos.
HY090	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El valor de uno de los argumentos de longitud de nombre es menor que 0, pero no igual a SQL_NTS.

Restricciones:

Utilice las llamadas a `SQLPrimaryKeys()` con moderación y guarde los resultados en lugar de repetir llamadas.

Consulta relacionada:
- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLForeignKeys—Obtener la lista de columnas de clave foránea" en la página 238

SQLRowCount—Obtener número de filas

Finalidad:

Especificación:	CLI de DB2 1.1	ODBC 1.0	ISO CLI

SQLRowCount() devuelve el número de filas de una tabla que se han visto afectadas por una sentencia UPDATE, INSERT, DELETE o SELECT con cursor desplazable ejecutada sobre la tabla.

Se debe invocar SQLExecute() o SQLExecDirect() antes de invocar esta función.

Sintaxis:

```
SQLRETURN   SQLRowCount      (SQLHSTMT  StatementHandle,   /* hstmt */
                              SQLINTEGER   FAR  *RowCountPtr);     /* pcrow */
```

Argumentos de la función:

Tabla 84. Argumentos de SQLRowCount

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia.
SQLINTEGER	*RowCountPtr*	salida	Puntero que apunta al lugar donde se almacena el número de filas afectadas por la sentencia.

Uso:

Si la última sentencia ejecutada a la que hace referencia el descriptor de contexto de sentencia de entrada no es una sentencia UPDATE, INSERT ni DELETE, o si la sentencia no se ejecutó satisfactoriamente, la función establece el contenido de *RowCountPtr* en -1.

El contaje no incluye las filas de otras tablas que pueden haber sido afectadas por la sentencia.

Códigos de retorno:
- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 85. SQLSTATE de SQLRowCount

SQLSTATE	Descripción	Explicación
40003 **08**S01	Anomalía de enlace de comunicaciones.	El enlace de comunicaciones entre la aplicación y la fuente de datos ha fallado antes de que se completara la función.
58004	Anomalía inesperada del sistema.	Error no recuperable del sistema.

Tabla 85. SQLSTATE de SQLRowCount (continuación)

SQLSTATE	Descripción	Explicación
HY001	Anomalía de asignación de memoria.	La CLI de DB2 no puede asignar la memoria necesaria para soportar la ejecución o realización de la función. Es probable que la memoria de nivel de proceso se haya agotado para el proceso de aplicaciones. Consulte la configuración del sistema operativo para obtener información sobre las limitaciones de memoria de nivel de proceso.
HY010	Error de secuencia de función.	Se llama a la función antes de llamar a `SQLExecute()` o `SQLExecDirect()` para el descriptor de contexto de sentencia (*StatementHandle*).
HY013	Error inesperado de manejo de memoria.	La CLI de DB2 no ha podido acceder a la memoria necesaria para soportar la ejecución o realización de la función.

Restricciones:

Ninguna.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLExecDirect—Ejecutar una sentencia directamente" en la página 225
- "SQLExecute—Ejecutar una sentencia" en la página 227
- "SQLNumResultCols—Obtener número de columnas resultantes" en la página 265

SQLSetConnectAttr—Establecer opciones referentes a una conexión

Finalidad:

Especificación:	CLI de DB2	ODBC 1.0	ISO CLI

`SQLSetConnectAttr()` establece opciones relacionadas con una conexión

Sintaxis:

```
SQLRETURN   SQLSetConnectAttr   (SQLHDBC       ConnectionHandle,
                                 SQLINTEGER    Attribute,
                                 SQLPOINTER    ValuePtr,
                                 SQLINTEGER    StringLength);
```

Argumentos de la función:

Tabla 86. Argumentos de SQLSetConnectAttr

Tipo de datos	Argumento	Uso	Descripción
SQLHDBC	*ConnectionHandle*	entrada	Descriptor de contexto de conexión.
SQLINTEGER	*Attribute*	entrada	Opción para definir.

Tabla 86. Argumentos de SQLSetConnectAttr (continuación)

Tipo de datos	Argumento	Uso	Descripción
SQLPOINTER	*ValuePtr*	entrada	Si *Attribute* es un atributo definido por ODBC y *ValuePtr* apunta a una serie de caracteres o almacenamiento intermedio binario, este argumento debe ser igual a la longitud de *ValuePtr*. Si *Attribute* es un atributo definido por ODBC y ValuePtr es un entero, no se tiene en cuenta *StringLength*. Si *Attribute* es un atributo de CLI de DB2, la aplicación indica la naturaleza del atributo estableciendo el argumento *StringLength*. *StringLength* puede tener los valores siguientes: • Si *ValuePtr* es un puntero que apunta a una serie de caracteres, *StringLength* es la longitud de la serie o SQL_NTS. • Si *ValuePtr* es un puntero que apunta a un almacenamiento intermedio binario, la aplicación coloca el resultado de la macro SQL_LEN_BINARY_ATTR(length) en *StringLength*. Esto coloca un valor negativo en *StringLength*. • Si *ValuePtr* es un puntero que apunta a un valor distinto de una serie de caracteres o serie binaria, el valor de *StringLength* debe ser SQL_IS_POINTER. • Si *ValuePtr* contiene un valor de longitud fija, *StringLength* es SQL_IS_INTEGER o SQL_IS_UINTEGER, según corresponda.
SQLINTEGER	*StringLength*	entrada	Si *ValuePtr* apunta a una serie de caracteres o almacenamiento intermedio binario, este argumento debe ser igual a la longitud de *ValuePtr*. Si *ValuePtr* es un puntero, pero no apunta a una serie de caracteres ni a un almacenamiento intermedio binario, *StringLength* debe tener el valor SQL_IS_POINTER. Si *ValuePtr* no es un puntero, el valor de *StringLength* debe ser SQL_IS_NOT_POINTER.

Uso:

Los atributos de una conexión permanecen vigentes hasta que son modificados por otra llamada a `SQLSetConnectAttr()` o hasta que la conexión se descarta mediante una llamada a `SQLDisconnect()`.

`SQLSetConnectAttr()` acepta información sobre atributos en uno de dos formatos diferentes: una serie de caracteres terminada en nulo o un valor entero de 32 bits.

El formato del atributo está indicado en la descripción del atributo. Las series de caracteres a las que apunta el argumento *ValuePtr* de `SQLSetConnectAttr()` tienen una longitud igual a *StringLength*.

Atributos de conexión:

A continuación se muestran los atributos definidos actualmente.

SQL_ATTR_AUTOCOMMIT (DB2 CLI/ODBC)
Valor entero de 32 bits que especifica el tipo de modalidad. Los valores soportados son:

- SQL_AUTOCOMMIT_ON = Cada sentencia se confirma automáticamente. Es el valor por omisión.

  En la modalidad de confirmación automática, todas las actualizaciones hechas por una sentencia pasan automáticamente a ser permanentes una vez ejecutada la sentencia. La modalidad de confirmación automática es el comportamiento por omisión. Por omisión, no está habilitado el soporte de transacciones, y además, no se garantiza la atomicidad en el ámbito de sentencia. Por ejemplo, la siguiente sentencia UPDATE puede fallar durante el proceso y puede que sólo se actualice un subconjunto de filas:

  ```
  UPDATE T SET A = A + 1
  ```

  Pueden haber varias razones por las que fallen las operaciones update/delete/insert. Por ejemplo, puede violarse una restricción de comprobación durante una actualización. Como consecuencia, puede haberse actualizado adecuadamente una parte de la tabla, mientras que el resto no, y los cambios no se pueden retrotraer.
- SQL_AUTOCOMMIT_OFF = La aplicación debe manual y explícitamente comprometer o retrotraer una transacción. El compromiso o la retrotracción de una transacción se lleva a cabo invocando `SQLEndTran()`. Para obtener más información sobre la utilización de `SQLEndTran()`, consulte “SQLEndTran—Solicitar COMMIT o ROLLBACK” en la página 223.

  En la modalidad de confirmación manual, las transacciones se inician implícitamente con el primer acceso a la base de datos utilizando SQLPrepare() y SQLExecDirect(). En este momento ya ha comenzado una transacción, aunque haya fallado la llamada. La transacción termina cuando se utiliza SQLEndTran() para retrotraer (ROLLBACK) o confirmar (COMMIT) la transacción.

  En la modalidad de confirmación manual, las transacciones pueden emitir cualquier sentencia SQL, incluyendo DDL y DML (por ejemplo, las sentencias CREATE TABLE o UPDATE)

SQL_ATTR_CONNECTION_DEAD (DB2 CLI/ODBC)
Un valor entero de 32 bits READ ONLY que indica si la conexión está o no está aún activa. DB2 CLI devolverá uno de los valores siguientes:

- SQL_CD_FALSE - la conexión sigue estando activa.
- SQL_CD_TRUE - la conexión está inactiva.

SQL_ATTR_LOGIN_TIMEOUT (DB2 CLI/ODBC)
Valor entero de 32 bits que corresponde al número de segundos que se deberá esperar a que finalice una petición de inicio de sesión antes de devolver el control a la aplicación.

SQL_ATTR_FILENAME_FORMAT (DB2 Everyplace)

Un entero de 32 bits especifica si el motor de bases de datos DB2e debe crear nombres de archivo en formato largo o de 8.3. Las aplicaciones sólo tienen permitido cambiar el formato de nombres de archivo si no existe ningún archivo de catálogo en la vía de acceso conectada cuando se invoca a SQLSetConnectAttr. Se devolverá SQL_ERROR con SQLState HY000 si se deniega un cambio del formato de nombres de archivo debido a que previamente existían archivos de catálogo. Por ejemplo, si una aplicación conecta con una vía de acceso en que ya existe archivos de catálogo de DB2 Everyplace, cualquier intento de cambiar el formato de nombres de archivo fallará. Si una aplicación conecta con una vía de acceso en que no existe ningún archivo de catálogo e intenta cambiar el formato de nombres de archivo después de la primera sentencia CREATE TABLE, SQLSetConnectAttr también devolverá SQL_ERROR. Esto es debido a que se crean archivos de catálogo durante la realmente primera sentencia CREATE TABLE, y no está permitido cambiar el formato de nombres de archivo después de la creación de archivos de catálogo. El formato de nombres de archivo por omisión depende de la plataforma. Actualmente, SQL_FILENAME_FORMAT_LONG es el valor por omisión para todas las plataformas soportadas.

Valores de atributos:

```
 SQL_FILENAME_FORMAT_LONG - los archivos se
crearán en formato de nombre de archivo largo.
```

```
 SQL_FILENAME_FORMAT_83- los archivos se crearán en formato de
nombre de archivo 8.3.
```

Códigos de retorno:

- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 87. SQLSTATEs de SQLSetConnectAttr

SQLSTATE	Descripción	Explicación
HY000	Error general.	El formato de nombres de archivo no se puede cambiar.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY014	No hay más descriptores de contexto.	CLI de DB2 no pudo asignar un descriptor de contexto debido a recursos internos.
HY090	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El valor de uno de los argumentos de longitud de nombre era menor que 0, pero no era igual a SQL_NTS.

Restricciones:

- El número de tablas que se pueden actualizar en una transacción está limitado. DB2 Everyplace permite un número máximo de 256 archivos abiertos dentro de una transacción, suponiendo que el sistema operativo también permita este número de archivos abiertos. Normalmente esto significa que puedan actualizarse aproximadamente 100 tablas. El número de tablas depende del uso

de índices y del número de descriptores de sentencia. Cuanto mayor sea el número de descriptores de contexto de sentencia activos, menos tablas se pueden actualizar potencialmente. Cada tabla se contabiliza una sola vez, aunque se acceda a ella y/o se actualice varias veces en una transacción.

- Se han añadido transacciones a DB2 Everyplace para permitir una actualización e inserción coherentes de varios registros asociados en varias tablas. Los cambios se escriben en las tablas de datos una vez que la aplicación confirme la transacción.
- Si la aplicación ha finalizado prematuramente sin confirmar la transacción actual, las actualizaciones contenidas en esa transacción se retrotraen automáticamente.
- Una vez que SQLEndTran vuelve, la transacción se confirma o se retrotrae.
- Cuando una aplicación conecta con una base de datos y ésta termina prematuramente (durante una transacción activa), la transacción se recupera. La base de datos recupera transacciones utilizando la lógica siguiente:
  - Si la transacción no ha finalizado, la base de datos *no* se actualiza.
  - Si la transacción ha finalizado, la base de datos *sí* que se actualiza con la información proporcionada por la transacción.
  - Si la recuperación se interrumpe, se realiza la acción apropiada en la próxima conexión.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194
- "SQLEndTran—Solicitar COMMIT o ROLLBACK" en la página 223

SQLSetStmtAttr—Establecer opciones referentes a una sentencia

Finalidad:

Especificación:	CLI de DB2 2.1	ODBC 1.0	ISO CLI

`SQLSetStmtAttr()` define opciones relacionadas con una sentencia.

Sintaxis:

```
SQLRETURN   SQLSetStmtAttr   (SQLHSTMT          StatementHandle,
                              SQLINTEGER        Attribute,
                              SQLPOINTER        ValuePtr,
                              SQLINTEGER        StringLength);
```

Argumentos de la función:

Tabla 88. Argumentos de SQLSetStmtAttr

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	entrada	Descriptor de contexto de sentencia.
SQLINTEGER	*Attribute*	entrada	Opción para definir.

Tabla 88. Argumentos de SQLSetStmtAttr (continuación)

Tipo de datos	Argumento	Uso	Descripción
SQLPOINTER	*ValuePtr*	entrada	Si *Attribute* es un atributo definido por ODBC y *ValuePtr* apunta a una serie de caracteres o almacenamiento intermedio binario, este argumento debe ser igual a la longitud de **ValuePtr*. Si *Attribute* es un atributo definido por ODBC y ValuePtr es un entero, no se tiene en cuenta *StringLength*. Si *Attribute* es un atributo de CLI de DB2, la aplicación indica la naturaleza del atributo estableciendo el argumento *StringLength*. *StringLength* puede tener los valores siguientes: • Si *ValuePtr* es un puntero que apunta a una serie de caracteres, *StringLength* es la longitud de la serie o SQL_NTS. • Si *ValuePtr* es un puntero que apunta a un almacenamiento intermedio binario, la aplicación coloca el resultado de la macro SQL_LEN_BINARY_ATTR(length) en *StringLength*. Esto coloca un valor negativo en *StringLength*. • Si *ValuePtr* es un puntero que apunta a un valor distinto de una serie de caracteres o serie binaria, el valor de *StringLength* debe ser SQL_IS_POINTER. • Si *ValuePtr* contiene un valor de longitud fija, *StringLength* es SQL_IS_INTEGER o SQL_IS_UINTEGER.
SQLINTEGER	*StringLength*	entrada	Si *ValuePtr* apunta a una serie de caracteres o almacenamiento intermedio binario, este argumento debe ser igual a la longitud de *ValuePtr*. Si *ValuePtr* es un puntero pero no apunta a una serie de caracteres o almacenamiento intermedio binario, *StringLength* debe tener el valor SQL_IS_POINTER. Si *ValuePtr* no es un puntero, *StringLength* debe ser SQL_IS_NOT_POINTER.

Uso:

Los atributos de una sentencia permanecen vigentes hasta que son modificados por otra llamada a `SQLSetStmtAttr()` o hasta que la sentencia se descarta mediante una llamada a `SQLFreeHandle()`. Invocar `SQLFreeStmt()` con las opciones SQL_CLOSE, SQL_UNBIND o SQL_RESET_PARAMS no restaura los atributos de una sentencia.

Algunos atributos de sentencia dan soporte a la sustitución de un valor similar si la fuente de datos no da soporte al valor especificado en *ValuePtr*. En tales casos, CLI de DB2 devuelve SQL_SUCCESS_WITH_INFO y SQLSTATE 01S02 (Valor de opción cambiado). Por ejemplo, si Attribute es SQL_ATTR_CONCURRENCY, *ValuePtr* es SQL_CONCUR_ROWVER, y la fuente de datos no da soporte a este

valor, CLI de DB2 sustituye SQL_CONCUR_VALUES y devuelve SQL_SUCCESS_WITH_INFO. Para determinar el valor sustituido, la aplicación invoca `SQLGetStmtAttr()`. El formato de la información establecida con *ValuePtr* depende del valor especificado para Attribute.

`SQLSetStmtAttr()` acepta información sobre atributos en uno de dos formatos diferentes: una serie de caracteres terminada en nulo o un valor entero de 32 bits. El formato del atributo está indicado en la descripción del atributo. Este formato es aplicable a la información devuelta para cada atributo en `SQLGetStmtAttr()`. Las series de caracteres a las que apunta el argumento *ValuePtr* de `SQLSetStmtAttr()` tienen una longitud igual a *StringLength*.

El bit de modificación:

DB2 Everyplace utiliza un bit de modificación para hacer un seguimiento de los cambios efectuados en un registro. El comportamiento del bit de modificación está determinado por los atributos de sentencia SQL_ATTR_DELETE_MODE, SQL_ATTR_READ_MODE y SQL_ATTR_DIRTYBIT_SET_MODE. La tabla siguiente muestra los estados del bit de modificación después de que se realicen ciertas operaciones de base de datos en un registro. La tabla asume que el parámetro SQL_ATTR_DIRTYBIT_SET_MODE se establece en SQL_DIRTYBIT_SET_BY_SYSTEM con el bit de modificación mantenido por el sistema.

Tabla 89. Estados del bit de modificación de DB2 Everyplace

Acciones en un registro	Estado del bit de modificación
Estado limpio (0) y a continuación INSERTAR	INSERTAR
Estado limpio (0) y a continuación SUPRIMIR	SUPRIMIR
Estado limpio (0) y a continuación ACTUALIZAR	ACTUALIZAR
SUPRIMIR y a continuación INSERTAR	ACTUALIZAR
SUPRIMIR y a continuación SUPRIMIR	No aplicable
SUPRIMIR y a continuación ACTUALIZAR	No aplicable
INSERTAR y a continuación INSERTAR	No aplicable
INSERTAR y a continuación SUPRIMIR	Eliminación física de registro
INSERTAR y a continuación ACTUALIZAR	INSERTAR
ACTUALIZAR y a continuación INSERTAR	No aplicable
ACTUALIZAR y a continuación SUPRIMIR	DELETE
ACTUALIZAR y a continuación ACTUALIZAR	ACTUALIZAR

El valor del bit de modificación se puede obtener realizando una consulta a la columna $dirty de una tabla. Por ejemplo, la sentencia siguiente devuelve el bit de modificación y la columna NAME de la tabla PHONEBOOK:

```
SELECT $dirty, NAME from PHONEBOOK
```

El bit de modificación puede tener los valores siguientes.

Tabla 90. Valores del bit de modificación de DB2 Everyplace

Descripción	Valor del bit de modificación
Registro no cambiado (LIMPIO)	0

Tabla 90. Valores del bit de modificación de DB2 Everyplace (continuación)

Descripción	Valor del bit de modificación
Registro suprimido (SUPRIMIR)	1
Registro insertado (INSERTAR)	2
Registro actualizado (ACTUALIZAR)	3

Atributos de sentencia:

A continuación se muestran los atributos definidos actualmente.

SQL_ATTR_CURSOR_SCROLLABLE (CLI de DB2)
Entero de 32 bits que especifica el nivel de soporte que la aplicación requiere. El establecimiento de este atributo afecta a llamadas posteriores a `SQLExecDirect()` y `SQLExecute()`. Los valores soportados son:

- SQL_NONSCROLLABLE

  No se requieren cursores desplazables en el descriptor de contexto de sentencia. Si la aplicación llama a `SQLFetchScroll()` en este descriptor de contexto, el único valor válido para `FetchOrientation()` es SQL_FETCH_NEXT. Es el valor por omisión.
- SQL_SCROLLABLE

  Se requieren cursores desplazables en el descriptor de contexto de sentencia. Cuando se llama a `SQLFetchScroll()`, la aplicación puede especificar cualquier valor válido para `FetchOrientation`, por lo que el cursor se puede posicionar en modalidades distintas de la secuencial.

SQL_ATTR_CURSOR_SENSITIVITY (CLI de DB2)
Valor entero de 32 bits que especifica si un cursor es sensible a la actividad de grabación de otro cursor. Los valores soportados son:

- SQL_UNSPECIFIED

  La actividad de grabación de otros cursores tiene un impacto indefinido sobre el cursor actual. Es el valor por omisión.
- SQL_INSENSITIVE

  La actividad de grabación de otros cursores no tiene ningún impacto sobre el cursor actual.

  Nota: Utilice este valor de atributo con moderación porque puede afectar al rendimiento.

SQL_ATTR_CURSOR_TYPE (CLI de DB2)
Valor entero de 32 bits que especifica el tipo de cursor. Los valores soportados son:

- SQL_CURSOR_FORWARD_ONLY = El cursor sólo se desplaza hacia adelante. Es el valor por omisión.
- SQL_CURSOR_STATIC = Los datos del conjunto resultante son estáticos.

Esta opción no se puede especificar para un cursor abierto.

SQL_ATTR_ROW_ARRAY_SIZE (CLI de DB2)
Valor entero de 32 bits que especifica el número de filas del conjunto de filas. Es el número de filas devueltas por cada llamada a `SQLFetch()` o `SQLFetchScroll()`. El valor por omisión es 1. Si el tamaño de conjunto de filas especificado excede el tamaño máximo de conjunto de filas soportado

por la fuente de datos, CLI de DB2 sustituye ese valor y devuelve SQLSTATE 01S02 (Valor de opción cambiado). Esta opción se puede especificar para un cursor abierto.

SQL_ATTR_ROW_BIND_TYPE (CLI de DB2)
Valor entero de 32 bits que define la orientación del enlace que se debe utilizar cuando se invoca `SQLFetch()` o `SQLFetchScroll()` para la sentencia asociada. El enlace basado en columnas se selecciona proporcionando la constante definida SQL_BIND_BY_COLUMN en *ValuePtr*. La longitud especificada en *ValuePtr* debe incluir espacio para todas las columnas enlazadas y el espacio de relleno necesario de la estructura o almacenamiento intermedio para asegurar que, cuando la dirección de una columna enlazada se incremente con la longitud especificada, el resultado apuntará al comienzo de la misma columna en la fila siguiente. Cuando se utiliza el operador **sizeof** con estructuras o uniones en C de ANSI, este comportamiento está asegurado. El enlace basado en columnas es la orientación predefinida del enlace para SQLFetchScroll().

SQL_ATTR_ROW_NUMBER (CLI de DB2)
Valor entero de 32 bits que es el número de la fila actual en el conjunto resultante entero. Si no se puede determinar el número de la fila actual o si no existe ninguna fila actual, CLI de DB2 devuelve 0. Este atributo se puede recuperar mediante una llamada a `SQLGetStmtAttr()`, pero no se puede establecer mediante una llamada a `SQLSetStmtAttr()`.

SQL_ATTR_ROW_STATUS_PTR (CLI de DB2)
Valor entero de 16 bits, sin signo, que apunta a una matriz de valores UWORD que contiene valores de estado de filas después de una llamada a `SQLFetch()` o `SQLFetchScroll()`. La matriz tiene tantos elementos como filas hay en el conjunto de filas. Este atributo de sentencia se puede definir como puntero nulo, en cuyo caso CLI de DB2 no devuelve valores de estado para filas. Este atributo se puede definir en cualquier momento, pero el nuevo valor no se utiliza hasta la siguiente llamada a `SQLFetch()` o `SQLFetchScroll()`.

SQL_ATTR_ROWS_FETCHED_PTR (CLI de DB2)
Valor entero de 32 bits, sin signo, que apunta a un almacenamiento intermedio en el que devolver el número de filas recuperadas después de una llamada a `SQLFetch()` o `SQLFetchScroll()`.

SQL_ATTR_DELETE_MODE (DB2 Everyplace)
Los valores soportados son:

- SQL_DELETE_MARK_ONLY

  Este es el valor por omisión del sistema. Cuando se ejecute una sentencia de SQL de supresión, los registros sólo están señalados como "delete" (suprimir). Su contenido sigue pudiéndose leer en el caso de que se establezca SQL_READ_INCLUDE_MARKED_DELETE.

- SQL_DELETE_PHYSICAL_REMOVE

  Una sentencia de supresión de SQL elimina físicamente los registros que satisfagan la condición de la cláusula WHERE, sin tener en cuenta su bit de modificación.

Por ejemplo, utilice la sintaxis siguiente para eliminar físicamente algunos registros ignorando el estado de los bits de parada:

```
SQLSetStmtAttr (stmt, SQL_ATTR_DELETE_MODE,
SQL_DELETE_PHYSICAL_REMOVE, 0)
```

A continuación, ejecute la sentencia de SQL siguiente para suprimir todos los registros de la tabla T donde X no sea igual a 0:

```
DELETE T WHERE X<>0
```

SQL_ATTR_DIRTYBIT_SET_MODE (DB2 Everyplace)

Valor entero de 32 bits que especifica el tipo de cursor. Los valores soportados son:

- SQL_DIRTYBIT_SET_BY_SYSTEM

  Este es el valor por omisión del sistema. Todo registro que se inserte, actualice o suprima tendrá un bit de modificación establecido, respectivamente, en INSERT, UPDATE o DELETE. No se permite ninguna operación UPDATE sobre la columna $dirty cuando esté establecido SQL_DIRTYBIT_SET_BY_SYSTEM.

- SQL_DIRTYBIT_SET_BY_APPLICATION

  La aplicación es responsable de establecer el bit de modificación al insertar, actualizar o eliminar registros. La semántica de cada una de las operaciones es como sigue:

  UPDATE
  El sistema establece el bit de modificación exactamente como lo ha especificado la aplicación. Por ejemplo, si una aplicación ejecuta la sentencia siguiente, todos los registros de la tabla se restablecen a 0 (CLEAN):

  ```
  UPDATE T SET $dirty=0 WHERE $dirty>0
  ```

  INSERT
  El bit de modificación del registro que recién insertado se establece en CLEAN.

  DELETE
  Si se establece SQL_DELETE_PHYSICAL_REMOVE, una operación DELETE elimina físicamente registros de la base de datos. De lo contrario, los valores de la columna $dirty se establecen en DELETE y los registros permanecen en la base de datos.

Por ejemplo, para limpiar el bit de modificación de un registro utilice la sentencia siguiente:

```
SQLSetStmtAttr (stmt, SQL_ATTR_DIRTYBIT_SET_MODE,
SQL_DIRTYBIT_SET_BY_APPLICATION, 0)
```

A continuación ejecute la sentencia de SQL siguiente:

```
UPDATE T SET $DIRTY=0 WHERE $DIRTY>0
```

En general, las aplicaciones pueden establecer SQL_DIRTYBIT_SET_BY_APPLICATION cuando los bits de parada no se necesitan para hacer el seguimiento de las actualizaciones de base de datos mediante usuarios finales.

SQL_ATTR_READ_MODE (DB2 Everyplace)

Valor entero de 32 bits que especifica el tipo de cursor. Los valores soportados son:

- SQL_READ_EXCLUDE_MARKED_DELETE

  Este es el valor por omisión del sistema. Todos los registros con bit de modificación establecidos en ″delete″ (suprimir) se ocultan de SQL.

- SQL_READ_INCLUDE_MARKED_DELETE

Una vez establecidos, los registros con el bit de modificación establecido en DELETE (suprimir) se pueden ver desde sentencias SELECT de SQL. Las aplicaciones pueden distinguir los registros suprimidos de otros registros buscando un registro en el bit de modificación.

Por ejemplo, utilice la sentencia siguiente para leer todos los registros que tengan el bit de modificación establecido, incluyendo aquellos que tengan un bit de modificación marcado como DELETE:

```
SQLSetStmtAttr
(stmt, SQL_ATTR_READ_MODE, SQL_READ_INCLUDE_MARKED_DELETE, 0)
```

A continuación ejecute la sentencia de SQL siguiente para recuperar todos los registros:

```
SELECT * FROM T WHERE $dirty<>0
```

SQL_ATTR_REORG_MODE (DB2 Everyplace)

Valor entero de 32 bits que especifica si se realiza una reorganización automática de la base de datos en tablas creadas por el usuario y si se permiten sentencias explícitas REORG de SQL. Los valores soportados son:

- SQL_REORG_ENABLED - Es el valor por omisión del sistema. La base de datos puede ser reorganizada por DB2 Everyplace o explícitamente por el usuario mediante una sentencia REORG de SQL.
- SQL_REORG_DISABLED - Las sentencias REORG de SQL están restringidas y la reorganización automática de tablas creadas por el usuario está inhabilitada.

Esta opción no se puede especificar para un cursor abierto.

Códigos de retorno:

- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 91. SQLSTATE de SQLSetStmtAttr

SQLSTATE	Descripción	Explicación
24000	Estado no válido del cursor.	Ya hay un cursor abierto para el descriptor de contexto de sentencia.
HY001	Error de asignación de memoria.	CLI de DB2 no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY010	Error de secuencia de función.	Se llama a la función mientras se produce una operación de datos en ejecución (`SQLPrepare()` o `SQLExecDirect()`). Se llamó a la función mientras estaba en curso una operación BEGIN COMPOUND y END COMPOUND de SQL.
HY014	No hay más descriptores de contexto.	CLI de DB2 no puede asignar un descriptor de contexto debido a recursos internos.
HY090	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El valor de uno de los argumentos de longitud de nombre es menor que 0, pero no igual a SQL_NTS.

Restricciones:

Ninguna.

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194

SQLTables - Obtener información de tabla

Finalidad:

Especificación:	**CLI de DB2** 2.1	**ODBC** 1.0	

`SQLTables()` devuelve una lista de nombres de tabla e información asociada almacenada en el catálogo del sistema de la fuente de datos conectada. La lista de nombres de tabla se devuelve como un conjunto de resultados, que se puede recuperar utilizando las mismas funciones que se utilizan para procesar un conjunto de resultados generado por una consulta.

Sintaxis:

```
SQLRETURN   SQLTables        (
                SQLHSTMT          StatementHandle,   /* hstmt */
                SQLCHAR      FAR  *CatalogName,      /* szCatalogName */
                SQLSMALLINT       NameLength1,       /* cbCatalogName */
                SQLCHAR      FAR  *SchemaName,       /* szSchemaName */
                SQLSMALLINT       NameLength2,       /* cbSchemaName */
                SQLCHAR      FAR  *TableName,        /* szTableName */
                SQLSMALLINT       NameLength3,       /* cbTableName */
                SQLCHAR      FAR  *TableType,        /* szTableType */
                SQLSMALLINT       NameLength4);      /* cbTableType */
```

Argumentos de la función:

Tabla 92. Argumentos de SQLTables

Tipo de datos	Argumento	Uso	Descripción
SQLHSTMT	*StatementHandle*	Entrada	Descriptor de contexto de sentencia.
SQLCHAR	*CatalogName*	Entrada	Es posible que el almacenamiento intermedio contenga un *pattern-value* para calificar el conjunto de resultados. *Catalog* es la primera parte de un nombre de tabla de 3 partes. DB2 Everyplace ignora este campo.
SQLSMALLINT	*NameLength1*	Entrada	Longitud de *CatalogName*. DB2 Everyplace ignora este campo.
SQLCHAR	*SchemaName*	Entrada	Es posible que el almacenamiento intermedio contenga un *pattern-value* para calificar el conjunto de resultados por medio de un nombre de esquema. DB2 Everyplace ignora este campo.
SQLSMALLINT	*NameLength2*	Entrada	Longitud de *SchemaName*. DB2 Everyplace ignora este campo.

Tabla 92. Argumentos de SQLTables (continuación)

Tipo de datos	Argumento	Uso	Descripción
SQLCHAR	*TableName*	Entrada	Es posible que el almacenamiento intermedio contenga un *pattern-value* para calificar el conjunto de resultados por medio de un nombre de tabla.
SQLSMALLINT	*NameLength3*	Entrada	Longitud de *TableName*.
SQLCHAR	*TableType*	Entrada	DB2 Everyplace sólo da soporte al tipo TABLE. DB2 Everyplace ignora este campo.
SQLSMALLINT	*NameLength4*	Entrada	DB2 Everyplace ignora este campo.

Tenga en cuenta que los argumentos de *TableName* aceptan patrones de búsqueda.

Uso:

La información de tabla se devuelve en un conjunto de resultados en el que cada tabla se representa por medio de una fila del conjunto de resultados.

A veces, una aplicación llama a `SQLTables()` con punteros nulos al argumento *TableName* para que no se efectúe ningún intento de restringir el conjunto de resultados devuelto. Para algunas fuentes de datos que contienen una gran cantidad de tablas, este escenario se asocia a un conjunto de resultados extremadamente grande y a tiempos de recuperación muy grandes.

El conjunto de resultados que devuelve `SQLTables()` contiene las columnas listadas en la Tabla 93 en el orden indicado. Las filas se ordenan por medio de TABLE_NAME.

Las llamadas a `SQLTables()` deben utilizarse con moderación, ya que en muchos casos se asignan a una consulta compleja y por tanto cara en el catálogo del sistema. Los resultados deben guardarse en vez de repetir las llamadas.

Las columnas VARCHAR del conjunto resultante de las funciones de catálogo se han declarado con un atributo de longitud máxima 128 para que sea coherente los límites de SQL92. Dado que los nombres de DB2 son inferiores a 128, la aplicación puede optar por poner siempre a un lado 128 caracteres (más el terminador nulo) para el almacenamiento intermedio de salida o, alternativamente llamar a `SQLGetInfo()` con SQL_MAX_TABLE_NAME_LEN para determinar las longitudes reales de la columna TABLE_NAME a la que da soporte el DBMS conectado.

Tabla 93. Columnas devueltas por SQLTables

Nombre de la columna	Tipo de datos	Descripción
TABLE_CAT	VARCHAR (128)	Siempre es una serie de longitud cero.
TABLE_SCHEM	VARCHAR (128)	Siempre es una serie de longitud cero.
TABLE_NAME	VARCHAR (128)	El nombre de la tabla.
TABLE_TYPE	VARCHAR (128)	Identifica el tipo proporcionado por el nombre en la columna TABLE_NAME. Siempre tiene el valor de serie 'TABLE'.
REMARKS	VARCHAR(254)	Contiene la información descriptiva sobre la tabla.

Códigos de retorno:
- SQL_SUCCESS
- SQL_SUCCESS_WITH_INFO
- SQL_ERROR
- SQL_INVALID_HANDLE

Diagnósticos:

Tabla 94. SQLTables SQLSTATEs

SQLSTATE	Descripción	Explicación
HY001	Error de asignación de memoria.	DB2 CLI no puede asignar la memoria necesaria para ejecutar o finalizar la función.
HY014	No hay más descriptores de contexto.	DB2 CLI no pudo asignar un descriptor debido a los recursos internos.
HY090	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El valor de uno de los argumentos de longitud de nombre era menor que 0, pero no era igual a SQL_NTS. La validez de uno de los argumentos de longitud del nombre superaba el valor máximo al que se da soporte para dicha fuente de datos. El valor soportado máximo puede obtenerse llamando a la función `SQLGetInfo()`.

Restricciones:

Ninguna.

Consulta relacionada:
- "SQLGetInfo—Obtener información general" en la página 257

Conversión de datos por funciones de CLI de DB2

CLI de DB2 gestiona la transferencia de datos, y su conversión si procede, entre la aplicación y DB2 Everyplace. Antes de que tenga lugar la transferencia de datos propiamente dicha, se indica el tipo de datos fuente, el tipo de datos destino o ambos tipos cuando se llama a `SQLBindParameter()`, `SQLBindCol()` o `SQLGetData()`. Estas funciones utilizan nombres simbólicos (tales como SQL_CHAR y SQL_C_CHAR) para identificar los tipos de datos implicados.

Por ejemplo, para enlazar un marcador de parámetro que corresponde a un tipo de datos de SQL de SQL_VARCHAR, con un tipo de almacenamiento intermedio C de entero largo de una aplicación, la llamada apropiada a `SQLBindParameter()` tendría este aspecto:

```
SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_LONG,
                  SQL_VARCHAR, 0, 0, long_ptr, 0, NULL);
```

La Tabla 95 en la página 286 muestra las conversiones soportadas entre los tipos de datos C y de SQL. La primera columna de la Tabla 95 contiene el tipos de datos de SQL. Las columnas restantes representan los tipos de datos C. Si la columna del tipo de datos C contiene:

D La conversión está soportada y es la conversión por omisión para el tipo de datos de SQL.

X DB2 Everyplace da soporte a la conversión.

en blanco
DB2 Everyplace no da soporte a la conversión.

Los límites existentes respecto a la precisión, la escala y las reglas para el truncamiento y el redondeo para conversiones de tipos siguen las normas de sintaxis de SQL.

Tabla 95. Conversiones de datos soportadas

SQL, tipo de datos	Conversión por omisión	Otras conversiones soportadas
BLOB	SQL C BINARY	SQL C CHAR
CHAR	SQL C CHAR	SQL C LONG SQL C SHORT SQL C TINYINT SQL C TYPE DATE SQL C TYPE TIME SQL C BINARY SQL C BIT SQL C TYPE TIMESTAMP
DATE	SQL C TYPE DATE	SQL C CHAR
DECIMAL	SQL C CHAR	SQL C LONG SQL C SHORT SQL C TINYINT SQL C BIT
INTEGER	SQL C LONG	SQL C CHAR SQL C SHORT SQL C TINYINT SQL C FLOAT SQL C DOUBLE SQL C BIT
SMALLINT	SQL C SHORT	SQL C CHAR SQL C LONG SQL C TINYINT SQL C FLOAT SQL C DOUBLE SQL C BIT
TIME	SQL C TYPE TIME	SQL C CHAR
TIMESTAMP	SQL C TYPE TIMESTAMP	SQL C CHAR
VARCHAR	SQL C CHAR	SQL C LONG SQL C SHORT SQL C TINYINT SQL C TYPE DATE SQL C TYPE TIME SQL C BINARY SQL C BIT SQL C TYPE TIMESTAMP

Consulta relacionada:

- "Clave para las descripciones de funciones de CLI de DB2" en la página 198
- "Resumen de las funciones de CLI de DB2" en la página 194

Métodos de JDBC soportados

Este capítulo contiene información sobre los métodos JDBC a los que da soporte DB2 Everyplace. Este capítulo contiene los apartados siguientes:

- "Visión general del soporte de JDBC de DB2 Everyplace"
- "Interfaces en el paquete de java.sql"
- "Interfaces en el paquete javax.sql" en la página 305

Visión general del soporte de JDBC de DB2 Everyplace

DB2 Everyplace soporta un subconjunto de los métodos definidos en la especificación de la API Java Database Connectivity (JDBC) ofrecida en el Sun Java Developer's Kit. La información sobre los métodos JDBC a los que da soporte DB2 Everyplace se modifica a partir de la documentación de Java Development Kit Versión 1.4.1 de Sun. DB2 Everyplace también da soporte a las interfaces Connection y Statement ampliadas.

Consulte los apartados "Clase de DB2eStatement" en la página 304 y "Clase de DB2eConnection" en la página 291 para obtener más información.

El controlador JDBC de DB2 Everyplace es compatible con el paquete opcional de JDBC para el perfil de CDC/Foundation especificado por medio de JSR 169.

Tareas afines:

- "Desarrollo de aplicaciones Java de DB2 Everyplace" en la página 17

Consulta relacionada:

- "Interfaz Blob" en la página 288
- "Interfaz Connection" en la página 290
- "Clase de DB2eConnection" en la página 291
- "Interfaz DatabaseMetaData" en la página 292
- "Interfaz Driver" en la página 295
- "Interfaz PreparedStatement" en la página 296
- "Interfaz ResultSet" en la página 297
- "Interfaz ResultSetMetaData" en la página 301
- "Interfaz Statement" en la página 303
- "Clase de DB2eStatement" en la página 304
- "Mensajes de SQLState notificados por JDBC" en la página 193

Interfaces en el paquete de java.sql

Este capítulo proporciona información sobre los métodos JDBC del paquete java.sql. Los temas que se tratan son:

- "Interfaz Blob" en la página 288
- "Interfaz CallableStatement" en la página 288
- "Interfaz Connection" en la página 290
- "Clase de DB2eConnection" en la página 291
- "Interfaz DatabaseMetaData" en la página 292
- "Interfaz Driver" en la página 295
- "Interfaz PreparedStatement" en la página 296
- "Interfaz ResultSet" en la página 297

- "Interfaz ResultSetMetaData" en la página 301
- "Interfaz Statement" en la página 303
- "Clase de DB2eStatement" en la página 304

Interfaz Blob

La interfaz Blob representa (correlaciona) un BLOB de SQL en el lenguaje de programación Java™. Un BLOB de SQL es un tipo incorporado que almacena un objeto grande binario como valor de columna en una fila de una tabla de base de datos. Un objeto BLOB es válido mientras dura la transacción en que se ha creado.

Los métodos de las interfaces ResultSet y PreparedStatement, como por ejemplo getBlob y setBlob, permite que un programador acceda al BLOB de SQL. La interfaz Blob proporciona métodos para obtener la longitud del valor de un BLOB (objeto grande binario) de SQL y para materializar el valor de un BLOB en el cliente.

java.sql, paquete

public interface **Blob**

La Tabla 96 lista los métodos de la interfaz Blob a los que da soporte DB2 Everyplace.

Tabla 96. Métodos de la interfaz Blob

Tipo de valor de retorno del método	Método
`InputStream`	**getBinaryStream**() Recupera el BLOB designado por esta instancia BLOB en forma de corriente.
`byte[]`	**getBytes**(long `pos`, int `longitud`) Devuelve, en forma de matriz de bytes, parte del valor del BLOB que este objeto BLOB designa, o todo el mencionado valor.
`long`	**longitud**() Devuelve el número de bytes del valor de BLOB designado por este objeto BLOB.

Tareas afines:

- "Desarrollo de aplicaciones Java de DB2 Everyplace" en la página 17

Consulta relacionada:

- "Visión general del soporte de JDBC de DB2 Everyplace" en la página 287
- "Mensajes de SQLState notificados por JDBC" en la página 193

Interfaz CallableStatement

La interfaz utilizada para ejecutar procedimientos almacenados SQL remotos. El parámetro de resultados debe registrarse como parámetro OUT. Los demás parámetros pueden utilizarse para la entrada, la salida o para ambos. A los parámetros se les hace referencia de modo secuencial, por número. El primer parámetro es 1.

Consulte la sección denominada "La consulta remota y el adaptador del procedimiento almacenado" en la publicación *DB2 Everyplace Sync Server Administration Guide* para obtener más detalles.

```
call <procedure-name> (?,?, ...)
```

Los valores del parámetro IN se establecen utilizando los métodos de definición heredados de PreparedStatement. El tipo de todos los parámetros OUT deben registrarse antes de ejecutar el procedimiento almacenado; sus valores se recuperan después de la ejecución por medio de los métodos get que se proporcionan en este punto. El tamaño del parámetro de salida se limita a 4K bytes.

`CallableStatement` puede devolver un `ResultSet`.

java.sql, paquete

CallableStatement de interfaz pública

amplía PreparedStatement

La Tabla 97 lista los métodos de la interfaz CallableStatement a los que da soporte DB2 Everyplace.

Tabla 97. Métodos de la interfaz CallableStatement

Tipo de valor de retorno del método	Método
`Blob`	**getBlob**(int `i`) JDBC 2.0 Obtiene el valor de un parámetro BLOB de JDBC como objeto `Blob` en el lenguaje de programación Java.
`byte[]`	**getBytes**(int `parameterIndex`) Obtiene el valor de un parámetro JDBC BINARY o VARBINARY como matriz de valores de `byte` en el lenguaje de programación de Java.
`Date`	**getDate**(int `parameterIndex`) Obtiene el valor de un parámetro DATE de JDBC como objeto `java.sql.Date`.
`int`	**getInt**(int `parameterIndex`) Obtiene el valor de un parámetro INTEGER de JDBC como `int` en el lenguaje de programación de Java.
`Object`	**getObject**(int `parameterIndex`) Obtiene el valor de un parámetro como `object` en el lenguaje de programación de Java.
`short`	**getShort**(int `parameterIndex`) Obtiene el valor de un parámetro SMALLINT de JDBC como `short` en el lenguaje de programación de Java.
`String`	**getString**(int `parameterIndex`) Recupera el valor de un parámetro CHAR, VARCHAR o LONGVARCHAR de JDBC como `String` en el lenguaje de programación de Java.
`Time`	**getTime**(int `parameterIndex`) Obtiene el valor de un parámetro TIME de JDBC como objeto `java.sql.Time`.
`Timestamp`	**getTimestamp**(int `parameterIndex`) Obtiene el valor de un parámetro TIMESTAMP de JDBC como objeto `java.sql.Timestamp`.
`void`	**registerOutParameter**(int `parameterIndex`, int `sqlType`) Registra el parámetro OUT en posición ordinal `parameterIndex` para el tipo de JDBC `sqlType`.
`boolean`	**wasNull**() Indica si el último parámetro OUT leído tenía o no el valor NULL de SQL.

Tareas afines:

- "Desarrollo de aplicaciones Java de DB2 Everyplace" en la página 17

Consulta relacionada:

- "Visión general del soporte de JDBC de DB2 Everyplace" en la página 287
- "Mensajes de SQLState notificados por JDBC" en la página 193

Interfaz Connection

La interfaz Connection establece una conexión (sesión) con una base de datos específica. En el contexto de una conexión, se ejecutan sentencias de SQL y se devuelven resultados.

Una base de datos de Connection puede proporcionar información que describe sus tablas, su gramática de SQL soportada, sus procedimientos almacenados, las posibilidades de esta conexión, etc. Esta información se obtiene mediante el método getMetaData.

java.sql, paquete

public interface **Connection**

La Tabla 98 lista los métodos de la interfaz Connection a los que da soporte DB2 Everyplace.

Tabla 98. Métodos de la interfaz Connection

Tipo de valor de retorno del método	Método
`void`	**clearWarnings**() Borra todos los avisos de los que se ha informado para este objeto Connection.
`void`	**close**() Libera una base de datos de Connection y los recursos de JDBC de modo inmediato en vez de esperar que los mismos se liberen de modo automático.
`void`	**commit** () Hace que sean permanentes los cambios hechos desde la última operación de confirmación o cancelación y libera los bloqueos de base de datos mantenidos actualmente por Connection.
`Statement`	**createStatement**() Crea un objeto `Statement` para enviar sentencias de SQL a la base de datos.
`Statement`	**createStatement**(int `resultSetType`, int `resultSetConcurrency`) JDBC 2.0. Crea un objeto Statement que generará objetos ResultSet con el tipo y la simultaneidad indicados.
`boolean`	**isClosed**() Comprueba si se ha cerrado una Conexión.
`DatabaseMetaData`	**getMetaData**() Obtiene los metadatos relativos a esta base de datos de Connection.
`SQLWarning`	**getWarnings**() Devuelve el primer aviso informado por llamadas a esta Connection.
`CallableStatement`	**prepareCall**(String `sql`) Crea un objeto `CallableStatement` para llamar procedimientos almacenados de base de datos.
`PreparedStatement`	**prepareStatement** (String `sql`) Crea un objeto `PreparedStatement` para enviar sentencias de SQL parametrizadas a la base de datos.
`PreparedStatement`	**prepareStatement**(String `sql`, int `resultSetType`, int `resultSetConcurrency`) JDBC 2.0. Crea un objeto PreparedStatement que generará objetos ResultSet con el tipo y la simultaneidad indicados.
`void`	**rollback** () Elimina los cambios hechos desde la última operación de confirmación o cancelación y libera los bloqueos de base de datos mantenidos actualmente por esta Connection.

Tabla 98. Métodos de la interfaz Connection (continuación)

Tipo de valor de retorno del método	Método
`void`	**setAutoCommit** (boolean `autoCommit`) Establece la modalidad de confirmación automática de esta conexión.

Tareas afines:

- "Desarrollo de aplicaciones Java de DB2 Everyplace" en la página 17

Consulta relacionada:

- "Visión general del soporte de JDBC de DB2 Everyplace" en la página 287
- "Mensajes de SQLState notificados por JDBC" en la página 193

Clase de DB2eConnection

La clase de DB2eConnection obtiene y establece determinados atributos de Connection. Para utilizar los métodos de clase de DB2eConnection sobre un objeto Connection, antes se tiene que convertir el objeto Connection en un objeto DB2eConnection. Estos métodos se implementan mediante llamadas a las funciones CLI/ODBC de SQLGetConnectAttr y SQLSetConnectAttr con los argumentos apropiados.

Consulte el capítulo 10, Funciones CLI/ODBC de DB2 soportadas para obtener más información.

com.ibm.db2e.jdbc package

DB2eConnection de clase pública

implanta Connection

La Tabla 99 lista los métodos de la clase DB2eConnection a los que da soporte DB2 Everyplace.

Tabla 99. Métodos de la clase DB2eConnection

Tipo de retorno del método	Método
`void`	**enableFilenameFormat83** (`boolean enable`) Habilita el motor de base de datos para crear nombres de archivo en el formato 8.3 si se cumple (true) la habilitación; en caso contrario habilita los nombres de archivo en formato long. El formato de nombre de archivo sólo puede cambiarse si no hay archivos de catálogo en la vía de acceso para esta conexión.
`boolean`	**isEnabledFilenameFormat83** () ¿Crea el motor de la base de datos nombres de archivo en el formato 8.3? O crea nombres de archivo en el formato long?

Tareas afines:

- "Desarrollo de aplicaciones Java de DB2 Everyplace" en la página 17

Consulta relacionada:

- "Visión general del soporte de JDBC de DB2 Everyplace" en la página 287
- "Mensajes de SQLState notificados por JDBC" en la página 193

- "SQLGetConnectAttr—Obtener el valor actual de un atributo de conexión" en la página 246
- "SQLSetConnectAttr—Establecer opciones referentes a una conexión" en la página 272

Interfaz DatabaseMetaData

La interfaz DatabaseMetaData proporciona información amplia sobre la base de datos como un todo.

Algunos de estos métodos toman argumentos String para los nombres de catálogo y de esquema. DB2 Everyplace ignora estos argumentos.

Algunos de los métodos aquí contenidos devuelven listas de información en forma de objetos ResultSet. Puede utilizar los métodos ResultSet normales, como por ejemplo getString y getInt, para recuperar los datos de estos ResultSets.

Si no se dispone de un formulario de metadatos indicado, estos métodos emiten una SQLException.

java.sql, paquete

public interface **DatabaseMetaData**

La Tabla 100 lista los campos de la interfaz DatabaseMetaData a los que da soporte DB2 Everyplace.

Tabla 100. Campos de DatabaseMetaData

Tipo de campo	Campo
`static int`	**columnNoNulls** Indica que es posible que la columna no admita valores nulos (NULL).
`static int`	**columnNullable** Indica que la columna permite definitivamente valores nulos (NULL).
`static int`	**columnNullableUnknown** Indica que la anulación de columnas es desconocida.

La Tabla 101 lista los métodos de la interfaz DatabaseMetaData a los que da soporte DB2 Everyplace.

Tabla 101. Métodos de la interfaz DatabaseMetaData

Tipo de valor de retorno del método	Método
`ResultSet`	**getColumns**(String `catalog`, String `schemaPattern`, String `tableNamePattern`, String `columnNamePattern`) Obtiene una descripción de las columnas de tabla disponibles en el catálogo especificado.
`Connection`	**getConnection**() JDBC 2.0 Recupera la conexión que ha producido este objeto de metadatos.

Tabla 101. Métodos de la interfaz DatabaseMetaData (continuación)

Tipo de valor de retorno del método	Método
`ResultSet`	**getCrossReference**(String `primaryCatalog`, String `primarySchema`, String `primaryTable`, String `foreignCatalog`, String `foreignSchema`, String `foreignTable`) Obtiene una descripción de las columnas de clave foránea de la tabla de claves foráneas que hace referencia a las columnas de clave primaria de la tabla de claves primarias (describe cómo una tabla importa la clave de otra.) Normalmente, debe devolver un solo par de clave foránea/clave primaria (la mayoría de tablas únicamente importan una clave foránea de una tabla una vez.) Están ordenadas por FKTABLE_NAME y KEY_SEQ.
`String`	**getDatabaseProductName**() ¿Cuál es el nombre de este producto de base de datos?
`String`	**getDatabaseProductVersion**() ¿Cuál es la versión de este producto de base de datos?
`int`	**getDriverMajorVersion**() ¿Cuál es el número de versión mayor de este controlador JDBC?
`int`	**getDriverMinorVersion**() ¿Cuál es el número de versión menor de este controlador JDBC?
`String`	**getDriverName**() ¿Cuál es el nombre de este controlador JDBC?
`String`	**getDriverName**() ¿Cuál es la versión de este controlador JDBC?
`ResultSet`	**getExportedKeys**(String `catalog`, String `schema`, String `table`) Obtiene una descripción de las columnas de clave foránea que hacen referencia a las columnas de clave primaria de una tabla (las claves foráneas exportadas por una tabla).
`String`	**getIdentifierQuoteString**() ¿Cuál es la serie utilizada para los identificadores de SQL? Devuelve un espacio " " si no se soporta el entrecomillado de identificadores.
`ResultSet`	**getImportedKeys**(String `catalog`, String `schema`, String `table`) Obtiene una descripción de las columnas de clave primaria a las que hacen referencia las columnas de clave foránea de una tabla (las claves primarias importadas por una tabla).
`int`	**getMaxBinaryLiteralLength**() ¿Cuántos caracteres hexadecimales se pueden tener en un literal binario en línea?
`int`	**getMaxCharLiteralLength**()¿Cuál es la longitud máxima para un literal de tipo carácter?
`int`	**getMaxColumnNameLength**() ¿Cuál es el límite en la longitud de un nombre de columna?
`int`	**getMaxColumnsInGroupBy**() ¿Cuál es el número máximo de columnas en una cláusula GROUP BY?
`int`	**getMaxColumnsInIndex**() ¿Cuál es el número máximo de columnas permitidas en un índice?
`int`	**getMaxColumnsInOrderBy**() ¿Cuál es el número máximo de columnas en una cláusula ORDER BY?
`int`	**getMaxColumnsInSelect**() ¿Cuál es el número máximo de columnas en una sentencia SELECT?
`int`	**getMaxConnections**() ¿Cuántas conexiones activas con esta base de datos puede haber a la vez?
`int`	**getMaxIndexLength**() ¿Cuál es la longitud máxima de un índice (en bytes)?

Tabla 101. Métodos de la interfaz DatabaseMetaData (continuación)

Tipo de valor de retorno del método	Método
`int`	**getMaxRowSize**() ¿Cuál es la longitud máxima de una fila simple?
`int`	**getMaxStatementLength**() ¿Cuál es la longitud máxima de una sentencia de SQL?
`int`	**getMaxStatements**() ¿Cuántas sentencias activas para esta base de datos pueden estar abiertas a la vez?
`int`	**getMaxTableNameLength**() ¿Cuál es la longitud máxima de un nombre de tabla?
`int`	**getMaxTablesInSelect**() ¿Cuál es el número máximo de tablas en una sentencia SELECT?
`int`	**getMaxUserNameLength**() ¿Cuál es la longitud máxima de un nombre de usuario?
`ResultSet`	**getPrimaryKeys**(String `catalog`, String `schema`, String `table`) Obtiene una descripción de las columnas de clave primaria de una tabla.
`String`	**getSearchStringEscape**() Obtiene la serie que se puede utilizar para los caracteres comodín de escape.
`ResultSet`	**getTables**(String `catalog`, String `schemaPattern`, String `tableNamePattern`, String[] `types`) Obtiene una descripción de las tablas disponibles en un catálogo.
`ResultSet`	**getUDTs**(String `catalog`, String `schemaPattern`, String `typeNamePattern`, int[] `types`) JDBC 2.0 Obtiene una descripción de los tipos definidos por el usuario en un esquema concreto. DB2 Everyplace siempre devuelve un conjunto de resultados vacío ya que no da soporte a los UDT.
`String`	**getURL**() ¿Cuál es el URL para esta base de datos?
`String`	**getUserName**() ¿Cuál es el nombre de usuario tal como lo conoce la base de datos?
`boolean`	**supportsColumnAliasing**() ¿Se da soporte a los alias de columna?
`boolean`	**supportsFullOuterJoins**() ¿Se da soporte a las uniones externas anidadas completas?
`boolean`	**supportsMixedCaseIdentifiers**() ¿Trata la base de datos a los identificadores de SQL sin comillas con mayúsculas y minúsculas como sensibles a las mayúsculas/minúsculas y como consecuencia los almacena mezclando mayúsculas y minúsculas?
`boolean`	**supportsMixedCaseQuotedIdentifiers**() ¿Trata la base de datos a los identificadores de SQL con comillas con mayúsculas y minúsculas como sensibles a las mayúsculas/minúsculas y como consecuencia los almacena mezclando mayúsculas y minúsculas?
`boolean`	**supportsNonNullableColumns**() ¿Pueden definirse las columnas como no anulables?
`boolean`	**supportsOrderByUnrelated**() ¿Puede la cláusula ″ORDER BY″ utilizar columnas que no estén en la sentencia SELECT?
`boolean`	**supportsOuterJoins**() ¿Se da soporte a alguna forma de unión externa?
`boolean`	**supportsPositionedDelete**() ¿Se da soporte a a una DELETE posicionada?

Tabla 101. Métodos de la interfaz DatabaseMetaData (continuación)

Tipo de valor de retorno del método	Método
`boolean`	**supportsPositionedUpdate**() ¿Se da soporte a a una UPDATE posicionada?
`boolean`	**supportsResultSetType**(`int type`) JDBC 2.0 ¿Da soporte la base de datos al tipo de conjunto de resultados concreto?
`boolean`	**supportsSchemasInTableDefinitions**() ¿Puede utilizarse un nombre de esquema en una sentencia de definición de tablas?
`boolean`	**supportsTransactions**() ¿Se soportan transacciones? De no ser así, el nivel de aislamiento es TRANSACTION_NONE.

Tareas afines:

- "Desarrollo de aplicaciones Java de DB2 Everyplace" en la página 17

Consulta relacionada:

- "Visión general del soporte de JDBC de DB2 Everyplace" en la página 287
- "Mensajes de SQLState notificados por JDBC" en la página 193

Interfaz Driver

La interfaz Driver es la infraestructura de Java SQL que admite varios controladores de base de datos.

Cuando se cargue una clase Driver, debería crearse una instancia de sí misma y registrarse con el DriverManager. Esto significa que un usuario puede cargar y registrar el controlador JDBC de DB2 Everyplace llamando a:

```
Class.forName("com.ibm.db2e.jdbc.DB2eDriver")
```

java.sql, paquete

public interface **Driver**

La Tabla 102 lista los métodos de la interfaz Driver a los que da soporte DB2 Everyplace.

Tabla 102. Métodos de la interfaz Driver

Tipo de valor de retorno del método	Método
`boolean`	**acceptsURL**(String `url`)Devuelve true si el controlador considera que puede abrir una conexión con el URL indicado.
`Connection`	**connect**(String `url`, Properties info) Intenta crear una conexión de base de datos con el URL indicado. Se puede utilizar el argumento java.util.Properties para pasar pares arbitrarios de código/valor de serie como argumentos de conexión. DB2 Everyplace soporta los pares siguientes de clave y valor específicos del controlador: • Clave: LOGIN_TIMEOUT Valor: número de segundos • Clave: DB2e_ENCODING Valor: codificación de caracteres
`int`	**getMajorVersion**() Obtiene el número de versión mayor del controlador.

Tabla 102. Métodos de la interfaz Driver (continuación)

Tipo de valor de retorno del método	Método
`int`	**getMinorVersion()** Obtiene el número de versión menor del controlador.
`boolean`	**jdbcCompliant()** Informa sobre si éste es un controlador JDBC COMPLIANT™ genuino.

Tareas afines:

- "Desarrollo de aplicaciones Java de DB2 Everyplace" en la página 17

Consulta relacionada:

- "Visión general del soporte de JDBC de DB2 Everyplace" en la página 287
- "Mensajes de SQLState notificados por JDBC" en la página 193

Interfaz PreparedStatement

La interfaz PreparedStatement crea un objeto que representa una sentencia de SQL compilada previamente.

Se ha compilado previamente una sentencia de SQL y se ha almacenado en un objeto PreparedStatement. Este objeto puede utilizarse para ejecutar de modo eficaz esta sentencia varias veces.

java.sql, paquete

public interface **PreparedStatement**

extends Statement

La Tabla 103 lista los métodos de la interfaz PreparedStatement a los que da soporte DB2 Everyplace.

Tabla 103. Métodos de la interfaz PreparedStatement

Tipo de valor de retorno del método	Método
`void`	**clearParameters()** Borra los valores de parámetro actuales de modo inmediato.
`boolean`	**execute()** Ejecuta cualquier tipo de sentencia de SQL.
`ResultSet`	**executeQuery()** Ejecuta la consulta de SQL en este objeto `PreparedStatement` y devuelve el conjunto de resultados generado por la consulta.
`int`	**executeUpdate()** Ejecuta la sentencia de SQL INSERT, UPDATE o DELETE en este objeto `PreparedStatement`.
`void`	**setBigDecimal** (int `índiceParámetros`, `BigDecimal x`) Establece el parámetro designado con un valor java.lang.BigDecimal. Este método no está disponible en el controlador JDBC de DB2 Everyplace para Palm OS.
`void`	**setBoolean** (int `índiceParámetros`, boolean `x`) Establece el parámetro designado en un valor `boolean` de Java. El controlador JDBC de DB2 Everyplace lo convierte a un valor SMALLINT de SQL cuando lo envía a la base de datos.
`void`	**setBlob**(int `i`, Blob `x`) JDBC 2.0 Establece un parámetro BLOB.
`void`	**setBytes**(int `índiceParámetros`, byte[ ]`x`) Establece el parámetro designado en una matriz de bytes de Java.
`void`	**setDate**(int `índiceParámetros`, Date `x`) Establece el parámetro designado en un valor de java.sql.Date.

Tabla 103. Métodos de la interfaz PreparedStatement (continuación)

Tipo de valor de retorno del método	Método
`void`	**setDouble**(int `parameterIndex`, double `x`) Establece el parámetro designado en un valor double de Java. El controlador JDBC de DB2 Everyplace lo convierte a un valor DECIMAL de SQL cuando lo envía a la base de datos.
`void`	**setFloat**(int `parameterIndex`, float `x`) Establece el parámetro designado en un valor float de Java. Cuando BigDecimal se convierte a float, si el BigDecimal es demasiado grande para representarlo como float, se convertirá a FLOAT.NEGATIVE_INFINITY o FLOAT.POSITIVE_INFINITY según corresponda.
`void`	**setInt** (int `índiceParámetros`, int `x`) Establece el parámetro designado en un valor `int` de Java.
`void`	**setLong**(int `parameterIndex`, long `x`) Establece el parámetro designado en un valor long de Java.
`void`	**setNull** (int `índiceParámetros`, int `sqlType`) Establece el parámetro designado en NULL de SQL.
`void`	**setObject**(int `parameterIndex`, Object `x`, int `targetSqlType`) Establece el valor del parámetro designado con el objeto concreto. **Restricciones de DB2 Everyplace**: • `targetSqlType` debe corresponderse con uno de los tipos de datos a los que da soporte DB2 Everyplace. • Se da soporte a las conversiones básicas y de Serie. Por ejemplo, si `targetSqlType` es `Types.INTEGER`, x debería ser un objeto `Integer` o `String`. • Si `targetSqlType` es `Types.DECIMAL`, x también puede ser un objeto `Double`, `Float` o `Long`. • Si `targetSqlType` es `Types.SMALLINT`, x también puede ser un objeto `Boolean`. • En Palm OS, si `targetSqlType` es `Types.DECIMAL`, x debería ser un objeto `String`.
`void`	**setShort** (int `índiceParámetros`, short `x`) Establece el parámetro designado en un valor `short` de Java.
`void`	**setString** (int `índiceParámetros`, String `x`) Establece el parámetro designado en un valor `String` de Java.
`void`	**setTime** (int `índiceParámetros`, Time `x`) Establece el parámetro designado en un valor de java.sql.Time.
`void`	**setTimestamp** (int `índiceParámetros`, Timestamp `x`) Establece el parámetro designado en un valor de java.sql.Timestamp.

Tareas afines:

- "Desarrollo de aplicaciones Java de DB2 Everyplace" en la página 17

Consulta relacionada:

- "Visión general del soporte de JDBC de DB2 Everyplace" en la página 287
- "Mensajes de SQLState notificados por JDBC" en la página 193

Interfaz ResultSet

La interfaz ResultSet proporciona acceso a una tabla de datos. Un objeto ResultSet normalmente se genera ejecutando una Sentencia.

Un objeto ResultSet mantiene un cursor apuntando hacia su fila de datos actual. Inicialmente, el cursor está colocado delante de la primera fila. El método `next()` mueve el cursor a la fila siguiente.

Los métodos getXXX recuperan valores de columna para la fila actual. Puede recuperar valores utilizando el número de índice de la columna o el nombre de la columna. En general, la utilización del índice de columnas es más eficaz. Las columnas se numeran a partir de uno.

java.sql, paquete

public interface **ResultSet**

La Tabla 104 lista los campos de la interfaz ResultSet a los que da soporte DB2 Everyplace.

Tabla 104. Campos de la interfaz ResultSet

Tipo de campo	Campo
`static int`	**CONCUR_READ_ONLY** La constante que indica la modalidad de simultaneidad para un objeto ResultSet que es posible que NO pueda actualizarse. **Nota**: DB2 Everyplace no da soporte a CONCUR_UPDATABLE. Si se especifica CONCUR_UPDATABLE para la modalidad de simultaneidad para un objeto ResultSet al crear un objeto Statement, el controlador de JDBC de DB2 Everyplace emite un SQLWarning en el objeto Connection que ha producido el objeto Statement y en su lugar utiliza CONCUR_READ_ONLY.
`static int`	**TYPE_FORWARD_ONLY** La constante que indica el tipo para un objeto ResultSet cuyo cursor sólo pueda moverse hacia adelante.
`static int`	**TYPE_SCROLL_INSENSITIVE** La constante que indica el tipo para un objeto ResultSet que pueda desplazarse pero que en general no resulte sensible a los cambios efectuados por otros. **Nota**: Utilice este tipo de objeto ResultSet con moderación, puesto que puede afectar al rendimiento. Este tipo utiliza SQL_INSENSITIVE como valor del atributo de sentencia CLI SQL_ATTR_CURSOR_SENSITIVITY. Consulte la documentación para la función de CLI SQLSetStmtAttr para obtener más detalles.
`static int`	**TYPE_SCROLL_SENSITIVE** La constante que indica el tipo para un objeto ResultSet que pueda desplazarse y que en general resulte sensible a los cambios efectuados por otros. **Nota**: Este tipo utiliza SQL_UNSPECIFIED para el valor del atributo de sentencia de CLI SQL_ATTR_CURSOR_SENSITIVITY. Consulte la documentación para la función de CLI SQLSetStmtAttr para obtener más detalles.

La Tabla 105 lista los métodos de la interfaz ResultSet a los que da soporte DB2 Everyplace.

Tabla 105. Métodos de la interfaz ResultSet

Tipo de valor de retorno del método	Método
`boolean`	**absolute**(int `fila`) JDBC 2.0. Mueve el cursor al número de fila indicado del conjunto resultante.
`void`	**afterLast**() JDBC 2.0. Mueve el cursor al final del conjunto resultante, justo detrás de la última fila.
`void`	**beforeFirst**() JDBC 2.0. Mueve el cursor al principio del conjunto resultante, justo delante de la primera fila.
`void`	**clearWarnings**() Borra todos los avisos de los que se ha informado para este objeto ResultSet.

Tabla 105. Métodos de la interfaz ResultSet (continuación)

Tipo de valor de retorno del método	Método
`void`	**close**() Libera de inmediato los recursos de JDBC y base de datos de este objeto ResultSet en lugar de esperar a que suceda esto cuando se cierre automáticamente.
`int`	**findColumn**(String `columnName`) Correlaciona el nombre de columna de ResultSet indicado con su índice de columnas de ResultSet.
`boolean`	**first**() JDBC 2.0. Mueve el cursor a la primera fila del conjunto resultante.
`BigDecimal`	**getBigDecimal**(int `índiceColumnas`) JDBC 2.0. Obtiene el valor de una columna de la fila actual como objeto java.math.BigDecimal con precisión completa. El controlador JDBC de DB2 Everyplace para Palm OS no soporta este método.
`BigDecimal`	**getBigDecimal**(int `índiceColumnas`, int `escala`) Obtiene el valor de la columna designada de la fila actual de este objeto ResultSet como objeto java.math.BigDecimal en el lenguaje de programación Java. El controlador JDBC de DB2 Everyplace para Palm OS no soporta este método.**Desaprobado.**
`BigDecimal`	**getBigDecimal**(String `nombreColumna`) JDBC 2.0. Obtiene el valor de una columna de la fila actual como objeto java.math.BigDecimal con precisión completa. El controlador JDBC de DB2 Everyplace para Palm OS no soporta este método.
`BigDecimal`	**getBigDecimal**(String `columnName`, int `escala`) obtiene el valor de la columna designada de la fila actual de este objeto ResultSet como objeto java.math.BigDecimal en el lenguaje de programación Java. El controlador JDBC de DB2 Everyplace para Palm OS no soporta este método.**Desaprobado.**
`Blob`	**getBlob**(int `índiceColumnas`) JDBC 2.0. Obtiene un valor de BLOB de la fila actual de este objeto ResultSet.
`Blob`	**getBlob**(String `nombreColumna`) JDBC 2.0. Obtiene un valor de BLOB de la fila actual de este objeto ResultSet.
`boolean`	**getBoolean**(int `columnIndex`) Obtiene el valor de una columna de la fila actual como boolean de Java.
`boolean`	**getBoolean**(String `columnName`) Obtiene el valor de una columna de la fila actual como boolean de Java.
`byte`	**getByte**(int `índiceColumnas`) Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como byte en el lenguaje de programación Java.
`byte`	**getByte**(String `nombreColumna`) Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como byte en el lenguaje de programación Java.
`byte[]`	**getByte**(int `índiceColumnas`) Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como matriz de bytes en el lenguaje de programación Java.
`byte[]`	**getBytes**(String `nombreColumna`) Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como matriz de bytes en el lenguaje de programación Java.
`int`	**getConcurrency**() JDBC 2.0. Devuelve la modalidad de simultaneidad del conjunto resultante.

Tabla 105. Métodos de la interfaz ResultSet (continuación)

Tipo de valor de retorno del método	Método
`Date`	**getDate**(int `índiceColumnas`) Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como objeto java.sql.Date en el lenguaje de programación Java.
`Date`	**getDate**(int `índiceColumnas`, Calendar `cal`) Devuelve el valor de la columna designada en la fila actual de este objeto ResultSet como objeto java.sql.Date en el lenguaje de programación Java.
`Date`	**getDate**(String `nombreColumna`) Devuelve el valor de la columna designada en la fila actual de este objeto ResultSet como objeto java.sql.Date en el lenguaje de programación Java.
`double`	**getDouble**(int `columnIndex`) Obtiene el valor de una columna de la fila actual como double de Java.
`double`	**getDouble**(String `columnName`) Obtiene el valor de una columna de la fila actual como double de Java.
`float`	**getFloat**(int `columnIndex`) Obtiene el valor de una columna de la fila actual como float de Java.
`float`	**getFloat**(String `columnName`) Obtiene el valor de una columna de la fila actual como float de Java.
`int`	**getInt**(int `índiceColumnas`) Devuelve el valor de la columna designada en la fila actual de este objeto ResultSet como entero en el lenguaje de programación Java.
`int`	**getInt**(String `nombreColumna`) Devuelve el valor de la columna designada en la fila actual de este objeto ResultSet como entero en el lenguaje de programación Java.
`long`	**getLong**(int `columnIndex`) Obtiene el valor de una columna de la fila actual como long de Java.
`long`	**getLong**(String `columnName`) Obtiene el valor de una columna de la fila actual como long de Java.
`ResultSetMetaData`	**getMetaData**() Recupera el número, los tipos y las propiedades de las columnas de este objeto ResultSet.
`Object`	**getObject**(int `índiceColumnas`) Obtiene el valor de una columna de la fila actual como objeto Java.
`Object`	**getObject**(String `nombreColumna`) Obtiene el valor de una columna de la fila actual como objeto Java.
`int`	**getRow**() JDBC 2.0. Recupera el número de la fila actual.
`short`	**getShort**(int `índiceColumnas`) Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como short en el lenguaje de programación Java.
`short`	**getShort**(String `nombreColumna`) Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como short en el lenguaje de programación Java.
`Statement`	**getStatement**() JDBC 2.0. Devuelve la sentencia (Statement) que ha producido este objeto ResultSet.
`String`	**getString**(int `índiceColumnas`) Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como String en el lenguaje de programación Java.
`String`	**getString**(String `nombreColumna`) Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como String en el lenguaje de programación Java.

Tabla 105. Métodos de la interfaz ResultSet (continuación)

Tipo de valor de retorno del método	Método
`Time`	**getTime**(int `índiceColumnas`) Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como objeto java.sql.Time en el lenguaje de programación Java.
`Time`	**getTime**(String `nombreColumna`) Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como objeto java.sql.Time en el lenguaje de programación Java.
`Timestamp`	**getTimestamp**(String `nombreColumna`) Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como objeto java.sql.Timestamp en el lenguaje de programación Java.
`Timestamp`	**getTimestamp**(int `índiceColumnas`) Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como objeto java.sql.Timestamp en el lenguaje de programación Java.
`int`	**getType**() JDBC 2.0. Devuelve el tipo de este conjunto resultante.
`SQLWarning`	**getWarnings**() Devuelve el primer aviso informado por llamadas a este ResultSet.
`boolean`	**isAfterLast**() JDBC 2.0. Indica si el cursor se encuentra detrás de la última fila del conjunto resultante.
`boolean`	**isBeforeFirst**() JDBC 2.0. Indica si el cursor se encuentra delante de la primera fila del conjunto resultante.
`boolean`	**isFirst**() JDBC 2.0. Indica si el cursor se encuentra en la primera fila del conjunto resultante.
`boolean`	**isLast**() JDBC 2.0. Indica si el cursor se encuentra en la última fila del conjunto resultante. No se soporta este método para conjuntos resultantes de tipo TYPE_FORWARD_ONLY.
`boolean`	**last**() JDBC 2.0. Mueve el cursor a la última fila del conjunto resultante.
`boolean`	**next**() Mueve el cursor una fila hacia abajo a partir de su posición actual.
`boolean`	**previous**() JDBC 2.0. Mueve el cursor a la fila anterior del conjunto resultante.
`boolean`	**relative**(int `filas`) JDBC 2.0. Mueve el cursor un número relativo de filas, ya sea positivo o negativo.
`boolean`	**wasNull**() Informa sobre si la última columna leída tenía un valor NULL de SQL.

Tareas afines:

- "Desarrollo de aplicaciones Java de DB2 Everyplace" en la página 17

Consulta relacionada:

- "Visión general del soporte de JDBC de DB2 Everyplace" en la página 287
- "Mensajes de SQLState notificados por JDBC" en la página 193
- "SQLSetStmtAttr—Establecer opciones referentes a una sentencia" en la página 276

Interfaz ResultSetMetaData

La interfaz ResultSetMetaData crea un objeto que se puede utilizar para averiguar los tipos y propiedades de las columnas de un ResultSet.

java.sql, paquete

public interface **ResultSetMetaData**

La Tabla 106 lista los campos de la interfaz ResultSetMetaData a los que da soporte DB2 Everyplace.

Tabla 106. Campos de la interfaz ResultSetMetaData

Tipo de campo	Campo
`static int`	**columnNoNulls** Constante que indica que una columna no admite valores nulos (NULL).
`static int`	**columnNullable** Constante que indica que una columna admite valores nulos (NULL).
`static int`	**columnNullableUnknown** La constante que indica que se desconoce la posibilidad de anular los valores de una columna.

La Tabla 107 lista los métodos de la interfaz ResultSetMetaData a los que da soporte DB2 Everyplace.

Tabla 107. Métodos de la interfaz ResultSetMetaData

Tipo de valor de retorno del método	Método
`String`	**getCatalogName**(int `column`) Obtiene un nombre de catálogo de tabla de la columna. DB2 Everyplace siempre devuelve "" (no aplicable).
`int`	**getColumnCount** () Devuelve el número de columnas de este objeto `ResultSet`.
`int`	**getColumnDisplaySize** (int `columna`)Indica la anchura máxima normal de la columna designada, en caracteres.
`String`	**getColumnLabel**(int `column`) Obtiene el título de columna sugerido para su utilización en copias impresas y pantallas.
`String`	**getColumnName** (int `columna`) Obtiene el nombre de la columna designada.
`int`	**getColumnType** (int `columna`) Obtiene el tipo SQL de la columna designada.
`String`	**getColumnTypeName**(int `column`) Recupera un nombre de tipo específico de base de datos de columna.
`int`	**getPrecision** (int `columna`) Obtiene el número de dígitos decimales de la columna designada.
`int`	**getScale** (int `columna`) Obtiene el número de dígitos de la columna designada que se encuentran a la derecha de la cola decimal.
`String`	**getSchemaName**(int `column`) Obtiene un nombre de esquema de la tabla de la columna. DB2 Everyplace siempre devuelve "" (no aplicable).
`int`	**isNullable** (int `columna`) Indica la posibilidad de anulación de valores de la columna designada.
`boolean`	**isWritable**(int `column`) Indica si es posible que resulte satisfactoria una grabación de la columna.

Tareas afines:

• "Desarrollo de aplicaciones Java de DB2 Everyplace" en la página 17

Consulta relacionada:

- "Visión general del soporte de JDBC de DB2 Everyplace" en la página 287
- "Mensajes de SQLState notificados por JDBC" en la página 193

Interfaz Statement

La interfaz Statement crea un objeto que se utiliza para ejecutar una sentencia de SQL estático y obtener los resultados producidos por la misma.

java.sql, paquete

public interface **Statement**

La Tabla 108 lista los métodos de la interfaz Statement a los que da soporte DB2 Everyplace.

Tabla 108. Métodos de la interfaz Statement

Tipo de valor de retorno del método	Método
`void`	**addBatch**(String `sql`) JDBC 2.0 Añade un mandato SQL al proceso por lotes actual de mandatos para la sentencia.
`void`	**clearBatch**() JDBC 2.0 Convierte el conjunto de mandatos en el proceso por lotes actual vacío.
`void`	**close**() Libera los recursos JDBC y de base de datos de este objeto `Statement` de inmediato, en lugar de esperar a que suceda esto cuando se cierre automáticamente.
`boolean`	**execute**(String `sql`) Ejecuta una sentencia de SQL que puede devolver varios resultados.
`int[]`	**executeBatch**() JDBC 2.0 Somete un proceso por lotes de mandatos a la base de datos para su ejecución.
`ResultSet`	**executeQuery**(String `sql`) Ejecuta una sentencia de SQL que devuelve un solo objeto `ResultSet`.
`int`	**executeUpdate**(String `sql`) Ejecuta una sentencia `INSERT`, `UPDATE` o `DELETE` de SQL.
`Connection`	**getConnection** () JDBC 2.0. Devuelve el objeto Connection que ha producido este objeto Statement.
`boolean`	**getMoreResults**() Se mueve al siguiente resultado de una Statement. DB2 Everyplace siempre devuelve falso (no hay más resultados).
`ResultSet`	**getResultSet**() Devuelve el resultado actual como objeto `ResultSet`.
`int`	**getResultSetConcurrency**() JDBC 2.0. Recupera la simultaneidad de conjuntos resultantes.
`int`	**getResultSetType**() JDBC 2.0. Determina el tipo del conjunto resultante.
`int`	**getUpdateCount**() Devuelve el resultado actual como número de actualización; si el resultado es un ResultSet o no hay más resultados, se devuelve -1.

La Tabla 109 en la página 304 lista los campos de la interfaz Statement a los que da soporte DB2 Everyplace.

Tabla 109. Campos de la interfaz Statement

Tipo de campo	Campo
`static int`	**SUCCESS_NO_INFO** La constante que indica que una sentencia de proceso por lotes se ha ejecutado satisfactoriamente, pero que no se dispone de un recuento del número de filas que ha afectado.

Tareas afines:

- "Desarrollo de aplicaciones Java de DB2 Everyplace" en la página 17

Consulta relacionada:

- "Visión general del soporte de JDBC de DB2 Everyplace" en la página 287
- "Mensajes de SQLState notificados por JDBC" en la página 193

Clase de DB2eStatement

La clase de DB2eStatement obtiene y establece determinados atributos de Statement. Para utilizar los métodos de clase de DB2eStatement sobre un objeto Statement, antes se tiene que convertir el objeto Statement en un objeto DB2eStatement. Estos métodos se implementan mediante llamadas a las funciones CLI/ODBC de SQLGetStmtAttr y SQLSetStmtAttr con los argumentos apropiados.

Consulte el apartado "Resumen de las funciones de CLI de DB2" en la página 194 para obtener más información.

com.ibm.db2e.jdbc package

DB2eStatement de clase pública

implanta Statement

La Tabla 110 lista los métodos de la clase DB2eStatement a los que da soporte DB2 Everyplace.

Tabla 110. Métodos de la clase DB2eStatement

Tipo de retorno del método	Método
`void`	**enableDeletePhysicalRemove**(boolean `enable`) Habilita o inhabilita la eliminación física de registros, independientemente de los valores de sus bits de modificación, en una sentencia DELETE SQL.
`void`	**enableDirtyBitSetByApplication**(boolean `enable`) Habilita la modalidad de aplicación si se cumple la habilitación (enable es true). De lo contrario, habilita la modalidad de sistema.
`void`	**enableReadIncludeMarkedDelete** (boolean `enable`) Hace que los registros suprimidos lógicamente resulten visibles o invisibles.
`void`	**enableReorg** (boolean `enable`) Habilita o inhabilita la reorganización de bases de datos, por parte de DB2 Everyplace o explícitamente por parte del usuario mediante una sentencia REORG SQL.
`boolean`	**isEnabledDeletePhysicalRemove**() ¿Una sentencia de SQL eliminará físicamente los registros, independientemente de los valores de sus bits de modificación? ¿O bien los registros sólo se marcarán como ″suprimir″ (delete)?

Tabla 110. Métodos de la clase DB2eStatement (continuación)

Tipo de retorno del método	Método
`boolean`	**isEnabledDirtyBitSetByApplication()** ¿El sistema de base de datos está en modalidad de aplicación? ¿O está en modalidad de sistema?
`boolean`	**isEnabledReadIncludeMarkedDelete()** ¿Los registros suprimidos lógicamente resultan visibles desde sentencias de SQL? ¿O están ocultos para SQL?
`boolean`	**isEnabledReorg()** ¿Puede realizar DB2 Everyplace una reorganización de la base de datos o la debe efectuar explícitamente el usuario mediante una sentencia REORG SQL? ¿O las sentencias REORG SQL están restringidas y está inhabilitada la reorganización automática de tablas de bases de datos creadas por el usuario?

En estos ejemplos, `st` representa un objeto Statement y `rs` representa un objeto ResultSet.

Para eliminar físicamente algunos registros de la tabla T ignorando el estado de los bits de modificación:

```
DB2eStatement db2e_st = (DB2eStatement) st;
db2e_st.enableDeletePhysicalRemove(true);
st.executeUpdate("DELETE FROM T WHERE X<>0");
```

Para leer todos los registros de la tabla T que tengan establecido el bit de modificación, incluyendo aquéllos que lo tienen marcado como DELETE:

```
DB2eStatement db2e_st = (DB2eStatement) st;
db2e_st.enableReadIncludeMarkedDelete(true);
rs = st.executeQuery("SELECT * FROM T WHERE $dirty<>0");
```

Para limpiar el bit de modificación de un registro de la tabla T:

```
DB2eStatement db2e_st = (DB2eStatement) st;
db2e_st.enableDirtyBitSetByApplication(true);
st.executeUpdate("UPDATE T SET $dirty=0 WHERE $dirty>0");
```

Tareas afines:

- "Desarrollo de aplicaciones Java de DB2 Everyplace" en la página 17

Consulta relacionada:

- "Visión general del soporte de JDBC de DB2 Everyplace" en la página 287
- "Mensajes de SQLState notificados por JDBC" en la página 193
- "SQLGetStmtAttr—Obtener el valor actual de un atributo de sentencia" en la página 260
- "SQLSetStmtAttr—Establecer opciones referentes a una sentencia" en la página 276

Interfaces en el paquete javax.sql

Este capítulo proporciona información sobre los métodos JDBC del paquete javax.sql. Los temas que se tratan son:

- "Interfaz DataSource" en la página 306

Interfaz DataSource

Fábrica de conexiones con la fuente de datos física a la que representa este objeto DataSource. El método preferido de obtener una conexión es una sustitución del recurso de DriverManager, un objeto DataSource.

Una instancia de un objeto DataSource puede utilizarse en un programa autónomo para crear objetos Connection. En el ejemplo siguiente, se utiliza una instancia de DB2eDataSource para crear una conexión (Connection) con una base de datos de DB2 Everyplace con el url ″jdbc:db2e:myDataSource″:

```
com.ibm.db2e.jdbc.DB2eDataSource ds = new com.ibm.db2e.jdbc.DB2eDataSource();
    ds.setUrl("jdbc:db2e:myDataSource");
    Connection con = ds.getConnection();
```

paquete javax.sql

DataSource de interfaz pública

Tabla 111 y Tabla 112 lista las propiedades de la interfaz DataSource a los que da soporte DB2 Everyplace. A las propiedades se puede acceder utilizando los métodos ″getter″ y ″setter″. (Las propiedades de DataSource sigan el convenio especificado para las propiedades de los componentes de ™ en la Especificación JavaBeans 1.01.)

Tabla 111. Propiedades de DataSource estándar a las que da soporte DB2 Everyplace

Nombre de propiedad	Tipo	Descripción
descripción	String	descripción de esta fuente de datos
contraseña	String	contraseña de la base de datos
usuario	String	nombre de la cuenta del usuario

La Tabla 112 lista las propiedades soportadas de la interfaz DataSource que son específicas para DB2 Everyplace.

Tabla 112. Propiedades específicas de DB2 Everyplace para la interfaz DataSource

Nombre de propiedad	Tipo	Descripción
codificación	String	codificación de caracteres
URL	String	fuente de datos

La Tabla 113 lista los métodos de la interfaz DataSource a los que da soporte DB2 Everyplace.

Tabla 113. Métodos de la interfaz DataSource

Tipo de valor de retorno del método	Método
`Connection`	**getConnection**() Intenta establecer una conexión con la fuente de datos a la que representa este objeto DataSource.
`Connection`	**getConnection** (java.lang.String `username`, java.lang.String `password`) Intenta establecer una conexión con la fuente de datos a la que representa este objeto DataSource.
`int`	**getLoginTimeout**() Obtiene el tiempo máximo en segundos que puede esperar esta fuente de datos mientras intenta conectarse a una base de datos.
`java.io.PrintWriter`	**getLogWriter**() Recupera el transcriptor de anotaciones para este objeto DataSource.
`void`	**setLoginTimeout**(int `seconds`) Establece el tiempo máximo en segundos que esperará esta fuente de datos mientras intenta conectarse a una base de datos.
`void`	**setLogWriter** (java.io.PrintWriter `out`) Establece el transcriptor de anotaciones para este objeto DataSource para el objeto java.io.PrintWriter en concreto.

Tareas afines:

- "Desarrollo de aplicaciones Java de DB2 Everyplace" en la página 17

Consulta relacionada:

- "Visión general del soporte de JDBC de DB2 Everyplace" en la página 287
- "Mensajes de SQLState notificados por JDBC" en la página 193

Clases .NET soportadas

Este capítulo contiene información sobre las clases .NET a las que da soporte DB2 Everyplace. Este capítulo contiene los apartados siguientes:

- "Miembros de DB2eCommandBuilder"
- "Miembros de DB2eCommand" en la página 308
- "Miembros de DB2eConnection" en la página 309
- "Miembros de DB2eDataAdapter" en la página 309
- "Miembros de DB2eDataReader" en la página 310
- "Miembros de DB2eError" en la página 312
- "Miembros de DB2eException" en la página 312
- "Miembros de DB2eParameter" en la página 312
- "Miembros de DB2eTransaction" en la página 313
- "Enumeración de DB2eType" en la página 314

Miembros de DB2eCommandBuilder

Tabla 114. Métodos estáticos públicos (Compartidos)

Método	Descripción
DeriveParameters	Recupera información de parámetro del procedimiento almacenado especificado en *DB2eCommand* y puebla la colección de Parámetros del objeto *DB2eCommand* especificado.

Tabla 115. Constructores de instancias públicas

Constructor	Descripción
DB2eCommandBuilder()	Sobrecargado. Inicialice una nueva instancia de la clase de *DB2eCommandBuilder*.
DB2eCommandBuilder(DB2eDataAdapter)	Sobrecargado. Inicialice una nueva instancia de la clase de *DB2eCommandBuilder* con el objeto *DB2eDataAdapter* asociado.

Tabla 116. Propiedades de instancias públicas

Propiedad	Descripción
DataAdapter	Obtiene o establece un objeto *DB2eDataAdapter* para el que este objeto *DB2eCommandBuilder* generará sentencias SQL.

Tabla 117. Métodos de instancias públicas

Método	Descripción
GetDeleteCommand	Obtiene el objeto *DB2eCommand* generado automáticamente para realizar supresiones en la base de datos.
GetInsertCommand	Obtiene el objeto *DB2eCommand* generado automáticamente para realizar inserciones en la base de datos.
GetUpdateCommand	Obtiene el objeto *DB2eCommand* generado automáticamente para realizar actualizaciones en la base de datos.
RefreshSchema	Renueva la información de esquema de base de datos utilizada para generar sentencias INSERT, UPDATE o DELETE.

Tabla 118. Métodos de instancias protegidas

Método	Descripción
Dispose	Sobrecargado.

Miembros de DB2eCommand

Tabla 119. Constructores de instancias públicas

Constructor	Descripción
DB2eCommand()	Sobrecargado. Inicialice una nueva instancia de la clase de *DB2eCommand*.
DB2eCommand(string)	Sobrecargado. Inicialice una nueva instancia de la clase de *DB2eCommand* con el texto de la consulta.
DB2eCommand(string, DB2eConnection)	Sobrecargado. Inicialice una nueva instancia de la clase de *DB2eCommand* con el texto de la consulta y un objeto de *DB2eConnection*.
DB2eCommand(string, DB2eConnection, DB2eTransaction)	Sobrecargado. Inicialice una nueva instancia de la clase de *DB2eCommand* con el texto de la consulta, un objeto de *DB2eConnection* y el objeto de *DB2eTransaction*.

Tabla 120. Propiedades de instancias públicas

Propiedad	Descripción
CommandText	Obtiene o establece el procedimiento almacenado o la sentencia SQL que ha de ejecutarse frente a la base de datos.
CommandType	Obtiene o establece un valor indicando el modo en que se interpreta la propiedad de *CommandText*.
Connection	Obtiene o establece el *DB2eConnection* que utiliza esta instancia del *DB2eCommand*.
DesignTimeVisible	Obtiene o establece un valor que indica si el objeto de mandatos debería ser visible o no en un control de interfaz personalizado.
Parámetros	Obtiene el *DB2eParameterCollection*.
Transaction	Obtiene o establece el *DB2eTransaction* en el que se ejecuta el *DB2eCommand*.
UpdatedRowSource	Obtiene o establece un valor que especifica el modo en que el método *Update* debería aplicar resultados de mandato al *DataRow*.

Tabla 121. Métodos de instancias públicas

Método	Descripción
CreateParameter	Crea una instancia nueva del objeto de *DB2eParameter*.
Dispose	Sobrecargado. Limpiar.
EnableDeletePhysicalRemove	Habilita o inhabilita la eliminación física de los registros.
EnableDirtyBitSetByApplication	Habilita la modalidad de aplicación si se cumple la habilitación (enable es true). De lo contrario, habilita la modalidad de sistema.
EnableReadIncludeMarkedDelete	Hace que los registros suprimidos lógicamente resulten visibles o invisibles.
EnableReorg	Habilita o inhabilita la reorganización de la base de datos por parte de DB2 Everyplace o explícitamente por parte del usuario con una sentencia REORG de SQL.
ExecuteNonQuery	Ejecuta una sentencia SQL frente a *Connection* y devuelve el número de filas afectados.
ExecuteReader	Sobrecargado. Envía el *CommandText* a Connection y crea un *DB2eDataReader*.
ExecuteScalar	Ejecuta la consulta y devuelve la primera columna de la primera fila en el conjunto de resultados que devuelve la consulta. Se ignoran las filas o columnas adicionales.
IsEnabledDeletePhysicalRemove	Compruebe si se ha habilitado o no la eliminación física. Si se ha habilitado, devolverá verdadero (true); en caso contrario devolverá falso (false).
IsEnabledDirtyBitSetByApplication	Compruebe si el sistema de la base de datos está en la modalidad de aplicación o en la modalidad de sistema. Si se ha habilitado, devolverá verdadero (true); en caso contrario devolverá falso (false).
IsEnabledReadIncludeMarkedDelete	Compruebe si los registros suprimidos lógicamente están visibles o no para la aplicación. Devuelve verdadero (true) si los registros suprimidos lógicamente están visibles para la aplicación; en caso contrario devuelve falso (false).
IsEnabledReorg	Comprueba si está habilitada la reorganización de la base de datos. Devuelve verdadero (true) si está habilitada; en caso contrario devuelve falso (false).
Prepare	Crea una versión preparada (o compilada) del mandato en la base de datos.

Miembros de DB2eConnection

Tabla 122. Métodos estáticos públicos (Compartidos)

Método	Descripción
ReleaseObjectPool	Indica que el descriptor de contexto de entorno de DB2e pueda liberarse cuando se libera la última conexión subyacente.

Tabla 123. Constructores de instancias públicas

Constructor	Descripción
DB2eConnection()	Sobrecargado. Inicialice una nueva instancia de la clase de *DB2eConnection*.
DB2eConnection(string)	Sobrecargado. Inicialice una newInitialize una instancia nueva de la clase de *DB2eConnection* con una serie de conexión especificada.

Tabla 124. Propiedades de instancias públicas

Propiedad	Descripción
ConnectionString	Obtiene o establece la serie utilizada para abrir una base de datos.
ConnectionTimeout	Obtiene o establece el tiempo que ha de esperarse mientras se intenta establecer una conexión antes de finalizar el intento y de generar un error.
Base de datos	Obtiene el nombre de la base de datos actual o la base de datos que ha de utilizarse después de que se abra una conexión.
ServerVersion	Obtiene una serie que contiene la versión del servidor al que se conecta el cliente.
State	Obtiene el estado actual de la conexión.

Tabla 125. Métodos de instancias públicas

Método	Descripción
BeginTransaction	Sobrecargado. Comienza una transacción en la base de datos.
ChangeDatabase	Cambia la base de datos actual asociada con una *DB2eConnection* abierta.
Close	Cierra la conexión con la base de datos. Este es el método preferido de cerrar cualquier conexión abierta.
CreateCommand	Crea y devuelve un objeto de *DB2eCommand* asociado con el *DB2eConnection*.
Open	Abre una conexión con una fuente de datos con los valores de propiedad especificados mediante la *ConnectionString*.

Tabla 126. Sucesos de instancias públicas

Suceso	Descripción
InfoMessage	Se produce cuando el DB2 Everyplace envía un mensaje informativo o de aviso.
StateChange	Se produce cuando cambia el estado de la conexión.

Miembros de DB2eDataAdapter

Tabla 127. Constructores de instancias públicas

Constructor	Descripción
DB2eDataAdapter()	Sobrecargado. Inicialice una nueva instancia de la clase *DB2eDataAdapter*.
DB2eDataAdapter(DB2eCommand)	Sobrecargado. Inicialice una nueva instancia de la clase *DB2eDataAdapter* con la sentencia SELECT de SQL especificada.
DB2eDataAdapter(string, DB2eConnection)	Sobrecargado. Inicialice una nueva instancia de la clase *DB2eDataAdapter* con la sentencia SELECT de SQL especificada y un objeto de *DB2eConnection*.
DB2eDataAdapter(string, string)	Sobrecargado. Inicialice una nueva instancia de la clase *DB2eDataAdapter* con la sentencia SELECT de SQL especificada y una serie de conexión.

Tabla 128. Propiedades de instancias públicas

Propiedad	Descripción
AcceptChangesDuringFill (se hereda de **DataAdapter**)	Obtiene o establece un valor que indica si se llama o no *AcceptChanges* en una *DataRow* después de que se añada a la *DataTable*.

Tabla 128. Propiedades de instancias públicas (continuación)

Propiedad	Descripción
ContinueUpdateOnError (se hereda de **DataAdapter**)	Obtiene o establece un valor que especifica si ha de generarse o no una excepción o la fila errónea cuando se encuentra un error durante una actualización de filas.
DeleteCommand	Obtiene o establece una sentencia SQL o procedimiento almacenado utilizado para suprimir registros en la base de datos.
InsertCommand	Obtiene o establece una sentencia SQL o procedimiento almacenado utilizado para insertar registros nuevos en la fuente de datos.
MissingMappingAction (se hereda de **DataAdapter**)	Determina la acción a adoptar cuando los datos de entrada no tienen una columna o tabla coincidente.
MissingSchemaAction (se hereda de **DataAdapter**)	Determina la acción a adoptar cuando el esquema *DataSet* existente no coincide con los datos de entrada.
SelectCommand	Obtiene o establece una sentencia SQL o procedimiento almacenado utilizado para seleccionar registros en la base de datos.
TableMappings (se hereda de **DataAdapter**)	Obtiene una colección que proporciona la asignación maestra entre una tabla fuente y una *DataTable* .
UpdateCommand	Obtiene o establece una sentencia SQL o procedimiento almacenado utilizado para actualizar registros en la base de datos.

Tabla 129. Métodos de instancias públicas

Método	Descripción
Clone	Crea un objeto que contiene toda la información relevante necesaria para generar un proxy utilizado para comunicarse con un objeto remoto.
Fill (se hereda de **DbDataAdapter**)	Añade o renueva filas a un *DataSet* o *DataTable* para que coincidan con las de la fuente de datos.
FillSchema (se hereda de **DbDataAdapter**)	Añade una *DataTable* a un *DataSet* y configura el esquema para que coincida con el de la fuente de datos.
GetFillParameters (se hereda de **DbDataAdapter**)	Obtiene los parámetros que establece el usuario al ejecutar una sentencia SELECT de SQL.
Update (se hereda de **DbDataAdapter**)	Invoca las sentencias INSERT, UPDATE o DELETE respectivos para cada fila insertada, actualizada o suprimida del DataSet.

Tabla 130. Sucesos de instancias públicas

Suceso	Descripción
FillError (se hereda de **DbDataAdapter**)	Se devuelve cuando se produce un error durante una operación de relleno.
RowUpdated	Se produce durante una operación de actualización después de que se ejecute un mandato frente a la base de datos.
RowUpdating	Se produce durante una *Update* antes de que se ejecute un mandato frente a la base de datos.

Miembros de DB2eDataReader

Tabla 131. Propiedades de instancias públicas

Propiedad	Descripción
Depth	Obtiene un valor que indica la profundidad de anidamiento para la fila actual.
FieldCount	Obtiene el número de columnas de la fila actual.
IsClosed	Indica si se cierra o no el *DB2eDataReader*.
Item	Sobrecargado. Obtiene el valor de la columna especificada en su formato nativo que se proporciona en el ordinal de la columna.

Tabla 131. Propiedades de instancias públicas (continuación)

Propiedad	Descripción
RecordsAffected	Obtiene el número de filas cambiadas, insertadas o suprimidas mediante la ejecución de la sentencia de SQL.

Tabla 132. Métodos de instancias públicas

Método	Descripción
Close	Cierra el objeto de *DB2eDataReader*.
GetByte	Obtiene el valor de la columna especificada como un byte.
GetBytes	Lee una corriente de bytes desde el desplazamiento de la columna especificada al almacenamiento intermedio como matriz, comenzando a partir del desplazamiento de almacenamiento intermedio concreto.
GetDataTypeName	Obtiene el nombre del tipo de datos fuente.
GetDate	Obtiene el valor de la columna especificada como objeto de *DateTime*.
GetDateTime	Obtiene el valor de la columna especificada como objeto de *DateTime*.
GetDecimal	Obtiene el valor de la columna especificada como objeto de *Decimal*.
GetDouble	Obtiene el valor de la columna especificada como número de coma flotante de precisión doble.
GetFieldType	Obtiene el *Type* que es el tipo de datos del objeto.
GetFloat	Obtiene el valor de la columna especificada como número de coma flotante de precisión única.
GetInt16	Obtiene el valor de la columna especificada como entero con signo de 16 bits.
GetInt32	Obtiene el valor de la columna especificada como entero con signo de 32 bits.
GetInt64	Obtiene el valor de la columna especificada como entero con signo de 64 bits.
GetName	Obtiene el nombre de la columna especificada.
GetOrdinal	Obtiene el ordinal de columna, dado el nombre de la columna.
GetSchemaTable	Devuelve una *DataTable* que describe los metadatos de columna del *DB2eDataReader*.
GetString	Obtiene el valor de la columna especificada como una serie.
GetTime	Obtiene el valor de la columna especificada como objeto de *TimeSpan*.
GetValue	Obtiene el valor de la columna en ordinal especificado en su formato nativo.
GetValues	Obtiene todas las columnas de atributos de la fila actual.
IsDBNull	Obtiene un valor que indica si la columna contiene o no valores inexistentes o que falten.

Tabla 132. Métodos de instancias públicas (continuación)

Método	Descripción
NextResult	Avanza el *DB2eDataReader* al siguiente resultado, al leer los resultados de las sentencias SQL de proceso por lotes. DB2 Everyplace no da soporte en la actualidad a las sentencias de SQL de proceso por lotes.
Read	Avanza el *DB2eDataReader* al siguiente registro.

Miembros de DB2eError

Tabla 133. Propiedades de instancias públicas

Propiedad	Descripción
Message	Obtiene una breve descripción del error.
NativeError	Obtiene la información de error de DB2 Everyplace.
SQLState	Obtiene el código de error de cinco caracteres que sigue al estándar de SQL de ANSI para la base de datos.

Miembros de DB2eException

Tabla 134. Propiedades de instancias públicas

Propiedad	Descripción
Errors	Obtiene una colección de uno o más objetos de *DB2eError* que proporcionan información detallada sobre las excepciones generadas por el Proveedor de datos .NET de DB2 Everyplace.
Message	Obtiene la descripción textual que describe el error.

Miembros de DB2eParameter

Tabla 135. Constructores de instancias públicas

Constructor	Descripción
DB2eParameter()	Sobrecargado. Inicialice una nueva instancia de la clase de *DB2eParameter*.
DB2eParameter(string, object)	Sobrecargado. Inicialice una nueva instancia de la clase de *DB2eParameter* con el nombre de parámetro y el valor del parámetro.
DB2eParameter(string, DB2eType)	Sobrecargado. Inicialice una nueva instancia de la clase de *DB2eParameter* con el nombre de parámetro y el tipo de datos.
DB2eParameter(string, DB2eType, int)	Sobrecargado. Inicialice una nueva instancia de la clase de *DB2eParameter* con el nombre de parámetro, el tipo de datos y la anchura.
DB2eParameter(string, DB2eType, int, string)	Sobrecargado. Inicialice una nueva instancia de la clase de *DB2eParameter* con el nombre de parámetro, el tipo de datos, la anchura y el nombre de columna fuente.

Tabla 135. Constructores de instancias públicas (continuación)

Constructor	Descripción
DB2eParameter(string, DB2eType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object)	Sobrecargado. Inicialice una nueva instancia de la clase de *DB2eParameter* con el nombre de parámetro, el tipo de datos, la anchura, la dirección del parámetro, el booleano anulable, la precisión numérica, la escala, el nombre de columna fuente, la versión fuente y el valor del parámetro.

Tabla 136. Propiedades de instancias públicas

Propiedad	Descripción
DB2eType	Obtiene o establece el *DB2eType* del parámetro.
DbType	
Direction	Obtiene o establece un valor que indica si el parámetro sea sólo de entrada, sólo de salida, bidireccional o un parámetro de valor de retorno de procedimiento almacenado.
IsNullable	Obtiene o establece un valor que indica si el parámetro acepta valores nulos.
ParameterName	Obtiene o establece el nombre del *DB2eParameter*.
Precisión	Obtiene o establece el número máximo de dígitos utilizados para representar la propiedad *Value*.
Scale	Obtiene o establece el número de lugares decimales en los que se ha resuelto *Value*.
Size	Obtiene o establece el tamaño máximo, en bytes, de los datos de la columna.
SourceColumn	Obtiene o establece el nombre de la columna fuente correlacionada con el *DataSet* y utilizada para cargar o devolver el *Value*.
SourceVersion	Obtiene o establece la *DataRowVersion* a utilizar al cargar el *Value*.
Value	Obtiene o establece el valor del parámetro.

Tabla 137. Métodos de instancias públicas

Método	Descripción
ToString	Obtiene una serie que contiene el *ParameterName*.

Miembros de DB2eTransaction

Tabla 138. Propiedades de instancias públicas

Propiedad	Descripción
Connection	Especifica el objeto de *DB2eConnection* asociado con la transacción.
IsolationLevel	Especifica el *IsolationLevel* para esta transacción.

Tabla 139. Métodos de instancias públicas

Método	Descripción
Commit	Confirma la transacción de la base de datos.

Tabla 139. Métodos de instancias públicas (continuación)

Método	Descripción
Rollback	Retrotrae una transacción desde un estado pendiente.

Enumeración de DB2eType

Especifica el tipo de datos de un campo, propiedad o *DB2eParameter*.

[Visual Basic]

Public Enum DB2eType

[C#]

public enum DB2eType

La tabla siguiente muestra las correlaciones entre tipos de datos de *DB2eType*, los tipos de datos de DB2 Everyplace (se muestra entre paréntesis) y los tipos de estructura de .NET.

Tabla 140. Correlaciones de tipo de datos

Miembro	Descripción
SmallInt	Valor numérico exacto con precisión 5 y escale 0 (con signo: –32,768 <= n <= 32,767, sin signo: 0 <= n <= 65,535) (SMALLINT). Se correlaciona con Int16.
Entero	Valor numérico exacto con la precisión 10 y la escala 0 (con signo: –2[31] <= n <= 2[31] – 1, sin signo:0 <= n <= 2[32] – 1) (INTEGER). Se correlaciona con Int32.
Char	Serie de caracteres de longitud fija (CHAR). Se correlaciona con String.
VarChar	Serie de caracteres de longitud variable (VARCHAR). Se correlaciona con String.
Decimal	Valor numérico, exacto, con signo con una precisión de al menos p y escala s, dónde 1 <= p <= 31 y s <= p. (DECIMAL). Se correlaciona con Decimal.
Date	Datos de fecha en el formato aaaa-mm-dd (DATE). Se correlaciona con DateTime.
Time	Datos de hora en el formato hh:mm:ss (TIME). Se correlaciona con TimeSpan.
Timestamp	Datos de indicación de la fecha y hora en el formato aaaa-mm-dd-hh.mm.ss.zzzzzz (TIMESTAMP). Se correlaciona con DateTime.
Blob	Corriente de datos binarios (BLOB). Se correlaciona con una Matriz del tipo Byte.

Requisitos:

NameSpace: IBM.Data.DB2.DB2e Namespace

Assembly: IBM.Data.DB2.DB2e.dll

API C de IBM Sync Client

Este capítulo proporciona información sobre la API-C de IBM Sync Client. Los temas que se tratan son:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2"
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Descripciones de funciones de la API C de IBM Sync Client" en la página 321

Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2

Este apartado resume los principales cambios efectuados en la API C de IBM Sync Client de Versión 8.1:

- Ahora se dispone de tres descriptores de contexto: servicio, configuración y motor. (Si no desea realizar sincronizaciones, no es necesario que abra el descriptor de contexto del motor.)
- Las preferencias en la API C de IBM Sync Client Versión 8.1 no son persistentes y algunas de ellas, las que constituyen realmente información esencial, se han eliminado. Por ejemplo, el nombre de host, el puerto, el nombre de usuario y la contraseña de la anterior función `isyncSetPref` son ahora parámetros obligatorios en la función `iscOpenService` para abrir un descriptor de contexto de servicio.
- Ahora, la modalidad de sincronización es implícita a la aplicación, y el parámetro de modalidad de sincronización ya no es necesario cuando se invoca a una sincronización.
- La interfaz con la función de escucha de sincronización se basa ahora en los sucesos. Ahora, las estructuras de sucesos que contiene información de los mismos se pasan a la aplicación.
- El estado de sincronización de un conjunto de suscripción (de su última sincronización) es persistente y se puede consultar más tarde.
- Se ha eliminado la función de escucha por omisión. Cuando se necesita una acción por omisión para un suceso, la aplicación simplemente devuelve el código `ISCRTNCB_Default`.
- Ahora, DB2e Everyplace da soporte al cifrado de datos para proteger la tabla que contiene datos delicados. Cuando desarrolle una aplicación de cliente sincronizado para sincronizar tablas cifradas, puede implementar (en la función de escucha) la consulta del motor de sincronización para el nombre de usuario y la contraseña de DB2 Everyplace.
- Los registros rechazados (incluyendo aquéllos que contienen conflictos u operaciones no permitidas) se pasan ahora a la aplicación a través de la función de escucha.
- Ahora, la aplicación gestiona el archivo de anotaciones (LOGDB-ISYNC). Es decir, que el motor de sincronización de Versión 8.1 ya no genera el archivo de anotaciones (LOGDB-ISYN) en un idioma nativo, tal como sucedía en la versión 7.2.1. En lugar de esto, y a efectos de servicios, el motor de sincronización generará un archivo de rastreo (TRACE-ISYN), que sólo está en inglés
- El motor de IBM Sync Client almacena todos los archivos (incluidos la configuración, el archivo de rastreo, los datos y las preferencias (si es pertinente) en un directorio:

  - En sistemas operativos Windows CE®: \ (directorio raíz)
  - En sistemas operativos EPOC: C:\Systems\Data\ISync\
  - En sistemas operativos Palm: la memoria principal
  - En otros sistemas operativos: el directorio actual
- El funcionamiento de la API de IBM Sync Client Versión 7.2.1 se sigue soportando mediante un reiniciador de API (la biblioteca isynce), que manejará la compatibilidad anterior de la API. El reiniciador de API también genera el archivo de anotaciones (LOGDB-ISYN) en idiomas nativos, en el mismo directorio que en la Versión 7.2.1, es decir:
  - En sistemas operativos Windows CE®: \Archivos de programa\ISync\
  - En sistemas operativos EPOC: C:\Systems\Apps\ISync\
  - En sistemas operativos Palm: la memoria principal
  - En otros sistemas operativos: el directorio actual

  Además, las opciones ISYNCOPTION_SkipConfig e ISYNCOPTION_UseAppSignature no funcionarán con las funciones isyncGo e isyncSetSyncMoe.

  Nota: No es necesario instalar la biblioteca del reiniciador de API (isynce) si se elige utilizar la API de IBM Sync Client Versión 8.1.

La Tabla 141 lista las diferencias principales entre las funciones de la API C de IBM Sync Client Versión 8.1 y la de IBM Sync Client Versión 7.2.

Tabla 141. Comparación de la API C de IBM Sync Client Versión 8.1 con la de la Versión 7.2

Versión 8.1	Versión 7.2	Observaciones
iscGetVersion	isyncGetVersion	No se necesita ningún descriptor de contexto en iscGetVersion.
iscServiceOpen iscConfigOpeniscEngineOpen	isyncOpen	Es necesario abrir tres descriptores de contexto. El host, el puerto, el nombre de usuario y la contraseña se especifican en iscServiceOpen y no son persistentes.
iscServiceClose iscConfigClose iscEngineClose	isyncClose	Es necesario cerrar tres descriptores de contexto.
iscEngineSetListener	isyncSetListener	Han cambiado el prototipo de escucha y la interfaz.
(Ninguna)	isyncDefaultListener	No existen más funciones de escucha externas por omisión. Para el manejo de sucesos por omisión, devuelve el código ISCRTNCB_Default.
iscEngineSetPref iscEngineGetPref	isyncSetPref isyncGetPref	Sólo se requieren dos preferencias (rastreo y tiempo de espera). Estas preferencias no son persistentes.
iscEngineSync iscEngineSyncConfig	isyncGo	Ya no se requiere modalidad de sincronización. Sólo puede actualizar la configuración con iscEngineSyncConfig.

Tabla 141. Comparación de la API C de IBM Sync Client Versión 8.1 con la de la Versión 7.2 (continuación)

Versión 8.1	Versión 7.2	Observaciones

`iscConfigEnableSubsSet`
`iscConfigDisableSubsSet`
`iscConfigResetSubsSet`
	`isyncSetSyncMode`	No hay más valores de modalidad de sincronización generales.
		
		Se puede saltar (inhabilitar) la sincronización de un conjunto de suscripción mediante `iscConfigDisableSubsSet`.

`iscConfigOpenCursor`
`iscConfigCloseCursor`
`iscConfigGetNextSubsSet`
`iscConfigSubsSetIsEnabled`
`iscConfigSubsSetIsReset`
	`isyncGetFirstApp`
	`isyncGetNextApp`
		Abre un cursor antes de repetir conjuntos de suscripción.
		
		Para consultar un conjunto de suscripción, se necesita un ID de conjunto de suscripción.

`iscEngineGetInfo`
`iscConfigPurge`
`iscConfigGetSubsSetStatus`
		API C nuevas en la Versión 8.1.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Resumen de funciones de la API C de IBM Sync Client"
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Clave para las descripciones de funciones de la API C de IBM Sync Client" en la página 321

Resumen de funciones de la API C de IBM Sync Client

La Tabla 142 lista las funciones de la API C de IBM Sync Client soportadas por DB2 Everyplace e incluye la finalidad de cada una de ellas.

Tabla 142. Lista de funciones de la API C de IBM Sync Client

Nombre de la función	Finalidad
iscGetVersion	Obtiene el número de versión de la API C de Sync Client.

Tabla 143. Lista de funciones de la API de servicios de IBM

Nombre de la función	Finalidad
iscServiceOpen	Abre un nuevo servicio.
iscServiceOpenEx	Abre un nuevo servicio utilizando las propiedades.
iscServiceClose	Cierra un servicio.

Tabla 144. Lista de funciones de la API de configuración de IBM

Nombre de la función	Finalidad
iscConfigOpen	Abre una conexión con el almacén de configuración.
iscConfigClose	Cierra una conexión con el almacén de configuración.
iscConfigPurge	Reinicializa la configuración.
iscConfigOpenCursor	Obtiene (descriptor de contexto de) un cursor para repetir conjuntos de suscripción.
iscConfigCloseCursor	Dispone un cursor abierto.
iscConfigGetNextSubsSet	Obtiene la descripción del siguiente conjunto de suscripción (si lo hay).
iscConfigEnableSubsSet	Habilita un conjunto de suscripción para su sincronización.
iscConfigDisableSubsSet	Inhabilita la sincronización de un conjunto de suscripción.
iscConfigResetSubsSet	Devuelve un conjunto de suscripción a la modalidad de restablecimiento.
iscConfigSubsSetIsEnabled	Consulta si un conjunto de suscripción está habilitado para la sincronización.
iscConfigSubsSetIsReset	Consulta si un conjunto de suscripción está restablecido.
iscConfigGetSubsSetStatus	Consulta el estado de sincronización de la sincronización anterior.

Tabla 145. Lista de funciones de la API de IBM Sync Engine

Nombre de la función	Finalidad
iscEngineOpen	Abre un descriptor de contexto para el motor de sincronización.
iscEngineClose	Cierra un descriptor de contexto abierto para el motor de sincronización.
iscEngineGetInfo	Obtiene información general sobre el motor de sincronización.
iscEngineSetListener	Informa a la sincronización acerca de la función de escucha definida por el usuario que debe utilizar.
iscEngineListenerPF	Tipo de datos para la función de escucha definida por el usuario.
iscEngineSetPref	Establece una preferencia.
iscEngineGetPref	Recupera el valor de una preferencia.
iscEngineSync	Inicia una sesión de sincronización.
iscEngineSyncConfig	Sincroniza la configuración suministrada con el servidor.

Conceptos afines:

• "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client"
- "Clave para las descripciones de funciones de la API C de IBM Sync Client" en la página 321

Tipos de datos de la API C de IBM Sync Client

La Tabla 146 lista los nuevos tipos de datos definidos por la API C de IBM Sync Client. Cuando llame a las funciones de la API C, asegúrese de que el tipo de argumento se ajusta al prototipo de las funciones.

Tabla 146. Tipos de datos para la API C de IBM Sync Client

Tipo de datos	Descripción
isy_VOID	Tipo vacío
isy_INT	Entero
isy_UINT	Entero sin signo
isy_INT16	Entero de dos bytes
isy_UINT16	Entero sin signo de dos bytes
isy_INT32	Entero de cuatro bytes
isy_UINT32	Entero sin signo de cuatro bytes
isy_ULONG	Entero largo sin signo
isy_BYTE	Tipo de un byte
isy_WORD	Tipo de una palabra
isy_DWORD	Tipo de dos palabras
isy_TCHAR	Tipo carácter
isy_BOOL	Tipo booleano
HISCSERV	Tipo de datos del descriptor de contexto de servicio
HISCCONF	Tipo de datos del descriptor de contexto de configuración
HISCENG	Tipo de datos del descriptor de contexto del motor de sincronización
HISCCSR	Tipo de datos de un cursor de repetición para conjuntos de suscripción

ISCEVT	Tipo de datos de un suceso de escucha:
	`typedef struct {`
	`  isy_INT32  code;`
	`  isy_UINT32  type;`
	`  isy_INT32  retry;`
	`  ISCSTATE  state;`
	`  ISCLISTENARG  *info;`
	`} ISCEVT;`

ISCSTATE	Tipo de datos de estado del suceso:
	`typedef struct {`
	`  isy_TCHAR  currSubsSet[ISCLEN_SubsSetName];`
	`  isy_TCHAR  currSubs[ISCLEN_SubsName];`
	`  isy_UINT32  subsType;`
	`  isy_INT32  syncProg;`
	`} ISCSTATE;`

Tabla 146. Tipos de datos para la API C de IBM Sync Client (continuación)

Tipo de datos	Descripción

ISCLISTENARG	Tipo de datos de información para la función de escucha de sincronización, que consta de una lista de argumentos de serie (argc, argv):
	`typedef struct {`
	`  isy_INT32  argc;`
	`  isy_TCHAR  **argv;`
	`} ISCLISTENARG;`

ISCLISTENCOLUMN	Tipo de datos de información para la función de escucha de sincronización, que consta de una columna de tabla que contiene la posición de la columna, la secuencia de la clave primaria, el tipo de columna, el tamaño de los datos y los datos reales de la columna:
	`typedef struct {`
	`  isy_INT32  pos;`
	`  isy_INT32  pkseq;`
	`  isy_INT32  type;`
	`  isy_INT32  size;`
	`  isy_BYTE  *data;`
	`} ISCLISTENCOLUMN;`
	
	En un archivo de cabecera de DB2 Everyplace, `sqlcli.h`, se definen varias constantes de tipo de columna para el tipo de columna. Los datos de la columna se representan como una serie de texto terminada en nulo. Esto es así con excepción del tipo de columna blob, en que los datos reales de la columna (el campo de datos) se representan en forma de serie plana de bytes NO terminada en nulo. Además, en el campo de tamaño se indica su tamaño (número de bytes).

ISCLISTENCONFLICT	Tipo de datos de información para la función de escucha de sincronización, que consta de un registro de tabla que contiene el nombre de tabla, la operación, el número de columnas y una matriz de información de las columnas (`ISCLISTENCOLUMN`):
	`typedef struct {`
	`  isy_TCHAR  table[ISCLEN_Table];`
	`  isy_INT32  op;`
	`  isy_INT32  colc;`
	`  ISCLISTENCOLUMN *colv;`
	`} ISCLISTENCONFLICT;`
	
	El campo *op* indica la operación rechazada, que es una de las constantes de operación siguientes (con valores reales entre paréntesis):
	• ISCCONST_OpDelete (1)
	• ISCCONST_OpInsert (2)
	• ISCCONST_OpUpdate (3)

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "Clave para las descripciones de funciones de la API C de IBM Sync Client" en la página 321

Descripciones de funciones de la API C de IBM Sync Client

Este capítulo describe las funciones de API C de IBM Sync Client.

Clave para las descripciones de funciones de la API C de IBM Sync Client

Las descripciones para cada función de la API C de Sync Client contienen las secciones siguientes:

Finalidad
Brinda una breve visión general de lo que realiza la función.

Sintaxis
Contiene el prototipo C genérico. El prototipo genérico se utiliza para todos los entornos, incluido Windows.

Argumentos de la función
Lista los argumentos de cada función, junto con el tipo de datos, la descripción y el tipo de uso (entrada o salida) de cada argumento.

Uso Proporciona información sobre cómo utilizar la función y describe cualquier posible consideración especial.

Códigos de retorno
Lista todos los posibles códigos de retorno de la función.

Restricciones
Indica las diferencias o limitaciones existentes al aplicar cada función de la API C de Sync Client.

Referencias
Lista las funciones relacionadas de la API C de Sync Client.

Nota: En la API C de Sync Client no existe apartado de Diagnóstico.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317

iscGetVersion()

Finalidad: `iscGetVersion()` obtiene el número de versión de la API C de Sync Client.

Sintaxis:

```
isy_UINT32 iscGetVersion();
```

Argumentos de la función:

Ninguna.

Uso:

`iscGetVersion()` se utiliza para recuperar el número de versión de la API C de Sync Client. El número de versión se devuelve como entero sin signo de 32 bits con formato *0xmmnnrrxx*, donde *mm*, *nn* y *rr* son la representación hexadecimal, respectivamente, de los números de versión mayor, menor y modificado. *xx* son valores reservados.

Ejemplo::

```
isy_UINT32  version;
       int verMajor, verMinor, verModi;
       version = iscGetVersion();
       verMajor = (int) (version >> 24);
       verMinor = (int) ((version >> 16) & 0x000000FF);
       verModi  = (int) ((version >> 8)  & 0x000000FF);
```

Códigos de retorno:

Número de versión de la API C de Sync Client.

Restricciones:

Ninguna.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "iscEngineGetInfo()" en la página 340
- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Clave para las descripciones de funciones de la API C de IBM Sync Client" en la página 321
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317

iscServiceOpen()

Finalidad: `iscServiceOpen()` abre un nuevo descriptor de contexto de servicio.

Sintaxis:

```
isy_INT32 iscServiceOpen(
    isy_CONST isy_TCHAR* host,
    isy_CONST isy_TCHAR* port,
    isy_CONST isy_TCHAR* username,
    isy_CONST isy_TCHAR* password,
    isy_CONST isy_VOID*  reserved,
    HISCSERV*            phServ);
```

Argumentos de la función:

La Tabla 147 lista los argumentos válidos utilizados con la función `iscServiceOpen()`.

Tabla 147. Argumentos de iscServiceOpen()

Tipo de datos	Argumento	Uso	Descripción
isy_CONST isy_TCHAR*	*host*	entrada	Nombre de host o el IP
isy_CONST isy_TCHAR*	*port*	entrada	Número de puerto
isy_CONST isy_TCHAR*	*username*	entrada	Nombre de usuario para el servicio solicitado
isy_CONST isy_TCHAR*	*password*	entrada	Contraseña para el servicio solicitado
isy_CONST isy_TCHAR*	*reserved*	entrada	(Reservado)
HISCSERV*	*phServ*	salida	Descriptor de contexto para un servicio

Uso:

`iscServiceOpen()` se utiliza para solicitar un nuevo descriptor de contexto para un servicio específico identificado por el nombre de host y el número de puerto. El nombre de usuario y la contraseña se especifican al solicitar un servicio. Si resulta satisfactoria, se devuelve un descriptor de contexto de servicio (HISCSERV) a través de *phServ. De lo contrario, *phServ es NULL y se devuelve el código de retorno.

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_OutOfMemory : Falta memoria
- ISCRTN_ResourceBusy : Recurso bloqueado (por ejemplo, por otra aplicación)
- ISCRTN_NotPermitted : Recurso no accesible (por ejemplo, no legible)
- ISCRTN_NotFound : Recurso no encontrado (por ejemplo, vía de acceso no encontrada)
- ISCRTN_Failed : Otros casos

Restricciones:

Ninguna.

Conceptos afines:

- “La aplicación Sync Client C/C++ de ejemplo” en la página 119

Tareas afines:

- “Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++” en la página 15

Consulta relacionada:

- “Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2” en la página 315
- “Tipos de datos de la API C de IBM Sync Client” en la página 319
- “Resumen de funciones de la API C de IBM Sync Client” en la página 317
- “iscServiceClose()” en la página 325
- “iscServiceOpenEx()” en la página 324

iscServiceOpenEx()

Finalidad: `iscServiceOpenEx()` abre un nuevo descriptor de contexto de servicio basado en una matriz de propiedad.

Sintaxis:

```
isy_INT32 iscServiceOpenEx(
    isy_CONST isy_TCHAR* URL,
    ISCPROPERTY*          property,
    isy_INT32            propertyNum,
    HISCSERV*            phServ);
```

Argumentos de la función:

La Tabla 148 lista el argumento válido utilizado con la función `iscServiceOpenEx()`.

Tabla 148. Argumento de iscServiceOpenEx()

Tipo de datos	Argumento	Uso	Descripción
isy_CONST isy_TCHAR	*URL*	entrada	Información de servidor como serie de URL

ISCPROPERTY	property	entrada	Matriz de propiedades del tipo ISCPROPERTY:
			`typedef struct {`
			`  isy_TCHAR *key;  //serie de ID de propiedad`
			`  isy_TCHAR *value; //serie de valor de propiedad`
			`} ISCPROPERTY;`
			
			Hay tres propiedades disponibles:
			• *isync.user* — Nombre de usuario de Cliente de sincronización
			• *isync.password* — Contraseña de Cliente de sincronización
			• *isync.encoding* — Codificación de caracteres de los datos de destino
			
			El nombre de usuario y las propiedades de contraseña son obligatorias.

isy_INT32	propertyNum	entrada	Número de propiedades.
HISCSERV	phServ	salida	Descriptor de contexto para un servicio.

Uso:

`iscServiceOpenEx()` se utiliza para solicitar un nuevo descriptor de contexto para un servicio específico desde un servidor con valores representados como matriz de propiedad. El servidor se identifica por medio de una serie de Uniform Resource Locator (URL), la cual es posible que contenga el protocolo, el nombre de sistema principal (o IP) y el número de puerto. Si el Servidor de sincronización está configurado para el Secure Socket Layer (SSL), la parte de protocolo del URL debe ser ″https://″, en caso contrario, será ″http://″. El número de puerto puede omitirse y los puertos por omisión para SSL y no SSL son el puerto 443 y el puerto 80, respectivamente. Todos los valores (incluyendo el nombre de usuario y la contraseña) se especifican en la matriz de propiedad. Si resulta satisfactoria, se devuelve un descriptor de contexto de servicio (HISCSERV) a través de *phServ*; en caso contrario, *phServ* es NULL y se devuelve el código de error. Al finalizar, el descriptor de contexto de servicio se cierra con `iscServiceClose()`.

Ejemplo:

```
      int rc = 0;
HISCSERV hSyncServ;
ISCPROPERTY properties[3] = {{"isync.user", "myUserName"},
                {"isync.password", "myPassword"},
                {"isync.encoding", "ISO8859_1"}}
rc = iscServiceOpenEx("http://localhost.mycom.com:80", properties, 3, &hSyncServ);
```

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_OutOfMemory : Falta memoria
- ISCRTN_ResourceInUse : Recurso bloqueado (por ejemplo, por otra aplicación)
- ISCRTN_NotPermitted : Recurso no accesible (por ejemplo, no legible)
- ISCRTN_NotFound : Recurso no encontrado (por ejemplo, vía de acceso no encontrada)
- ISCRTN_Failed : Otros casos

Restricciones:

Ninguna.

Consulta relacionada:

- "iscConfigClose()" en la página 327

iscServiceClose()

Finalidad: `iscServiceOpen()` abre un descriptor de contexto de servicio abierto.

Sintaxis:

```
isy_INT32 iscServiceClose(
    HISCSERV  hServ);
```

Argumentos de la función:

La Tabla 149 lista el argumento válido utilizado con la función `iscServiceClose()`.

Tabla 149. Argumento de iscServiceClose()

Tipo de datos	Argumento	Uso	Descripción
HISCSERV	*hServ*	entrada	Descriptor de contexto de servicio

Uso:

Utilice `iscServiceClose()` para liberar el almacenamiento de un descriptor de contexto previamente abierto.

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_Failed : Otros casos

Restricciones:

El hecho de realizar varias llamadas a `iscServiceClose()` puede ocasionar errores y se debe evitar.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscServiceOpen()" en la página 322

iscConfigOpen()

Finalidad: `iscConfigOpen()` abre una conexión con el almacén de configuración.

Sintaxis:

```
isy_INT32 iscConfigOpen(
    HISCSERV   hServ,
    isy_TCHAR *path,
    HISCCONF  *phConf);
```

Argumentos de la función:

La Tabla 150 lista los argumentos válidos utilizados con la función `iscConfigOpen()`.

Tabla 150. Argumentos de iscConfigOpen()

Tipo de datos	Argumento	Uso	Descripción
HISCSERV	*hServ*	entrada	Descriptor de contexto de servicio
isy_TCHAR*	*path*	entrada	Vía de acceso del directorio de trabajo
HISCCONF*	*phConf*	salida	Conexión de configuración

Uso:

`iscConfigOpen()` abre una conexión con el almacén de configuración que se especifica en la vía de acceso indicada para un servicio específico. Si resulta satisfactoria, se devuelve un descriptor de contexto de configuración (HISCCONF) a través de *phServ. De lo contrario, *phServ es NULL y se devuelve el código de retorno. Si se trata de un nuevo servicio (ya sea un nuevo host o un nuevo puerto), se crea una nueva configuración vacía para ese servicio.

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_OutOfMemory : Falta memoria
- ISCRTN_ResourceBusy : Recurso bloqueado (por ejemplo, por otra aplicación)
- ISCRTN_NotPermitted : Recurso no accesible (por ejemplo, no legible)
- ISCRTN_NotFound : Recurso no encontrado (por ejemplo, vía de acceso no encontrada)
- ISCRTN_Failed : Otros casos

Restricciones:

Ninguna.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscConfigClose()"

iscConfigClose()

Finalidad: `iscConfigClose()` cierra una conexión de almacén de configuración abierta.

Sintaxis:

```
isy_INT32 iscConfigClose(
    HISCCONF    hConf);
```

Argumentos de la función:

La Tabla 151 lista el argumento válido utilizado con la función `iscConfigClose()`.

Tabla 151. Argumento de iscConfigClose()

Tipo de datos	Argumento	Uso	Descripción
HISCCONF	*hConf*	entrada	Conexión de configuración

Uso:

`iscConfigClose()` cierra una conexión de almacén de configuración previamente abierta.

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_Failed : Otros casos

Restricciones:

Ninguna.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319

- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscConfigOpen()" en la página 326

iscConfigPurge()

Finalidad: `iscConfigPurge()` vacía toda la información de suscripciones del almacén de configuración.

Sintaxis:

```
isy_INT32 iscConfigPurge(
    HISCCONF    hConf);
```

Argumentos de la función:

La Tabla 152 lista el argumento válido utilizado con la función `iscConfigPurge()`.

Tabla 152. Argumento de iscConfigPurge()

Tipo de datos	Argumento	Uso	Descripción
HISCCONF	*hConf*	entrada	Conexión de configuración

Uso:

`iscConfigPurge()` elimina toda la información de suscripciones contenida en el almacén de configuración. Durante la siguiente sincronización, el motor capta de nuevo la configuración del servidor y realiza una renovación total de todos los conjuntos de suscripción.

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_Failed : Otros casos

Restricciones:

Ninguna.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscConfigResetSubsSet()" en la página 334

iscConfigOpenCursor()

Finalidad: `iscConfigOpenCursor()` obtiene un cursor para procesar de forma repetida todos los conjuntos de suscripción.

Sintaxis:

```
isy_INT32 iscConfigOpenCursor(
    HISCCONF    hConf,
    HISCCSR    *phCursor);
```

Argumentos de la función:

La Tabla 153 lista los argumentos válidos utilizados con la función `iscConfigOpenCursor()`.

Tabla 153. Argumentos de iscConfigOpenCursor()

Tipo de datos	Argumento	Uso	Descripción
HSYNCCONF	*hConf*	entrada	Conexión de configuración
HISCCSR*	*phCursor*	salida	Cursor devuelto para la repetición de conjuntos de suscripción

Uso:

Cuando se necesite un proceso de repetición sobre todos los conjuntos de suscripción, utilice `iscConfigOpenCursor()` para obtener un cursor apropiado. A continuación, utilice `iscConfigGetNextSubsSet()` para obtener cada conjunto de suscripción y la descripción correspondiente al mismo.

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_Failed : Otros casos

Restricciones:

Cuando se llama a `iscConfigOpenCursor()`, se invalidad todos los cursores que se habían abierto previamente, y se deben cerrar. Cualquier intento de procesar conjuntos de suscripción con cursores cerrados genera el código de retorno ISCRTN_Failed. Es decir, que no se puede anidar una repetición de los conjuntos de suscripción dentro de otra repetición. De forma parecida, los cursores abiertos se invalidan cuando se sincroniza la configuración (utilizando `iscEngineSync()` o `iscEngineSyncConfig()`).

Conceptos afines:

- “La aplicación Sync Client C/C++ de ejemplo” en la página 119

Tareas afines:

- “Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++” en la página 15

Consulta relacionada:

- “Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2” en la página 315
- “Tipos de datos de la API C de IBM Sync Client” en la página 319
- “Resumen de funciones de la API C de IBM Sync Client” en la página 317
- “iscConfigCloseCursor()” en la página 330
- “iscConfigGetNextSubsSet()” en la página 331

iscConfigCloseCursor()

Finalidad: `iscConfigCloseCursor()` dispone un cursor abierto.

Sintaxis:

```
isy_INT32 iscConfigCloseCursor(
    HISCCONF    hConf,
    HISCCSR     hCursor);
```

Argumentos de la función:

La Tabla 154 lista los argumentos válidos utilizados con la función `iscConfigCloseCursor()`.

Tabla 154. Argumentos de iscConfigCloseCursor()

Tipo de datos	Argumento	Uso	Descripción
HISCCONF	*hConf*	entrada	Conexión de configuración
HISCCSR	*hCursor*	entrada	Cursor para repetir conjuntos de suscripción

Uso:

Cuando se abra un cursor con `iscConfigOpenCursor()` pero el cursor no sea necesario, ciérrelo con `iscConfigCloseCursor()`. De no hacerlo, el cursor abierto puede ocasionar pérdidas de memoria u otros problemas de coherencia de la configuración. No intente utilizar el descriptor de contexto cerrado una vez que se haya cerrado el cursor, puesto que podría ocasionar errores inesperados.

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_Failed : Otros casos

Restricciones:

Ninguna.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscConfigOpenCursor()" en la página 328
- "iscConfigGetNextSubsSet()" en la página 331

iscConfigGetNextSubsSet()

Finalidad: `iscConfigGetNextSubsSet()` obtiene la descripción (si la hay) de un cursor y lo mueve al siguiente conjunto de suscripción.

Sintaxis:

```
isy_INT32 iscConfigGetNextSubsSet(
    HISCCONF    hConf,
    HISCCSR     hCursor,
    isy_TCHAR*  id,
    isy_TCHAR*  name);
```

Argumentos de la función:

La Tabla 155 lista los argumentos válidos utilizados con la función `iscConfigGetNextSubsSet()`.

Tabla 155. Argumentos de iscConfigGetNextSubsSet()

Tipo de datos	Argumento	Uso	Descripción
HISCCONF	*hConf*	entrada	Conexión de configuración
HISCCSR	*hCursor*	entrada	Cursor para repetir conjuntos de suscripción
isy_TCHAR*	*id*	salida	ID del conjunto de suscripción
isy_TCHAR*	*nombre*	salida	Nombre del conjunto de suscripción

Uso:

`iscConfigGetNextSubsSet()` obtiene el ID de conjunto de suscripción del servidor, recupera el nombre del conjunto de suscripción (si lo hay) y mueve el cursor al siguiente conjunto de suscripción.

Ejemplo:

```
isy_TCHAR  id[ISCLEN_SubsSetID];
isy_TCHAR  name[ISCLEN_SubsSetName];
isy_INT32  isReset, isEnabled;
HISCCSR    hCursor;
isy_INT32  rc;

// inicia la iteración de todos los conjuntos de suscripción
rc = iscConfigOpenCursor(hConf, &hCursor);
while (rc == ISCRTN_Succeeded) {
    rc = iscConfigGetNextSubsSet(hConf, hCursor, id, name);
    if (rc == ISCRTN_Succeeded) {
        isReset = iscConfigSubsSetIsReset(hConf, id);
        isEnabled = iscConfigSubsSetIsEnabled(hConf, id);
        // proceso del conjunto de suscripción
        ...
        // obtención de la suscripción siguiente
    } // final de proceso
} // final de iteración
iscConfigCloseCursor(hConf, hCursor);
```

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_Empty : No existen más conjuntos de suscripción
- ISCRTN_Failed : Otros casos

Restricciones:

Ninguna.

Conceptos afines:

- “La aplicación Sync Client C/C++ de ejemplo” en la página 119

Tareas afines:

- “Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++” en la página 15

Consulta relacionada:

- “Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2” en la página 315
- “Tipos de datos de la API C de IBM Sync Client” en la página 319
- “Resumen de funciones de la API C de IBM Sync Client” en la página 317
- “iscConfigSubsSetIsReset()” en la página 336
- “iscConfigSubsSetIsEnabled()” en la página 335

iscConfigEnableSubsSet()

Finalidad: `iscConfigEnableSubsSet()` habilita un conjunto de suscripción de la configuración para su sincronización.

Sintaxis:

```
isy_INT32 iscConfigEnableSubsSet(
    HISCCONF    hConf,
    isy_TCHAR*  id);
```

Argumentos de la función:

La Tabla 156 lista los argumentos válidos utilizados con la función `iscConfigEnableSubsSet()`.

Tabla 156. Argumentos de iscConfigEnableSubsSet()

Tipo de datos	Argumento	Uso	Descripción
HISCCONF	*hConf*	entrada	Conexión de configuración
isy_TCHAR*	*id*	entrada	ID del conjunto de suscripción

Uso:

Inicialmente, todos los conjuntos de suscripción están habilitados para su sincronización. Las funciones `iscConfigEnableSubsSet()` e `iscConfigDisableSubsSet()` habilitan e inhabilitan la capacidad de sincronización de un conjunto de suscripción, que se especifica mediante el ID indicado.

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_NotFound : No se encuentra el conjunto de suscripción.
- ISCRTN_Failed : Otros casos

Restricciones:

Ninguna.

Conceptos afines:

- “La aplicación Sync Client C/C++ de ejemplo” en la página 119

Tareas afines:

- “Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++” en la página 15

Consulta relacionada:

- “Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2” en la página 315
- “Tipos de datos de la API C de IBM Sync Client” en la página 319
- “Resumen de funciones de la API C de IBM Sync Client” en la página 317
- “iscConfigDisableSubsSet()”
- “iscConfigSubsSetIsEnabled()” en la página 335

iscConfigDisableSubsSet()

Finalidad: `iscConfigDisableSubsSet()` inhabilita la sincronización de un conjunto de suscripción.

Sintaxis:

```
isy_INT32 iscConfigDisableSubsSet(
    HISCCONF    hConf,
    isy_TCHAR*  id);
```

Argumentos de la función:

La Tabla 157 lista los argumentos válidos utilizados con la función `iscConfigDisableSubsSet()`.

Tabla 157. Argumentos de iscConfigDisableSubsSet()

Tipo de datos	Argumento	Uso	Descripción
HISCCONF	*hConf*	entrada	Conexión de configuración
isy_TCHAR*	*id*	entrada	ID del conjunto de suscripción

Uso:

Inicialmente, todos los conjuntos de suscripción están habilitados para su sincronización. Las funciones `iscConfigEnableSubsSet()` e `iscConfigDisableSubsSet()` habilitan e inhabilitan la capacidad de sincronización de un conjunto de suscripción, que se especifica mediante el ID indicado.

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_NotFound : No se encuentra el conjunto de suscripción.
- ISCRTN_Failed : Otros casos

Restricciones:

Ninguna.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscConfigEnableSubsSet()" en la página 332
- "iscConfigSubsSetIsEnabled()" en la página 335

iscConfigResetSubsSet()

Finalidad: `iscConfigResetSubsSet()` restablece un conjunto de suscripción de la configuración a la modalidad de restablecimiento.

Sintaxis:

```
isy_INT32 iscConfigResetSubsSet(
    HISCCONF    hConf,
    isy_TCHAR*  id);
```

Argumentos de la función:

La Tabla 158 lista los argumentos válidos utilizados con la función `iscConfigResetSubsSet()`.

Tabla 158. Argumentos de iscConfigResetSubsSet()

Tipo de datos	Argumento	Uso	Descripción
HISCCONF	*hConf*	entrada	Conexión de configuración
isy_TCHAR*	*id*	entrada	ID del conjunto de suscripción

Uso:

Si un conjunto de suscripción está en modalidad de restablecimiento al sincronizarlo, el motor de sincronización elimina los datos del cliente para dicho conjunto de suscripción. El motor de sincronización, simplemente, capta (o vuelve a captar) los datos del servidor; este proceso se denomina renovación. Una vez que se sincroniza un conjunto de suscripción, éste *deja de estar* en modalidad de restablecimiento. Utilice `iscConfigResetSubsSet()` para devolver el conjunto de suscripción especificado a la modalidad de restablecimiento.

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_NotFound : No se encuentra el conjunto de suscripción.
- ISCRTN_Failed : Otros casos

Restricciones:

Ninguna.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscConfigSubsSetIsReset()" en la página 336

iscConfigSubsSetIsEnabled()

Finalidad: `iscConfigSubsSetIsEnabled()` consulta si un conjunto de suscripción está habilitado para su sincronización.

Sintaxis:

```
isy_INT32 iscConfigSubsSetIsEnabled(
    HISCCONF    hConf,
    isy_TCHAR*  id);
```

Argumentos de la función:

La Tabla 159 lista los argumentos válidos utilizados con la función `iscConfigSubsSetIsEnabled()`.

Tabla 159. Argumentos de iscConfigSubsSetIsEnabled()

Tipo de datos	Argumento	Uso	Descripción
HISCCONF	*hConf*	entrada	Conexión de configuración
isy_TCHAR*	*id*	entrada	ID del conjunto de suscripción

Uso:

`iscConfigSubsSetIsEnabled()` se utiliza para realizar una consulta si un conjunto de suscripción, especificado por el ID indicado, tiene habilitada la sincronización. Inicialmente, todos los conjuntos de suscripción están habilitados para su sincronización.

Códigos de retorno:

- ISCRTN_True : El conjunto de suscripción tiene habilitada la sincronización.
- ISCRTN_False : El conjunto de suscripción no tiene habilitada la sincronización.
- ISCRTN_NotFound : No se encuentra el conjunto de suscripción.
- ISCRTN_Failed : Otros casos

Restricciones:

Ninguna.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscConfigSubsSetIsReset()"

iscConfigSubsSetIsReset()

Finalidad: `iscConfigSubsSetIsReset()` realiza una consulta si un conjunto de suscripción está en modalidad de restablecimiento.

Sintaxis:

```
isy_INT32 iscConfigSubsSetIsReset(
    HISCCONF    hConf,
    isy_TCHAR*  id);
```

Argumentos de la función:

La Tabla 160 lista los argumentos válidos utilizados con la función `iscConfigSubsSetIsReset()`.

Tabla 160. Argumentos de iscConfigSubsSetIsReset()

Tipo de datos	Argumento	Uso	Descripción
HISCCONF	*hConf*	entrada	Conexión de configuración
isy_TCHAR*	*id*	entrada	ID del conjunto de suscripción

Uso:

Inicialmente, todos los conjuntos de suscripción están establecidos en modalidad de restablecimiento. Sin embargo, si un conjunto de suscripción se sincroniza, la modalidad del conjunto de suscripción cambia. Utilice `iscConfigResetSubsSet()` para devolver un conjunto de suscripción, especificado por el ID indicado, a la modalidad de restablecimiento.

Códigos de retorno:

- ISCRTN_True : El conjunto de suscripción está en modalidad de restablecimiento.
- ISCRTN_False : El conjunto de suscripción no está en modalidad de restablecimiento.
- ISCRTN_NotFound : No se encuentra el conjunto de suscripción.
- ISCRTN_Failed : Otros casos

Restricciones:

Ninguna.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscConfigSubsSetIsEnabled()" en la página 335

iscConfigGetSubsSetStatus()

Finalidad: `iscConfigGetSubsSetStatus()` obtiene el estado de sincronización de un conjunto de suscripción.

Sintaxis:

```
isy_INT32 iscConfigGetSubsSetStatus(
    HISCCONF    hConf,
    isy_TCHAR*  id);
```

Argumentos de la función:

La Tabla 161 lista los argumentos válidos utilizados con la función `iscConfigGetSubsSetStatus()`.

Tabla 161. Argumentos de iscConfigGetSubsSetStatus()

Tipo de datos	Argumento	Uso	Descripción
HISCCONF	*hConf*	entrada	Conexión de configuración
isy_TCHAR*	*id*	entrada	ID del conjunto de suscripción

Uso:

Utilice `iscConfigGetSubsSetStatus()` para consultar el estado de sincronización de un conjunto de suscripción (que tiene el ID suministrado) durante su última sincronización.

Códigos de retorno:

- ISCRTN_Succeeded : La sincronización del conjunto de suscripción ha resultado satisfactoria.
- ISCRTN_Ready : El conjunto de suscripción está habilitado. El proceso de sincronización ha empezado pero todavía no ha sincronizado el conjunto de suscripción.
- ISCRTN_Canceled : La sincronización del conjunto de suscripción se ha cancelado.
- ISCRTN_Failed : La sincronización del conjunto de suscripción ha fallado.
- ISCRTN_NotFound : No se encuentra el conjunto de suscripción.

Restricciones:

Ninguna.

Conceptos afines:

- “La aplicación Sync Client C/C++ de ejemplo” en la página 119

Tareas afines:

- “Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++” en la página 15

Consulta relacionada:

- “Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2” en la página 315
- “Tipos de datos de la API C de IBM Sync Client” en la página 319
- “Resumen de funciones de la API C de IBM Sync Client” en la página 317
- “iscEngineSync()” en la página 351
- “iscConfigSubsSetIsEnabled()” en la página 335

iscEngineOpen()

Finalidad: `iscEngineOpen()` abre un descriptor de contexto para el motor de sincronización.

Sintaxis:

```
isy_INT32 iscEngineOpen(
    HISCCONF    hConf,
    HISCENG     *phEngine);
```

Argumentos de la función:

La Tabla 162 lista los argumentos válidos utilizados con la función `iscEngineOpen()`.

Tabla 162. Argumentos de iscEngineOpen()

Tipo de datos	Argumento	Uso	Descripción
HISCCONF	*hConf*	entrada	Descriptor de contexto de configuración
HISCENG*	*phEngine*	salida	Descriptor de contexto para el motor de sincronización

Uso:

Utilice `iscEngineOpen()` para abrir un descriptor de contexto para el motor de sincronización (HISCENG) al sincronizar la configuración especificada. El descriptor de contexto se devuelve a través de *phEngine cuando termina satisfactoriamente la sincronización. Si la sincronización no finaliza satisfactoriamente, el valor de *phEngine es NULL y se devuelve un código de error.

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_OutOfMemory : Falta memoria
- ISCRTN_ResourceBusy : Recurso bloqueado (por ejemplo, por otra aplicación)

- ISCRTN_NotPermitted : Recurso no accesible (por ejemplo, el recurso no es legible)
- ISCRTN_NotFound : Recurso no encontrado (por ejemplo, la vía de acceso no se ha encontrado)
- ISCRTN_Failed : Otros casos

Restricciones:

Evite realizar muchas llamadas a `iscEngineOpen()`, puesto que la realización de muchas llamadas abre muchos descriptores de contexto para el motor de sincronización y puede ocasionar problemas de coherencia.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscEngineClose()"

iscEngineClose()

Finalidad: `iscEngineClose()` cierra un descriptor de contexto abierto para el motor de sincronización.

Sintaxis:

```
isy_INT32 iscEngineClose(
    HISCENG        hEngine);
```

Argumentos de la función:

La Tabla 163 lista el argumento válido utilizado con la función `iscEngineClose()`.

Tabla 163. Argumento de iscEngineClose()

Tipo de datos	Argumento	Uso	Descripción
HISCENG	*hEngine*	entrada	Descriptor de contexto para el motor de sincronización

Uso:

Utilice `iscEngineClose()` para cerrar un descriptor de contexto abierto para el motor de sincronización.

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_Failed : Otros casos

Restricciones:

El hecho de realizar varias llamadas a `iscEngineClose()` puede ocasionar errores y se debe evitar.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscEngineOpen()" en la página 338

iscEngineGetInfo()

Finalidad: `iscEngineGetInfo()` obtiene información general sobre el motor de sincronización.

Sintaxis:

```
isy_INT32 iscEngineGetInfo(
    HISCENG             hEngine,
    isy_TCHAR    *info,
    isy_INT32     infoLen);
```

Argumentos de la función:

La Tabla 164 lista los argumentos válidos utilizados con la función `iscEngineGetInfo()`.

Tabla 164. Argumentos de iscEngineGetInfo()

Tipo de datos	Argumento	Uso	Descripción
HISCENG	*hEngine*	entrada	Descriptor de contexto para el motor de sincronización
isy_TCHAR*	*info*	salida	Puntero al almacenamiento intermedio que almacena la información de retorno
isy_INT32	*infoLen*	entrada	Tamaño del almacenamiento intermedio proporcionado

Uso:

`iscEngineGetInfo()` proporciona información del motor de sincronización a efectos de servicio. El contenido y el formato de la información pueden cambiar en el futuro. Por consiguiente, las aplicaciones sólo deben visualizar o registrar esta información. No utilice esta información como entrada para el proceso de programas de aplicación.

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_ValTruncated : La longitud real de la información es superior a infoLen.
- ISCRTN_Failed : Otros casos

Restricciones:

Ninguna.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscGetVersion()" en la página 321

iscEngineSetListener()

Finalidad: `iscEngineSetListener()` registra la función de escucha definida por el usuario con el motor de sincronización. Durante una sesión de sincronización, se llama a la función de escucha cuando se produce un suceso de sincronización (como, por ejemplo, el inicio de la sincronización) o un error.

Sintaxis:

```
isy_INT32 iscEngineSetListener(
    HISCENG             hEngine,
    iscEngineListenerPF syncListener,
    isy_UINT32          syncListenerData);
```

Argumentos de la función:

La Tabla 165 lista los argumentos válidos utilizados con la función `iscEngineSetListener()`.

Tabla 165. Argumentos de iscEngineSetListener()

Tipo de datos	Argumento	Uso	Descripción
HISCENG	*hEngine*	entrada	Descriptor de contexto para el motor de sincronización
iscEngineListenerPF	*syncListener*	entrada	Dirección de la función de escucha definida por el usuario
isy_UINT32	*syncListenerData*	entrada	Datos que la aplicación desea remitir a la función de escucha definida por el usuario

Uso:

Registrando una función de escucha definida por el usuario, la aplicación tiene una vista en el proceso de sincronización. Se notifica a la aplicación cuando se producen sucesos o errores durante la sincronización. La aplicación puede personalizar métodos para presentar estos sucesos o errores a los usuarios.

Ejemplo:

```
// Se define la función syncListener con el prototipo siguiente:
isy_INT32 mySyncListener(
    isy_UINT32 listenerData,
    ISCEVT*     event,
    isy_VOID*   pExtraInfo);...
// El descriptor de contexto para el motor de sincronización se pasa
// a la función de escucha
iscEngineSetListener(hEngine, mySyncListener, (isy_UINT32) hEngine);
```

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_Failed : Otros casos

Restricciones:

La función de escucha definida por el usuario debe seguir el protocolo del motor de sincronización o es posible que éste no funcione correctamente.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscEngineSync()" en la página 351

iscEngineListenerPF

Finalidad: `iscEngineListenerPF` define el prototipo con que debe cumplir la función de escucha definida por el usuario registrada en `iscEngineSetListener()`.

Sintaxis:

```
typedef isy_INT32 (*iscEngineListenerPF)(
    isy_UINT32  listenerData,
    ISCEVT*     event,
    isy_VOID*   pExtraInfo);
```

Argumentos de la función:

La Tabla 166 en la página 343 lista los argumentos válidos utilizados con el tipo de función `iscEngineSetListenerPF`.

Tabla 166. Argumentos de iscEngineListenerPF

Tipo de datos	Argumento	Uso	Descripción
isy_UINT32	*listenerData*	entrada	Los datos establecidos en el argumento *syncListenerData* por *iscEngineSetListener()* se devuelven a la función de escucha
ISCEVT*	*event*	entrada	Objeto de suceso
isy_VOID*	*pExtraInfo*	entrada	Reservado

Uso:

Para utilizar una función de escucha definida por el usuario para supervisar el progreso de la sincronización, antes debe hacer que la función cumpla con el tipo de función `iscEngineSetListenerPF`. A continuación, registre la función de escucha mediante la función `iscEngineSetListener()`. Luego, se notificará a la función de escucha cuando se produzcan sucesos de sincronización. El argumento `event` es una estructura que contiene informaciones diversas sobre el suceso.

La Tabla 167 lista todos los campos de la estructura de sucesos y la finalidad de cada campo.

Tabla 167. Campos de suceso de iscEngineListenerPF

Campo	Descripción
type	El tipo de suceso puede ser uno de los valores siguientes (los valores reales aparecen entre paréntesis): **ISCEVTTYPE_Info (1)** Información relativa al progreso de la sincronización. **ISCEVTTYPE_Conflict (2)** Operaciones conflictivas o rechazadas en el proceso de sincronización. **ISCEVTTYPE_Query (3)** Se necesita cierta información para que la sincronización pueda continuar. La aplicación debe proporcionar cierta información necesaria (basada en el código de suceso) para que el motor de sincronización pueda continuar. **ISCEVTTYPE_Retry (4)** Se produce una excepción y se necesita una instrucción de reintento o cancelación para continuar la sincronización. **ISCEVTTYPE_Error (5)** Se ha producido un error y el motor de sincronización no puede continuar sincronizando el conjunto de suscripción. **ISCEVTTYPE_Fatal (6)** Se ha producido un error muy grave y el motor de sincronización no puede continuar sincronizando conjuntos de suscripción.

Tabla 167. Campos de suceso de iscEngineListenerPF (continuación)

Campo	Descripción
state	Estado del suceso, que contiene los subcampos siguientes: **currSubsSet** Nombre del conjunto de suscripción, si no está vacío. **currSubs** Nombre de la suscripción, si no está vacío. **subsType** Tipo de la suscripción, si es distinto de 0, ordenado así: • 100–999 : Reservado • 1000–9999 : Tipo de suscripción registrado • 10000+ : Tipo de suscripción personalizado Los originadores predefinidos son (los valores reales aparecen entre paréntesis): – ISCSUBSTYPE_Config (100) : Configuración – ISCSUBSTYPE_File (101) : Suscripción de archivo – ISCSUBSTYPE_DB2e (102) : Suscripción de tabla de DB2 Everyplace **syncProg** Progreso de la sincronización expresando en forma de porcentaje.
retry	Número de reintentos del mismo suceso, si es distinto de 0.
info	Información opcional específica del suceso (si no es NULL), que es una matriz de argumentos de serie para los sucesos que no son conflictivos. Para los sucesos conflictivos, el tipo de datos es ISCLISTENCONFLICT.

El campo **event.info** contiene cierta información opcional específica del suceso. El código de suceso se utiliza para identificar e interpretar esta información.

La Tabla 168 lista todos los códigos de suceso por categoría de tipo de suceso.

Tabla 168. Códigos de suceso de iscEngineListenerPF. Tipo de suceso: ISCEVTTYPE_Info

Código de suceso	Inform. suceso (argc)	Descripción
ISCEVT_InfGeneral (1000)	NULL	Información general (para depuración).
ISCEVT_InfSyncStarted (1001)	NULL	Sincronización iniciada.
ISCEVT_InfPrepMsg (1002)	NULL	Preparando mensaje.
ISCEVT_InfSendMsg (1003)	NULL	Enviando mensaje.
ISCEVT_InfWaitMsg (1004)	NULL	En espera de respuesta del servidor.
ISCEVT_InfApplyMsg (1005)	NULL	Aplicando mensaje del servidor.
ISCEVT_InfCancelingSync (1006)	NULL	Cancelando sincronización.
ISCEVT_InfSubsSetStarted (1007)	NULL	Iniciada la sincronización de un conjunto de suscripción.
ISCEVT_InfSyncingSubs (1008)	NULL	Ha comenzado la sincronización de una suscripción.
ISCEVT_InfSubsSetFailed (1009)	NULL	La sincronización de un conjunto de suscripción ha fallado.
ISCEVT_InfSubsSetCanceled (1010)	NULL	Se ha cancelado la sincronización de un conjunto de suscripción.
ISCEVT_InfSubsSetSucceeded (1011)	NULL	La sincronización de un conjunto de suscripción ha finalizado satisfactoriamente.
ISCEVT_InfSyncSucceeded (1012)	NULL	Sincronización satisfactoria.
SCEVT_InfSyncFailed (1013)	NULL	La sincronización (de algunos conjuntos de suscripción) ha fallado.
ISCEVT_InfSyncCanceled (1014)	NULL	Sincronización cancelada (por el usuario).
ISCEVT_InfSyncProg (1015)	NULL	Progreso de la sincronización expresando en forma de porcentaje.

Tabla 168. Códigos de suceso de iscEngineListenerPF (continuación). Tipo de suceso: ISCEVTTYPE_Info

Código de suceso	Inform. suceso (argc)	Descripción
ISCEVT_InfNoNewChange (1016)	NULL	No hay ningún cambio del servidor nuevo; saltarse las fases de extracción y confirmación.
ISCEVT_InfLoginFailed (1017)	NULL	La información de inicio de sesión especificada supera el proceso de autentificación.

Tabla 169. Códigos de suceso de iscEngineListenerPF. Tipo de suceso: ISCEVTTYPE_Conflict

Código de suceso	Inform. suceso (argc)	Descripción
ISCEVT_CftReject (2000)	ISCLISTENCONFLICT	Se han encontrado conflictos de datos en la sincronización. Los datos conflictivos reales se representan en forma de estructura `ISCLISTENCONFLICT` y su puntero de referencia se devuelve a la aplicación a través de *event.info*.

Tabla 170. Códigos de suceso de iscEngineListenerPF. Tipo de suceso: ISCEVTTYPE_Retry

Código de suceso	Inform. suceso (argc)	Descripción
ISCEVT_TryNetConn (4601)	NULL	Volver a intentar conectar con el servidor.
ISCEVT_TrySendRequest (4602)	NULL	Volver a intentar enviar la petición.
ISCEVT_TryRecvReply (4603)	NULL	Volver a intentar recibir la respuesta.
ISCEVT_TryRecvTimeout (4604)	NULL	Esperar más tiempo la respuesta de recepción.
SCEVT_TryRecvAck (4605)	NULL	Volver a intentar recibir un reconocimiento.

Tabla 171. Códigos de suceso de iscEngineListenerPF. Tipo de suceso: ISCEVTTYPE_Query

Código de suceso	Inform. suceso (argc)	Descripción
ISCEVT_QueCancel (5000)	NULL	Consultar si el usuario cancela y vuelve (los valores reales aparecen entre paréntesis): • ISCRTNCB_ReplyYes (3): Si el usuario cancela • ISCRTNCB_ReplyNo (2) : Si el usuario decide continuar • ISCRTNCB_Default (0) : Valor por omisión (es decir, ISCRTNCB_ReplyNo)
ISCEVT_QueCancelUponError (5001)	NULL	Consultar si el usuario cancela y vuelve (los valores reales aparecen entre paréntesis): • ISCRTNCB_ReplyYes (3): Si el usuario cancela • ISCRTNCB_ReplyNo (2) : Si el usuario decide continuar • ISCRTNCB_Default (0) : Valor por omisión (es decir, ISCRTNCB_ReplyNo)
ISCEVT_QueLogin (5002)	ISCLISTENARG(3) info->argv[0] info->argv[1] info->argv[2]	Información de inicio de sesión solicitada por un adaptador. La función de escucha debe proporcionar la información solicitada en event_info y tiene que devolver *ISCRTNCB_Done* con el valor real (1). Nombre de destino de la fuente de datos Almacenamiento intermedio en blanco para contener el nombre de usuario Almacenamiento intermedio en blanco para contener la contraseña
ISCEVT_QueSubsTarget (5003)	ISCLISTENARG(1) info->argv[0]	Información de base de datos solicitada por un adaptador. La función de escucha puede proporcionar la información solicitada en la información del suceso y devolver ISCRTNCB_Done o devolver ISCRTNCB_Default para utilizar el directorio de destino por omisión. Directorio para la suscripción.

Tabla 172. Códigos de suceso de iscEngineListenerPF. Tipo de suceso: ISCEVTTYPE_Error

Código de suceso	Inform. suceso (argc)	Descripción
ISCEVT_ErrOpenAdapter (300)	NULL	No se ha podido abrir el adaptador <*nombre adaptador*>.
ISCEVT_ErrLoadAdapter (301)	NULL	No se ha podido cargar el adaptador <*nombre adaptador*>.
ISCEVT_ErrCloseAdapter (302)	NULL	No se ha podido cerrar el adaptador <*nombre adaptador*>.
ISCEVT_ErrAuthenticateKey (306)	NULL	Ha fallado la autentificación (clave de cifrado no válida); se cancela anormalmente la sincronización.
ISYNCEVT_ErrClientCryptoFailed (307)	NULL	Ha fallado el cifrado o descifrado del cliente; se cancela anormalmente la sincronización.

Tabla 172. Códigos de suceso de iscEngineListenerPF (continuación). Tipo de suceso: ISCEVTTYPE_Error

Código de suceso	Inform. suceso (argc)	Descripción
ISCEVT_ErrEncryptNotAvail (308)	NULL	No se dispone de cifrado.
ISCEVT_ErrEncryptLibOpen (309)	NULL	No se ha podido abrir la biblioteca de cifrado.
ISCEVT_ErrSubsNotFound (311)	NULL	El servidor no ha encontrado la suscripción.
ISCRTN_ErrSubsNotAvail (312)	NULL	El servidor ha bloqueado la suscripción.
ISCRTN_ErrSubsDefAltered (316)	NULL	Se ha modificado la definición de la suscripción desde la última vez que el motor de sincronización sincronizó la configuración.
ISCEVT_ErrAllocResource (400)	NULL	No se han podido asignar recursos de adaptador.
ISCEVT_ErrConnectData (401)	NULL	No se ha podido conectar con los datos de destino.
ISCEVT_ErrDisconnectData (402)	NULL	No se ha podido desconectar de los datos de destino.
ISCEVT_ErrNoData (403)	NULL	No se han encontrado datos.
ISCEVT_ErrMessageFormat (412)	NULL	Formato de mensaje inesperado.
ISCEVT_ErrNotFound (413)	ISCLISTENARG(2) info->argv[0] info->argv[1]	No se han encontrado los datos solicitados. Nombre de destino de la fuente de datos Nombre de los datos
ISCEVT_ErrEndOfData (414)	NULL	Fin de datos inesperado.
ISCEVT_ErrDataTooLong (415)	ISCLISTENARG(3) info->argv[0] info->argv[1] info->argv[2]	Los datos son demasiado largos y se han truncado. Nombre de destino de la fuente de datos Nombre de los datos Nombre del elemento de datos (si no está vacío)
ISCEVT_ErrSyncDisabled (417)	NULL	El servidor ha informado de que el usuario no está habilitado.
ISCEVT_ErrServerException (418)	NULL	El servidor ha informado de excepciones desconocidas.
ISCEVT_ErrReadOnly (420)	ISCLISTENARG(2) info->argv[0] info->argv[1]	Se ha intentado actualizar datos de sólo lectura. Nombre de destino de la fuente de datos Nombre de los datos
ISCEVT_ErrOperation (421)	NULL	Operación no permitida sobre los datos.
ISCEVT_ErrUnauthorized (423)	NULL	No se tiene autorización para acceder a los datos de destino.
ISCEVT_ErrNotAvailable (424)	ISCLISTENARG(2) info->argv[0] info->argv[1]	Los datos solicitados no están disponibles. Nombre de destino de la fuente de datos Nombre de los datos
ISCEVT_ErrNotSupported (425)	ISCLISTENARG(3) info->argv[0] info->argv[1] info->argv[2]	No se da soporte a los datos solicitados. Nombre de destino de la fuente de datos Nombre de los datos Nombre del elemento de datos (si no está vacío)
ISCEVT_ErrNetConn (601)	NULL	No se ha podido conectar con el servidor.
ISCEVT_ErrSendRequest (602)	NULL	No se ha podido enviar la petición.
ISCEVT_ErrRecvReply (603)	NULL	No se ha podido recibir la respuesta.
ISCEVT_ErrRecvTimeout (604)	NULL	Se ha excedido el tiempo de espera mientras se recibía la respuesta.
ISCEVT_ErrRecvAck (605)	NULL	No se ha podido recibir un reconocimiento.
ISCRTN_ErrCloseNetLib (608)	NULL	No se ha podido cerrar la biblioteca de la red
ISCEVT_ErrOutOfMemory (610)	NULL	Falta de memoria.
ISCEVT_ErrInternal (698)	ISCLISTENARG(1) info->argv[0]	Se han producido otros errores internos. Estado del error (en forma de serie).

Tabla 173. Códigos de suceso de iscEngineListenerPF. Tipo de suceso: ISCEVTTYPE_Fatal

Código de suceso	Inform. suceso (argc)	Descripción
ISCEVT_FatSyncCfgAbort (303)	NULL	Ha fallado la sincronización de la configuración; se cancela anormalmente la sincronización.
ISCEVT_FatAuthenticateFailed (304)	NULL	Ha fallado la autentificación; se cancela anormalmente la sincronización.
ISCEVT_FatIncompVersion (310)	NULL	Versión de cliente de sincronización incompatible.
ISCEVT_FatInvalidSession (313)	NULL	ID de sesión no válido.
ISCEVT_FatSyncGroup (314)	NULL	El usuario no pertenece a ningún grupo de sincronización.
ISCEVT_FatRegisterDevice (315)	NULL	No se ha podido registrar el dispositivo para el usuario.
ISCEVT_FatNetOpenConn (600)	NULL	No se ha podido abrir una conexión con el servidor.
ISCEVT_FatOpenNetLib (606)	NULL	No se ha podido cargar la biblioteca de la Red.
ISCEVT_FatResolveHost (609)	NULL	No se ha podido resolver el nombre de host.
ISCEVT_FatServerForbidden (611)	NULL	Está prohibido sincronizar con el servidor.
ISCEVT_FatServerNotFound (612)	NULL	No se ha encontrado el servidor
ISCEVT_FatServer (613)	NULL	Error del servidor.
ISCEVT_FatServerNotAvail (614)	NULL	El servidor no está respondiendo.
ISCEVT_FatNetUnknown (699)	NULL	Error desconocido de la red.

Ejemplo:

```
isy_INT32 mySyncListener(
    isy_UINT32 listenerData,
    ISCEVT*     event,
    isy_VOID*  pExtraInfo)
{
    char *statusMsg = appEventCodeToMessage(event);
    int   timesRetried;

    switch (event->type) {
      case ISCEVTTYPE_Info:
        appStatusBar(statusMsg);
        // appStatusBar puede ser cualquier rutina que muestre el statusMsg
        // (por ejemplo, en una barra de estado)
                return ISCRTNCB_Done;

      case ISCEVTTYPE_Retry:
        timesRetried = event->retry;
        if (timesRetried >= 3) // Intentar como máximo 3 veces
            return ISCRTNCB_ReplyNo;
        else
            return appRetryCancelBox(statusMsg, 10); // tiempo de espera 10 seg
        // appRetryCancelBox puede ser cualquier rutina que muestre una ventana
        // con dos botones: Cancelar y Reintentar. Devuelve
        //   ISCRTNCB_ReplyYes, si el usuario pulsa Reintentar
        //   ISCRTNCB_ReplyNo, si el usuario pulsa Cancelar
        // Si el usuario no hace una selección, devuelve ISCRTNCB_Default.
          break;

        // todos los otros tipos de suceso no importan
      default:
        return ISCRTNCB_Default;
    } // switch (event->type)
} // mySyncListener
```

Códigos de retorno:

- ISCRTNCB_ReplyYes : El usuario responde Sí a la consulta.
- ISCRTNCB_ReplyNo* : El usuario responde No a la consulta.
- ISCRTNCB_Default : No hay respuesta; emprender la acción por omisión.

Si el tipo de suceso es ISCEVTTYPE_Retry, la función de escucha devuelve uno de los códigos siguientes:

Si el tipo de suceso es ISCEVTTYPE_Query, el significado del código de retorno depende del valor del código del suceso. En otras palabras, la función de escucha comprueba el código del suceso y devuelve el valor apropiado. Pero, si el usuario no responde a la consulta, la aplicación devuelve el código siguiente:

- ISCRTNCB_Default : No hay respuesta; emprender la acción por omisión.

Para los tipos de suceso distintos de ISCEVTTYPE_Retry e ISCEVTTYPE_Query, el motor de sincronización ignora el código de retorno. La función de escucha simplemente devuelve ISCRTNCB_Done.

Nota: Para aquellos sucesos que no son de interés, la función de escucha simplemente devuelve ISCRTNCB_Default y permite que el motor de sincronización emprenda la acción por omisión.

Nota: Un asterisco (*) encima indica la acción por omisión para diversos tipos de suceso.

Restricciones:

La función de escucha definida por el usuario debe seguir el protocolo del motor de sincronización. En caso contrario, es posible que el motor de sincronización no funcione correctamente.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscEngineSync()" en la página 351
- "iscEngineSetListener()" en la página 341

iscEngineSetPref()

Finalidad: `iscEngineSetPref()` establece las preferencias del motor de sincronización.

Sintaxis:

```
isy_INT32 iscEngineSetPref(
    HISCENG             hEngine,
    isy_CONST isy_INT32  prefID,
    isy_CONST isy_TCHAR *prefVal);
```

Argumentos de la función:

La Tabla 174 en la página 349 lista los argumentos válidos utilizados con la función `iscEngineSetPref()`.

Tabla 174. Argumentos de iscEngineSetPref()

Tipo de datos	Argumento	Uso	Descripción
HISCENG	*hEngine*	entrada	Descriptor de contexto para el motor de sincronización
isy_CONST isy_INT32	*prefID*	entrada	ID de preferencia, que es uno de los valores siguientes: • ISCPREF_Timeout: Duración del tiempo de espera para recibir mensajes • ISCPREF_Trace: Rastreo detallado.
isy_CONST isy_TCHAR*	*prefVal*	entrada	Nuevo valor de preferencia que se debe establecer. Existen algunas constantes de preferencia predefinidas. Para la preferencia ISCPREF_Trace: • ISCCONST_TraceON: Activar el rastreo de depuración detallado • ISCCONST_TraceOFF: Desactivar el rastreo de depuración detallado Para la preferencia ISCPREF_Timeout: • ISCCONST_TimeoutNever: Nunca se excede del tiempo de espera mientras se espera la respuesta del servidor. • ISCCONST_TimeoutMinimum: Duración mínima del tiempo de espera

Uso:

Utilice `iscEngineSetPref()` para establecer las preferencias del motor de sincronización. Esta preferencias no son persistentes y se deben restablecer cada vez que se abre un nuevo descriptor de contexto para el motor de sincronización.

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_UnknownID: Desconocido
- ISCRTN_ValTooLong: La duración del prefVal indicado es demasiado larga.
- ISCRTN_Failed: Otros errores

Restricciones:

Los valores de preferencia suministrados deben estar dentro de los límites de preferencia especificados:

- ISCPREF_Trace : 1
- ISCPREF_Timeout : 11

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscEngineGetPref()"

iscEngineGetPref()

Finalidad: `iscEngineGetPref()` recupera el valor de preferencia actual.

Sintaxis:

```
isy_INT32 iscEngineGetPref(
    HISCENG             hEngine,
    isy_CONST isy_INT32  prefID,
    isy_TCHAR           *prefVal,
    isy_CONST isy_INT32  prefLen);
```

Argumentos de la función:

La Tabla 175 lista los argumentos válidos utilizados con la función `iscEngineGetPref()`.

Tabla 175. Argumentos de iscEngineGetPref()

Tipo de datos	Argumento	Uso	Descripción
HISCENG	*hEngine*	entrada	Descriptor de contexto para el motor de sincronización
isy_CONST isy_INT32	*prefID*	entrada	ID de preferencia, que es uno de los valores siguientes: • ISCPREF_Timeout: Duración del tiempo de espera para recibir mensajes • ISCPREF_Trace: Rastreo detallado.
isy_TCHAR*	*prefVal*	salida	Puntero al almacenamiento intermedio para almacenar el valor de preferencia devuelto. Existen algunas constantes de preferencia predefinidas. Para la preferencia ISCPREF_Trace: • ISCCONST_TraceON: Activar el rastreo de depuración detallado • ISCCONST_TraceOFF: Desactivar el rastreo de depuración detallado Para la preferencia ISCPREF_Timeout: • ISCCONST_TimeoutNever: Nunca se excede del tiempo de espera mientras se espera la respuesta del servidor. • ISCCONST_TimeoutMinimum: Duración mínima del tiempo de espera
isy_CONST isy_INT32	*prefLen*	entrada	Tamaño del almacenamiento intermedio proporcionado (prefVal).

Uso:

Utilice `iscEngineGetPref()` para obtener el valor de preferencia (que es un valor por omisión o el valor establecido por `iscEngineSetPref()`) de un motor de sincronización.

Códigos de retorno:

- ISCRTN_Succeeded : Bien
- ISCRTN_UnknownID : Se ha proporcionado un prefID desconocido
- ISCRTN_ValTruncated : La longitud real del valor de preferencia es mayor que prefLen.
- SCRTN_Failed : Otros errores

Restricciones:

El almacenamiento intermedio proporcionado debe ser suficientemente grande para almacenar los valores de las diversas preferencias:

- ISCPREF_Trace : 1
- ISCPREF_Timeout : 11
- ISCPREF_CodePage: 15

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscEngineSetPref()" en la página 348

iscEngineSync()

Finalidad: `iscEngineSync()` inicia una sesión de sincronización.

Sintaxis:

```
isy_INT32 iscEngineSync(
    HISCENG        hEngine);
```

Argumentos de la función:

La Tabla 176 lista el argumento válido utilizado con la función `iscEngineSync()`.

Tabla 176. Argumento de iscEngineSync()

Tipo de datos	Argumento	Uso	Descripción
HISCENG	*hEngine*	entrada	Descriptor de contexto para el motor de sincronización

Uso:

Utilice `iscEngineSync()` para iniciar una sesión de sincronización que sincronice la configuración especificada en `iscEngineOpen()`. Un conjunto de suscripción está en modalidad de restablecimiento si nunca se ha sincronizado. Cuando el motor de sincronización realiza una sincronización de dicho conjunto de suscripción, el cliente de sincronización capta los datos del Sync Server; este proceso se denomina renovación. Una vez que finaliza la renovación, el motor de sincronización sincroniza los datos cambiados cuando se vuelve a sincronizar el conjunto de suscripción; este proceso se denomina sincronización. El motor de sincronización siempre sincroniza primero la configuración. Si falla la sincronización de la configuración, el motor de sincronización no sigue procesando los conjuntos de suscripción posteriores y se detiene la sesión de sincronización. Si falla el motor de sincronización en un conjunto de suscripción (pero no en la configuración), el motor de sincronización sigue procesando los conjuntos de suscripción restantes, si los hay.

Códigos de retorno:

- ISCRTN_Succeeded : La sincronización ha finalizado satisfactoriamente.
- ISCRTN_Failed : La sincronización ha fallado.
- ISCRTN_Canceled : Los usuarios han cancelado la sincronización.

El código de retorno de `iscEngineSync()` es el conjunto (que sigue a la prioridad listada a continuación) del estado de sincronización de todos los conjuntos de suscripción que ha sincronizado:

ISCRTN_Canceled > ISCRTN_Failed > ISCRTN_Succeeded

Restricciones:

Ninguna.

Conceptos afines:

- “La aplicación Sync Client C/C++ de ejemplo” en la página 119

Tareas afines:

- “Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++” en la página 15

Consulta relacionada:

- “Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2” en la página 315
- “Tipos de datos de la API C de IBM Sync Client” en la página 319
- “Resumen de funciones de la API C de IBM Sync Client” en la página 317
- “iscConfigPurge()” en la página 328
- “iscEngineSyncConfig()”

iscEngineSyncConfig()

Finalidad: `iscEngineSyncConfig()` inicia una sesión de sincronización que únicamente sincroniza la configuración.

Sintaxis:

```
isy_INT32 iscEngineSyncConfig(
    HISCENG       hEngine);
```

Argumentos de la función:

La Tabla 177 lista el argumento válido utilizado con la función `iscEngineSyncConfig()`.

Tabla 177. Argumento de iscEngineSyncConfig()

Tipo de datos	Argumento	Uso	Descripción
HISCENG	*hEngine*	entrada	Descriptor de contexto para el motor de sincronización.

Uso:

Cuando cambia la configuración en el servidor, `iscEngineSyncConfig()` actualiza la configuración sin volver a sincronizar todos los conjuntos de suscripción.

Códigos de retorno:

- ISCRTN_Succeeded : La sincronización ha finalizado satisfactoriamente.
- ISCRTN_Failed : La sincronización ha fallado.
- ISCRTN_Canceled : Los usuarios han cancelado la sincronización.

Restricciones:

Ninguna.

Conceptos afines:

- "La aplicación Sync Client C/C++ de ejemplo" en la página 119

Tareas afines:

- "Desarrollo de aplicaciones DB2 Everyplace Sync Client utilizando C/C++" en la página 15

Consulta relacionada:

- "Comparaciones entre la API C de IBM Sync Client Versión 8.1 y la Versión 7.2" en la página 315
- "Tipos de datos de la API C de IBM Sync Client" en la página 319
- "Resumen de funciones de la API C de IBM Sync Client" en la página 317
- "iscEngineSync()" en la página 351
- "iscConfigPurge()" en la página 328

Capítulo 19. Tablas base del catálogo del sistema DB2 Everyplace

El gestor de la base de datos crea y mantiene un conjunto de tablas base del catálogo del sistema. Este apéndice contiene una descripción de cada tabla base del catálogo del sistema, incluyendo nombres de columnas y tipos de datos. Todas las tablas base del catálogo del sistema las crea el gestor de la base de datos. Las tablas base del catálogo no se pueden crear ni eliminar explícitamente. Las tablas base del catálogo del sistema se actualizan durante el funcionamiento normal en respuesta a las sentencias SQL de definición de datos, rutinas de entorno y ciertos programas de utilidad. Los datos en las tablas base del catálogo del sistema están disponibles mediante programas de utilidad de búsqueda SQL normales. Las tablas base del catálogo del sistema no se pueden modificar utilizando mandatos de manipulación de datos de SQL normales. Para poder acceder a las tablas de catálogo del sistema es necesario utilizar un identificar delimitado.

Tabla 178. Tablas base del catálogo del sistema

Descripción	Tabla base del catálogo
tablas	355
columnas	355
restricciones referenciales	356
usuarios	356

DB2eSYSTABLES:

La tabla base del catálogo del sistema contiene una fila para cada tabla que se cree. Todas las tablas de catálogo tienen entradas en el catálogo DB2eSYSTABLES.

Tabla 179. DB2eSYSTABLES tabla base del catálogo del sistema

Nombre de la columna	Tipo de datos	Anulable	Descripción
TNAME	VARCHAR (19)		Nombre de tabla
NUMCOLS	INTEGER (4)		Número de columnas
FLAGS	INTEGER (4)		(Sólo uso interno)
NUMKEY	INTEGER (4)		Número de columnas en la clave primaria
CHK	BLOB (512)	Sí	restricción de comprobación (sólo uso interno)
IDXINFO	BLOB (700)	Sí	Índice (sólo uso interno)
NUMREFS	INTEGER (4)	Sí	Clave primaria y foránea (sólo uso interno)
F_ID	INTEGER (4)	Sí	(Sólo uso interno)
PD	BLOB (4096)	Sí	(Sólo uso interno)

DB2eSYSCOLUMNS:

La tabla base del catálogo del sistema contiene una fila para cada columna que se defina para una tabla.

Tabla 180. DB2eSYSCOLUMNS tabla base del catálogo del sistema

Nombre de la columna	Tipo de datos	Anulable	Descripción
CNAME	VARCHAR (19)		Nombre de la columna
TNAME	VARCHAR (19)		Nombre de tabla
TYPE	INTEGER (4)		Tipo de datos
ATTR	INTEGER (4)		(Sólo uso interno)

© Copyright IBM Corp. 1999,2003

Tabla 180. DB2eSYSCOLUMNS tabla base del catálogo del sistema (continuación)

Nombre de la columna	Tipo de datos	Anulable	Descripción
LENGTH	INTEGER (4)		Longitud de la columna
POS	INTEGER (4)		Número de columna
FLAGS	INTEGER (4)		(Sólo uso interno)
KEYSEQ	INTEGER (4)		Posición ordinal de la columna en la clave primaria
SCALE	INTEGER (4)		Escala para columna decimal
DEF	VARCHAR (128)	Sí	Valor por omisión (uso interno)

DB2eSYSRELS:

La tabla base del catálogo del sistema contiene una fila para cada restricción referencial.

Tabla 181. DB2eSYSRELS tabla base del catálogo del sistema

Nombre de la columna	Tipo de datos	Anulable	Descripción
RMD_ID	INTEGER (4)		Clave primaria y foránea (sólo uso interno)
PKTABLE_NAME	VARCHAR (19)		Nombre de tabla padre
PKCOLUMN_NAME	VARCHAR (19)		Columna de clave primaria de la tabla padre
FKTABLE_NAME	VARCHAR (19)		Nombre de tabla hijo
FKCOLUMN_NAME	VARCHAR (19)		Nombre de la columna de clave foránea de la tabla hijo
ORDINAL_POSITION	INTEGER (4)		Posición de la columna en la referencia de la clave foránea

DB2eSYSUSERS:

La tabla DB2eSYSUSERS se crea automáticamente cuando se crea la primera tabla cifrada o cuando se ejecuta la primera sentencia GRANT. Esta tabla está fuertemente vinculada a la base de datos y a los datos cifrados; no se puede mover a otra base de datos DB2 Everyplace que contenga datos cifrados distintos.

Esta tabla base del catálogo del sistema contiene una fila para cada nombre registrado definida para una base de datos.

Tabla 182. DB2eSYSCOLUMNS tabla base del catálogo del sistema

Nombre de la columna	Tipo de datos	Anulable	Descripción
USERNAME	VARCHAR (19)		Parte de la clave primaria, sensible a las mayúsculas/minúsculas. Nombre del usuario asociado a esta fila.
DATABASENAME	VARCHAR (19)		Para usos futuros. Se almacena una serie vacía. Parte de la clave primaria.
TABLENAME	VARCHAR (19)		Para usos futuros. Se almacena una serie vacía. Parte de la clave primaria.
ENCMETHOD	VARCHAR (198)		Para usos futuros. Se almacena una serie vacía. Parte de la clave primaria.
PRIVILEGES	CHAR (19)	Sí	Define privilegios del usuario. Actualmente sólo se admite el valor 'E', que indica cifrado.
ENCKEYDATA	BLOB (64)	Sí	Se utiliza para regenerar una clave cifrada.
ATTIME	TIMESTAMP (26)	Sí	Hora en que se ha añadido el usuario o se ha modificado por última vez el registro, la que sea más reciente de las dos.
VALIDATE	BLOB (64)	Sí	Verifica que el registro es auténtico y que ha añadido el usuario un usuario autentificado.
GRANTOR	VARCHAR (19)	Sí	Nombre del usuario que ha registrado el nombre de usuario de la columna 1.
INTERNALDATA	BLOB (255)	Sí	(Usos futuros internos)

Capítulo 20. Límites de DB2 Everyplace

La tabla siguiente describe ciertos límites de DB2 Everyplace y SQL. Cumplir con el caso más restrictivo puede ayudar al programador a diseñar programas de aplicaciones que son fáciles de transportar. Es posible que se restrinjan aún más muchos de estos límites debido a las limitaciones de sistema y memoria física impuestas por los dispositivos.

Tabla 183. Límites de la base de datos de DB2 Everyplace y de SQL

Descripción	Límite
Tamaño máximo de la tabla (en un sistema de 32 bits)	2 Gigabytes
Longitud máxima de una base de datos	75 bytes
Número máximo de tablas en un almacén de datos	65535
Número máximo de índices en una tabla	15
Número máximo de claves foráneas en una tabla	8
Número máximo de columnas en un índice	8
Número máximo de columnas en una clave primaria	8
Longitud máxima de sentencia de SQL	64 kilobytes
Número máximo de conexiones a una vía de acceso de almacén de datos	1
Número máximo de líneas en una tabla	Limitado por el tamaño de la tabla
Número máximo de columnas en una tabla	256
Longitud máxima para una columna CHAR	32767 bytes
Longitud máxima para una columna VARCHAR o BLOB	32767 bytes
Longitud máxima acumulativa para las columnas de longitud fija de 32767 de una fila	32767 bytes
Número máximo de descriptores de contexto de sentencia por conexión	20
Longitud máxima de restricciones de comprobación	512 bytes
Máximo tamaño de decimales	31 dígitos
Longitud máxima de cada columna en un solo índice	1024 bytes

Consulta relacionada:

- "Visión general del soporte de sentencia de SQL de DB2 Everyplace" en la página 137

© Copyright IBM Corp. 1999,2003

Capítulo 21. Palabras reservadas de DB2 Everyplace

Las siguientes palabras reservadas de DB2 Everyplace no se pueden utilizar como identificadores a menos que estén especificadas como identificadores delimitados. Por ejemplo:

La sentencia siguiente ocasionará un error de SQL:

```
CREATE TABLE tab1 (select int)
```

Utilice comillas dobles y no ocasionará un error de SQL:

```
CREATE TABLE tab1 ("select" int)
```

Palabras reservadas de DB2 Everyplace:

```
ALL, AND, AS, ASC,
BEGIN, BLOB, BY,
DATABASE
CALL, CHAR, CHAR, CHECK, COMMIT, CONCAT, CREATE, CURRENT,
DATE, DECIMAL, DEFAULT, DELETE, DESC, DISTINCT, DROP,
ENCRYPT
FETCH, FOR, FOREIGN, FROM,
GRANT, GROUP,
IN, INDEX, INSERT, INT, INTEGER, INTO, IS,
KEY,
LIKE, LIMIT,
NEW, NOT, NULL,
OF, ON, ONLY, OR, ORDER,
PRIMARY,
READ, REFERENCES, REORG,REVOKE,ROLLBACK,
SELECT, SET, SMALLINT,
TABLE, TIME, TIMESTAMP, TO,TRANSACTION
UPDATE, UPSERT, USING
VALUES, VARCHAR,
WHERE, WITH
```

A efectos de compatibilidades futuras, no utilice como identificadores las palabras reservadas de IBM SQL e ISO/ANSI SQL92 siguientes. Las palabras reservadas de IBM SQL que DB2 Everyplace no utiliza actualmente son las siguientes:

```
ACQUIRE ADD AFTER ALIAS ALLOCATE
ALLOW ALTER ANY ASUTIME AUDIT AUTHORIZATION
AUX AUXILIARY AVG BEFORE BETWEEN BINARY
BUFFERPOOL CALLED CAPTURE CASCADED CASE
CAST CCSID CHARACTER CLOSE CLUSTER COLLECTION
COLLID COLUMN COMMENT CONDITION CONNECT
CONNECTION CONSTRAINT CONTAINS CONTINUE
COUNT COUNT_BIG CROSS CURRENT_DATE CURRENT_LC_PATH
CURRENT_PATH CURRENT_SERVER CURRENT_TIME
CURRENT_TIMESTAMP CURRENT_TIMEZONE CURRENT_USER
CURSOR DATA DATABASE DAY DAYS DBA DBINFO
DBSPACE DB2GENERAL DB2SQL DECLARE DESCRIPTOR
DETERMINISTIC DISALLOW DISCONNECT DO DOUBLE
DSSIZE DYNAMIC EDITPROC ELSE ELSEIF END
END-EXEC ERASE ESCAPE EXCEPT EXCEPTION
EXCLUSIVE EXECUTE EXISTS EXIT EXPLAIN
EXTERNAL FENCED  FIELDPROC FILE FINAL FREE
FULL FUNCTION GENERAL GENERATED GO GOTO
GRANT GRAPHIC HANDLER HAVING HOUR HOURS
IDENTIFIED IF IMMEDIATE INDICATOR INNER
INOUT INSENSITIVE INTEGRITY INTERSECT
ISOBID ISOLATION JAVA JOIN LABEL LANGUAGE
```

© Copyright IBM Corp. 1999,2003

```
LC_CTYPE LEAVE LEFT LINKTYPE LOCAL LOCALE
LOCATOR LOCATORS LOCK LOCKSIZE LONG LOOP
MAX MICROSECOND MICROSECONDS MIN MINUTE
MINUTES MODE MODIFIES MONTH MONTHS NAME
NAMED NHEADER NO NODENAME NODENUMBER NULLS
NUMPARTS OBID OPEN OPTIMIZATION OPTIMIZE
OPTION OUT OUTER PACKAGE PAGE PAGES
PARAMETER PART PARTITION PATH PCTFREE
PCTINDEX PIECESIZE PLAN POSITION PRECISION
PREPARE PRIQTY PRIVATE PRIVILEGES PROCEDURE
PROGRAM PSID PUBLIC QUERYNO READS RECOVERY
RELEASE RENAME REPEAT RESET RESOURCE RESTRICT
RESULT RETURN RETURNS REVOKE RIGHT ROW ROWS
RRN RUN SCHEDULE SCHEMA SCRATCHPAD SECOND
SECONDS SECQTY SECURITY SHARE SIMPLE SOME
SOURCE SPECIFIC SQL STANDARD STATIC STATISTICS
STAY STOGROUP STORES STORPOOL STYLE SUBPAGES
SUBSTRING SUM SYNONYM TABLESPACE THEN TO
TRANSACTION TRIGGER TRIM TYPE UNDO UNION
UNIQUE UNTIL USAGE USER USING VALIDPROC
VARIABLE VARIANT VCAT VIEW VOLUMES WHEN
WHILE WLM WORK WRITE YEAR YEARS
```

Las palabras reservadas de ISO/ANS SQL92 que no son utilizadas por IBM SQL son las siguientes.

```
ABSOLUTE ACTION ARE ASSERTION AT BIT_LENGTH
BOTH CATALOG CHAR_LENGTH CHARACTER_LENGTH
COALESCE COLLATE COLLATION CONSTRAINTS CONVERT
CORRESPONDING DEALLOCATE DEC DEFERRABLE DEFERRED
DESCRIBE DIAGNOSTICS DOMAIN EXEC EXTRACT FALSE
FIRST FLOAT FOUND FULL GET GLOBAL IDENTITY
INITIALLY INPUT INTERVAL LAST LEADING LEVEL
LOWER MATCH MODULE NAMES NATIONAL NATURAL
NCHAR NEXT NULLIF NUMERIC OCTET_LENGTH OUTPUT
OVERLAPS PAD PARTIAL PRESERVE PRIOR REAL
RELATIVE SCROLL SECTION SESSION SESSION_USER
SIZE SPACE SQLCODE SQLERROR SQLSTATE SYSTEM_USER
TEMPORARY TIMEZONE_HOUR TIMEZONE_MINUTE
TRAILING TRANSLATION TRUE UNKNOWN UPPER
VALUE VARYING WHENEVER ZONE
```

Consulta relacionada:

- "Visión general del soporte de sentencia de SQL de DB2 Everyplace" en la página 137

Capítulo 22. Soporte de idioma nacional (NLS)

Este capítulo contiene información sobre el soporte de idioma nacional (NLS) proporcionado por DB2 Everyplace, incluyendo información sobre los países, idiomas y páginas de códigos (juegos de códigos) a los que se da soporte, y sobre cómo configurar y utilizar las características de NLS de DB2 Everyplace con los dispositivos y aplicaciones. DB2 Everyplace da soporte a los juegos de caracteres de un solo byte, de doble byte y de múltiples bytes y a UNICODE. Se soportan tanto UNICODE como otros códigos distintos de UNICODE (ANSI) en todos los sistemas operativos Win32. Lea los temas siguientes para ver una explicación del modo en que se da soporte a las páginas de códigos y a UNICODE.

- "Soporte NLS por sistema operativo de DB2 Everyplace"
- "Codificación de caracteres en aplicaciones Java" en la página 363
- "Habilitadores de idioma de DB2 Everyplace" en la página 363
- "Soporte UNICODE en DB2 Everyplace" en la página 364

Soporte NLS por sistema operativo de DB2 Everyplace

La Tabla 184 muestra los sistemas operativos y los correspondientes idiomas para los cuales existe soporte NLS.

Tabla 184. Soporte NLS

Idioma	Win32	WinCE	Linux	Palm OS	Symbian OS	Neutrino
Inglés	Codepage/ UNICODE	UNICODE	Codepage/ UTF-8	Codepage	UNICODE	UTF-8
Francés	Codepage/ UNICODE	UNICODE	Codepage/ UTF-8	Codepage	UNICODE	UTF-8
Alemán	Codepage/ UNICODE	UNICODE	Codepage/ UTF-8	Codepage	UNICODE	UTF-8
Italiano	Codepage/ UNICODE	UNICODE	Codepage/ UTF-8	Codepage	UNICODE	UTF-8
Español	Codepage/ UNICODE	UNICODE	Codepage/ UTF-8	Codepage	UNICODE	UTF-8
Chino simplificado	Codepage/ UNICODE	UNICODE	Codepage/ UTF-8	Codepage • Instalar habilitador	N/D	UTF-8
Chino tradicional	Codepage/ UNICODE	UNICODE • Instalar habilitador para Pocket PC	Codepage/ UTF-8	Codepage • Instalar habilitador Acer S60 tiene un Palm OS en chino tradicional incorporado.	N/D	UTF-8

© Copyright IBM Corp. 1999,2003

Tabla 184. Soporte NLS (continuación)

Idioma	Win32	WinCE	Linux	Palm OS	Symbian OS	Neutrino
Coreano	Codepage/ UNICODE	UNICODE • Instalar habili-tador	Codepage/ UTF-8	Codepage • Instalar habili-tador	N/D	UTF-8
Japonés	Codepage/ UNICODE	UNICODE	Codepage/ UTF-8	Codepage	N/D	UTF-8
Hebreo	N/D	N/D	N/D	Codepage • Instalar habili-tador	N/D	N/D
Checo	Codepage/ UNICODE	UNICODE • Instalar habili-tador	N/D	Codepage • Instalar habili-tador	UNICODE	N/D
Árabe	N/D	N/D	N/D	Codepage • Instalar habili-tador	N/D	N/D

Para los sistemas operativos Palm OS, QNX Neutrino, Linux, Windows NT y Windows 2000, sin soporte UNICODE, la información sobre el entorno nacional se utiliza para determinar la página de códigos correcta. No existe la conversión interna de series dentro de DB2 Everyplace. Cada serie que se acepta se almacena tal cual. Las aplicaciones de consulta deben utilizar los mismos valores de página de códigos que los que se utilizaron durante el almacenamiento. Esto es similar a cómo DB2 Universal Database proporciona el NLS. DB2 Everyplace no proporciona funciones de conversión de páginas de códigos. Las bases de datos de DB2 Everyplace que se crearon en un sistema utilizando una página de códigos específica sólo se pueden desplegar en sistemas que utilizan la misma página de códigos. Las tablas que se crearon con una página de códigos específica se pueden utilizar en todos los dispositivos que dan soporte a esa página de códigos, excepto cuando sea necesario un habilitador de idioma determinado. Las aplicaciones que acceden a una base de datos deben encargarse de interpretar correctamente los datos de tipo carácter.

DB2 Everyplace detecta la codificación utilizada actualmente examinando el entorno nacional establecido o disponible actualmente.

En Palm OS, también se utiliza la presencia de habilitadores de idiomas para determinar la página de códigos.

Consulta relacionada:

- "Codificación de caracteres en aplicaciones Java" en la página 363
- "Habilitadores de idioma de DB2 Everyplace" en la página 363
- "Soporte UNICODE en DB2 Everyplace" en la página 364

Codificación de caracteres en aplicaciones Java

Los programas Java utilizan internamente texto UNICODE; sin embargo, los datos tipo carácter de una tabla DB2 Everyplace pueden estar en un formato distinto de UNICODE, en función del sistema operativo y del idioma en que se haya creado la tabla. Para sistemas operativos Windows CE y Symbian OS, el controlador JDBC de DB2 Everyplace recupera e inserta texto como de formato UTF-8. Para otros sistemas operativos soportados, se utiliza la codificación de caracteres por omisión del sistema. La codificación por omisión se suele determinar mediante el atributo "file.encoding" de la propiedad del sistema Java.

Por ejemplo, en el sistema operativo Win32, un usuario puede elegir utilizar una versión UNICODE o no UNICODE de la interfaz CLI; por consiguiente, en la misma máquina, una base de datos puede tener codificación en formato UTF-8 y otra tener codificación por código local. Para permitir que una aplicación JDBC acceda a datos de ambas bases de datos, DB2 Everyplace proporciona una manera de que los usuarios indiquen dinámicamente el formato de codificación de datos que una aplicación debe utilizar.

El controlador JDBC de DB2 Everyplace convierte las series Java en bytes según el formato especificado por la aplicación. El formato especificado por la aplicación altera temporalmente la codificación de caracteres por omisión del sistema operativo.

Los usuarios puede especificar dinámicamente el formato de codificación de datos de la aplicación por medio de la interfaz JDBC. Para ello:

1. Cree un objeto java.util.Properties.
   - Clave: DB2e_ENCODING
   - Valor: codificación de caracteres.

   Utilice el valor UTF-8 para especificar DB2 Everyplace utilizando la codificación UTF-8, o utilice cualquier codificación de caracteres soportada por la JVM.
2. Utilice uno de los dos métodos siguientes para pasar el objeto java.util.properties:
   - Para establecer conexión con un URL de base de datos determinado:

     Utilice el método estático Connection getConnection(String url, Properties info) de la clase DriverManager del paquete java.sql.
   - Para crear una conexión de base de datos con un URL determinado:

     Utilice el método Connection connect(String url, Properties info) de la clase Interface Driver del paquete java.sql.

Consulta relacionada:

- "Soporte NLS por sistema operativo de DB2 Everyplace" en la página 361
- "Habilitadores de idioma de DB2 Everyplace"
- "Soporte UNICODE en DB2 Everyplace" en la página 364

Habilitadores de idioma de DB2 Everyplace

Para asegurarse de que el dispositivo portátil utilizado puede visualizar correctamente todos los caracteres del idioma que se está utilizando, puede instalar habilitadores de idioma en el dispositivo portátil. La tabla siguiente lista los habilitadores que puede utilizar con DB2 Everyplace.

Tabla 185. Habilitadores de idioma para dispositivos portátiles

Idioma	Habilitador y sistema operativo
Árabe	Sakhr Arabic Palm 2.0
Chino simplificado	CWP v1.0 para Palm
Chino tradicional	• CJKOS 3.21 para dispositivos de color Palm OS (los registros de clasificación de la opción CJK pueden producir resultados inesperados). • Gismosoft Chinese Small_Knife 2.0 sólo para Pocket PC • Acer S60 tiene un Palm OS en chino tradicional incorporado
Checo	• RedGrep GNU-czech0.71 para Palm OS • Sunnysoft InterWrite5.5P Pro para Windows CE
Hebreo	Penticon Technologies Ltd. Hebrew Support+3.20c para Palm OS
Coreano	• HANME 2.0 para Palm OS • HANTIP 2.01for Palm OS CessHan para Casio E-115 1.0 sobre Windows CE

Consulta relacionada:

- "Codificación de caracteres en aplicaciones Java" en la página 363
- "Soporte NLS por sistema operativo de DB2 Everyplace" en la página 361
- "Soporte UNICODE en DB2 Everyplace"

Soporte UNICODE en DB2 Everyplace

En los sistemas operativos que dan soporte a UNICODE (Windows CE, Symbian OS, Windows NT y Windows 2000), DB2 Everyplace acepta las series UNICODE sólo como series de Entrada/Salida. Estas series UNICODE se guardan en formato UTF–8 dentro del motor de DB2 Everyplace. Un carácter UNICODE puede necesitar de uno a tres bytes de espacio de almacenamiento después de la conversión a UTF–8. Una serie de caracteres almacenada en un servidor de bases de datos tal como IBM DB2 Universal Database, puede necesitar más espacio cuando la cadena se descarga y almacena en DB2 Everyplace en dispositivos Windows CE.

Notas sobre la interfaz UNICODE de CLI:

- Las funciones UNICODE de CLI de DB2 Everyplace tienen el carácter ″W″ al final. Definiendo la macro UNICODE (que es el valor por omisión del sistema en Windows CE), las funciones normales de la CLI se correlacionarán de manera automática con las funciones UNICODE correspondientes. Para escribir código portátil, defina la macro ″UNICODE″ y deje que el sistema realice las conversiones.
- Cuando el soporte UNICODE está habilitado, los tipos de datos SQL_C_CHAR, SQL_C_TCHAR y SQL_C_WCHAR tienen el mismo significado.
- Muchas funciones de la CLI tienen una longitud de serie (o almacenamiento intermedio) como parámetro de entrada/salida.

– Para las funciones que en las que el *Tipo de argumento* es SQLCHAR* (o SQLWCHAR* para la función W), la longitud es el número de caracteres. Por ejemplo:

```
SQLRETURN   SQLExecDirect    (SQLHSTMT          hstmt,
                              SQLCHAR     FAR   *szSqlStr,
                              SQLINTEGER        cbSqlStr);
```

La serie UNICODE `L"ABCD"` tiene cuatro caracteres.

– Para las funciones que en las que el *Tipo de argumento* es SQLPOINTER, la longitud es el número de bytes. Por ejemplo:

```
SQLRETURN  SQLGetData       (SQLHSTMT          hstmt,
                             SQLUSMALLINT      icol,
                             SQLSMALLINT       fCType,
                             SQLPOINTER        rgbValue,
                             SQLINTEGER        cbValueMax,
                             SQLINTEGER  FAR   *pcbValue);
```

Las longitudes del parámetro de entrada `cbValueMax` y del parámetro de salida `*pcbValue` están en bytes. La serie UNICODE `L"ABCD"` ocho 8 bytes.

- Las funciones de UNICODE también pueden hacer que SQL_NTS indique una serie acabada en NULL.

Consejos para escribir código portátil:

- Utilice SQLTCHAR en vez de SQLCHAR o SQLWCHAR.
- Utilice las funciones `_tcs`*XXXX* en vez de `str`*XXXX* (ANSI) o `wcs`*XXXX* (UNICODE). Por ejemplo, `utilice _tcslen()` en vez de `wcslen()` o `strlen()`.
- Utilice `_TEXT()` (o `TEXT()`) para acomodar series literales. Por ejemplo, `_TEXT("ABCD")` se puede interpretar como una cadena de caracteres ANSI o UNICODE dependiendo de la definición de macro.
- Utilice `sizeof(NombreMatriz)/sizeof(TCHAR)` para determinar el tamaño de una matriz de caracteres.

Para obtener un ejemplo, consulte el código de ejemplo SampleCLP de Windows CE incluido con DB2 Everyplace.

Consulta relacionada:

- “Codificación de caracteres en aplicaciones Java” en la página 363
- “Habilitadores de idioma de DB2 Everyplace” en la página 363
- “Soporte NLS por sistema operativo de DB2 Everyplace” en la página 361

Capítulo 23. El conjunto de información de DB2 Everyplace

La biblioteca de DB2 Everyplace consta de la ayuda en línea, en HTML, y de manuales en formato PDF y HTML. Esta sección describe la información proporcionada y cómo acceder a ella.

Toda la información sobre productos puede también consultarse en línea en la dirección www.ibm.com/software/data/db2/everyplace/library.html

Archivos PDF y HTML de DB2 Everyplace

Los manuales de DB2 Everyplace y las Notas del release se pueden visualizar en formatos HTML y PDF, directamente del CD-ROM. La información sobre DB2 Everyplace está traducida a distintos idiomas; pero no toda la información está traducida a cada uno de los idiomas. Cuando una información no está disponible en un idioma determinado, se proporciona en el idioma inglés.

Cuando instale DB2 Everyplace en su estación de trabajo, la documentación se almacenará en `\DB2everyplace\docs`. La tabla siguiente lista los manuales almacenados en el directorio **docs**.

Tabla 186. Manuales disponibles para DB2 Everyplace

Título del manual	Descripción	Nombre del archivo PDF	Directorio de HTML
DB2 Everyplace Guía del usuario y de instalación (SC10–3941–00)	• Instalación de componentes de DB2 Everyplace en una estación de trabajo. • Instalación de la base de datos y las aplicaciones de ejemplo de DB2 Everyplace en un dispositivo portátil o incorporado. • Configuración y mantenimiento de un dispositivo portátil o incorporado. • Utilización de las aplicaciones de ejemplo de DB2 Everyplace.	dsyiug.pdf	dsyiug

© Copyright IBM Corp. 1999,2003

Tabla 186. Manuales disponibles para DB2 Everyplace (continuación)

Título del manual	Descripción	Nombre del archivo PDF	Directorio de HTML
DB2 Everyplace Guía de desarrollo de aplicaciones (SC10–3942–00)	• Creación de aplicaciones de DB2 Everyplace en las plataformas disponibles. • El código fuente y las aplicaciones DB2 Everyplace de ejemplo. • Sentencias de SQL soportadas, Estados de SQL, CLI/ODBC de DB2, métodos de JDBC, API C de IBM Sync Client, API de IBM Java Sync y Soporte de idioma nacional. • Acceso a una base de datos DB2 Everyplace • Utilización del cifrado local de los datos.	dsyadg.pdf	dsyadg
DB2 Everyplace Sync Server Administration Guide (SC18–7186–00)	• Configuración y mantenimiento de Sync Server. • Conexión de Sync Server a fuentes de datos. • Configuración de comunicaciones entre el Sync Server y dispositivos portátiles e incorporados. • Configuración y mantenimiento de bases de datos locales y remotas. • Gestión de usuarios y datos.	dsysag.pdf	dsysag

Documentación en línea sobre DB2 Everyplace

La Ayuda en línea está disponible en el Centro de administración de dispositivos portátiles de DB2 Everyplace Sync Server y en el Mobile Application Builder de DB2 Everyplace.

Parte 5. Apéndices

© Copyright IBM Corp. 1999,2003

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 geográfica. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se puede utilizar dicho producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no infrinja 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
E.E.U.U.

En el caso de consultas sobre licencias referentes a información de doble byte (DBCS), consulte al Departamento de Propiedad Intelectual de IBM en su país o envíe consultas por escrito a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japón

El párrafo siguiente no es aplicable al Reino Unido ni a ningún país en el que tales disposiciones sean incompatibles con la legislación local:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL", SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN DE DERECHOS, COMERCIABILIDAD O IDONEIDAD PARA UN FIN DETERMINADO. Algunos estados no permiten la exclusión de garantías expresas o implícitas en determinadas transacciones, por lo que es posible que esta declaración no sea aplicable en su caso.

Esta publicación puede contener 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/o cambios en los productos y/o 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

© Copyright IBM Corp. 1999,2003

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 esos sitios Web.

Cuando envía información a IBM, IBM puede utilizar o distribuir dicha información en la forma en que IBM 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 este) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
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 manual y todo el material bajo licencia asociado a él, los proporciona IBM según los términos del Convenio del Cliente IBM, el Convenio Internacional de Licencia de Programas de IBM o cualquier convenio equivalente entre el usuario e IBM.

Los datos de rendimiento contenidos en este documento se obtuvieron en un entorno controlado. Por tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse hecho 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 cualquier otra afirmación referente a productos no IBM. Las preguntas sobre las prestaciones de productos no 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.

Esta publicación 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 nombre 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 no intencionada.

LICENCIA DE COPYRIGHT:

Este manual puede contener programas de aplicación de ejemplo escritos en lenguaje fuente, que muestra técnicas de programación en diversas plataformas

operativas. Puede copiar, modificar y distribuir estos programas de ejemplo de la forma que desee, sin pago alguno a IBM, con los fines de desarrollar, utilizar, comercializar o distribuir programas de aplicación 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 tanto, IBM no puede asegurar ni implicar la fiabilidad, utilidad o función de estos programas.

Cada copia o porción 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 derivan de programas de ejemplo de IBM Corp. © Copyright IBM Corp. _especifique el año o años_. Reservados todos los derechos.

Este producto incluye software desarrollado por 3Com y sus colaboradores.:

Copyright (c) 1998 3Com/Palm Computing Division. Reservados todos los derechos. La redistribución y el uso en los formatos fuente y binario, con o sin modificación, están permitidos siempre que se cumplan las condiciones siguientes:

1. Las redistribuciones de código fuente deben conservar la nota de copyright mostrada anteriormente, la presente lista de condiciones y la nota de renuncia que viene a continuación.
2. Las redistribuciones en formato binario deben reproducir la nota de copyright mostrada anteriormente, la presente lista de condiciones y la nota de renuncia que viene a continuación en la documentación y/u otros elementos proporcionados con la distribución.
3. Todos los textos publicitarios que mencionen características o el uso de este software deben mostrar el siguiente reconocimiento: Este producto incluye software desarrollado por 3Com y sus colaboradores.
4. Ni 3Com ni los nombres de sus colaboradores pueden utilizarse para avalar o promocionar productos derivados de este software sin permiso específico previo por escrito.

3COM Y SUS COLABORADORES PROPORCIONAN ESTE SOFTWARE ''TAL CUAL'', SIN GARANTÍAS EXPRESAS NI IMPLÍCITAS, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE COMERCIABILIDAD O IDONEIDAD PARA UN FIN DETERMINADO. EN NINGÚN CASO 3COM NI SUS COLABORADORES SERÁN RESPONSABLES POR NINGÚN DAÑO DIRECTO, INDIRECTO, INCIDENTAL, ESPECIAL, EJEMPLAR NI CONSECUENTE (INCLUIDOS, PERO SIN LIMITARSE A ELLOS, LA OBTENCIÓN DE BIENES O SERVICIOS SUSTITUTOS; LA PÉRDIDA DE USO, DATOS O BENEFICIOS; O LA INTERRUPCIÓN DE LA ACTIVIDAD COMERCIAL) CUALQUIERA QUE SEA LA CAUSA (INCLUIDA LA NEGLIGENCIA), QUE SURJA COMO CONSECUENCIA DEL USO DEL PRESENTE SOFTWARE, INCLUSO SI SE INFORMA DE LA POSIBILIDAD DE TALES DAÑOS.

Marcas registradas

Los términos siguientes, que pueden estar indicados por un asterisco (*), son marcas registradas de International Business Machines Corporation en los Estados Unidos y/o en otros países.

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2
z/OS

Los términos siguientes son marcas registradas de otras empresas:

Microsoft, Windows y Windows NT son marcas registradas de Microsoft Corporation.

Java, y las marcas registradas y logotipos basados en Java y Solaris, son marcas registradas de Sun Microsystems, Inc. en los Estados Unidos y/o en otros países.

Tivoli y NetView son marcas registradas de Tivoli Systems Inc. en los Estados Unidos y/o en otros países.

UNIX es una marca registrada en los Estados Unidos y/o en otros países bajo licencia exclusiva de X/Open Company Limited.

Otros nombres de empresas, productos o servicios, que pueden estar indicados por un doble asterisco (**), pueden ser marcas registradas o marcas de servicio de otras empresas.

Glosario

A

Administrador (MDAC). Interfaz gráfica que permite al usuario crear, editar y visualizar objetos de sincronización y sus relaciones entre sí. El Administrador también permite ver el estado de sincronización de clientes individuales y mensajes de error.

agenda personal electrónica (PDA). Dispositivo de bolsillo que se utiliza para tareas personales de organización (tales como llevar una agenda de actividades o tomar notas) y que incluye servicios de teléfono, fax y conexión a una red.

archivo de anotaciones cronológicas. Objeto de Administrador que contiene mensajes de error de sincronización y sus descripciones.

autenticación. Proceso de validar el ID y la contraseña de un usuario por comparación con las entradas de la base de datos de control de administración para comprobar que el usuario está autorizado para utilizar Sync Server para sincronizar datos.

autorización. En la seguridad de sistemas informáticos, derecho que se otorga a un usuario para comunicarse con un sistema o hacer uso de él.

B

base de datos corporativa. Véase *base de datos fuente*.

base de datos de réplica. Base de datos que el Sync Server utiliza internamente para almacenar los datos necesarios para la sincronización y duplicación.

base de datos destino. Base de datos de DB2 Everyplace, contenida en un dispositivo portátil, en la que se copian datos procedentes de una base de datos fuente.

base de datos fuente. Base de datos que reside en un servidor fuente que contiene los datos que deben copiarse en un sistema destino.

base de datos local. Base de datos que está ubicada físicamente en el sistema que se está utilizando. Compárese con *base de datos remota*.

base de datos maestra. Véase *base de datos fuente*.

base de datos remota. Base de datos que está ubicada físicamente en un sistema distinto del que se está utilizando. Compárese con base de datos local. El sistema remoto puede ser o no portátil.

BLOB. Véase *gran objeto binario*.

C

calificador de Apply. Serie de caracteres que identifica definiciones de suscripción que son exclusivas de cada instancia del programa Apply de DataPropagator.

Centro de control. Interfaz gráfica que muestra objetos de base de datos (tales como bases de datos y tablas) y su relación entre ellos. Desde el Centro de control, se pueden realizar las tareas proporcionadas por las herramientas DBA Utility, Visual Explain y Performance Monitor.

Centro de control de DB2. Véase *Centro de control*.

clave. Columna o colección ordenada de columnas que se identifican en la descripción de una tabla, índice o restricción de referencia.

clave primaria. Clave exclusiva que forma parte de la definición de una tabla. Una clave primaria es la clave padre por omisión de una definición de restricción referencial. En DB2 Everyplace Sync Server Versión 7, cada fuente de duplicación debe tener una sola clave primaria.

cliente. Programa o usuario que se comunica con un servidor de bases de datos y accede a él. Los clientes se definen mediante el Administrador.

conjunto de suscripción. Objeto del Administrador que contiene suscripciones de duplicación. Para que los miembros de un grupo accedan a los datos y archivos definidos en las suscripciones de duplicación, se debe crear un conjunto de suscripción, asignarle suscripciones y luego asignarlo a un grupo. El objeto del conjunto de suscripción sustituye al objeto de la aplicación.

consulta. Petición de información de la base de datos basada en condiciones específicas; por ejemplo, una petición de una lista de todos los clientes de una tabla de clientes cuyo saldo sea superior a 150.000 Pts.

D

DBCS. Véase *juego de caracteres de doble byte*.

DB2 DataPropagator. Producto de duplicación que proporciona un método automático de duplicación de datos fuente en sistemas de destino. Durante la sincronización de datos portátiles, las bases de datos reflejo y remota actúan como fuente y destino de los datos. DataPropagator duplica en la base de datos

© Copyright IBM Corp. 1999,2003

remota los cambios hechos por los sistemas cliente en la base de datos reflejo, y también duplica en la base de datos reflejo los cambios procedentes de la base de datos remota.

detección de conflictos. Proceso de detectar una fila no actualizada en una tabla destino que fue actualizada por una aplicación de usuario. Cuando se detecta un conflicto, se rechaza la transacción que provocó el conflicto.

DHCP. Véase *Protocolo de configuración dinámica de los host*.

dispositivo de bolsillo. Referente a cualquier dispositivo informático que se pueda sostener en la mano. Son ejemplos de tales dispositivos los sistemas PC de bolsillo y las agendas personales electrónicas (personal digital assistants, PDA).

DPROP. Véase *DB2 DataPropagator*.

duplicación. Proceso en el que se toman los cambios almacenados en el archivo de anotaciones o diario de la base de datos, en el servidor fuente, y se aplican en un servidor destino.

F

filtro. Dispositivo o programa que separa datos, señales o información de acuerdo con criterios determinados.

filtro de datos. Véase *filtro.*

fuente de duplicación. Tabla de base de datos que está definida como fuente de la duplicación. Una vez definida una tabla de base de datos como fuente de duplicación, la tabla puede aceptar peticiones de copia.

G

gran objeto binario (binary large object, BLOB). Secuencia de bytes cuyo tamaño está comprendido entre 0 y 2 gigabytes. Esta secuencia de bytes no tiene una página de códigos ni un juego de caracteres asociados. Los objetos de imagen, audio y vídeo se almacenan en forma de objetos BLOB.

grupo. Colección de clientes que tienen necesidades similares respecto a la sincronización de datos portátiles. Para cada grupo se definen características de sincronización, tales como qué aplicaciones necesitan acceder los usuarios del grupo para realizar sus tareas y a qué conjuntos de datos corporativos necesitan acceder.

I

IBM Sync. Nombre del icono que sirve para representar el componente cliente del software de DB2 Everyplace Sync Server.

informática distribuida (pervasive computing). Utilización de una infraestructura informática que incluye dispositivos de información desde los cuales el usuario puede acceder a una amplia gama de servicios a través de una red (incluidos los servicios que generalmente se ofrecen a través de Internet). Estos dispositivos de información pueden ser televisores, automóviles, teléfonos, refrigeradores y hornos microondas. La informática distribuida proporciona un acceso apropiado a la información pertinente y la capacidad para responder de acuerdo con esa información.

J

juego de caracteres de doble byte (double-byte character set, DBCS). Juego de caracteres en el que cada carácter se representa mediante dos bytes.

L

LAN inalámbrica. En las aplicaciones inalámbricas, un usuario de portátiles se puede conectar a una red de área local (LAN) mediante una conexión de radiofrecuencia. La tecnologías inalámbricas para la conexión LAN incluyen espectro de velocidad, microondas y luz infrarroja.

Lenguaje de consulta estructurada (Structured Query Language, SQL). Lenguaje de programación que se utiliza para definir y manejar datos de una base de datos relacional.

LOB. Véase *objeto grande*.

M

MDAC. Véase *Administrador*.

O

objeto.

1. Cualquier elemento que se pueda crear o manipular con SQL; por ejemplo, tablas, vistas, índices o paquetes.
2. En la programación o el diseño orientado a objetos, abstracción que consta de datos y operaciones asociadas a dichos datos.

objeto de sincronización. Elemento gestionable, dentro del Administrador, que contiene información sobre aspectos del proceso de sincronización de datos utilizado. Existen cinco tipos de objetos de

sincronización: grupo, cliente, conjunto de suscripción, suscripción y archivo de anotaciones.

objeto grande (large object, LOB). Secuencia de bytes cuyo tamaño puede ser de hasta 2 gigabytes. Puede ser de uno de estos tres tipos: BLOB (binario), CLOB (caracteres de un solo byte o mixtos) o DBCLOB (caracteres de doble byte).

ODBC. Véase *Open Database Connectivity*.

Open Database Connectivity (ODBC). API que permite acceder a sistemas de gestión de bases de datos utilizando SQL invocable, el cual no necesita la utilización de un preprocesador de SQL. La arquitectura ODBC permite a los usuarios añadir módulos, denominados controladores de bases de datos, que enlazan la aplicación, en tiempo de ejecución, a los sistemas de gestión de bases de datos que hayan elegido. No es necesario enlazar directamente las aplicaciones con los módulos de todos los sistemas de gestión de bases de datos soportados.

P

PDA. Véase *agenda personal electrónica*.

persistente. Relativo a datos que se conservan al pasar de una sesión a otra, generalmente en almacenamiento no volátil, tal como un sistema de base de datos o un directorio.

portátil. Relativo a los procesos de cálculo que se realizan en un sistema portátil o dispositivo de bolsillo por parte de un usuario que se desplaza con frecuencia y utiliza diferentes tipos de conexiones de red (por ejemplo, línea conmutada, LAN o comunicaciones inalámbricas).

privilegio. Derecho a tener acceso a un objeto específico de la base de datos de un modo específico. Estos derechos los controlan los usuarios con autorización SYSADM (administrador del sistema), autorización DBADM (administrador de bases de datos) o los creadores de los objetos. Los privilegios incluyen derechos tales como crear, suprimir y seleccionar datos de las tablas.

Protocolo de configuración dinámica del host (Dynamic Host Configuration Protocol, DHCP). Protocolo de Internet para automatizar la configuración de sistemas que hacen uso de TCP/IP.

pulsar. Utilización de un puntero para interaccionar con un dispositivo de bolsillo.

PVC. Véase *informática distribuida*.

Q

QBE. Véase *Query-by-Example*.

Query-by-Example. Aplicación que permite al usuario visualizar y modificar dinámicamente los datos almacenados en una tabla DB2 Everyplace.

R

RAS. Véase *servicio de acceso remoto*.

renovar. Proceso en el que todos los datos de interés de una tabla de usuario se copian en la tabla destino, sustituyendo los datos existentes.

S

servicio de acceso remoto (Remote Access Service, RAS). Programa de Windows que gestiona las conexiones entre dos sistemas.

servidor corporativo. Véase *servidor fuente*.

servidor de bases de datos. Unidad funcional que proporciona servicios para bases de datos.

servidor fuente. Ubicación de base de datos de la fuente de duplicación.

sesión de sincronización. Transacción en la que los usuarios de portátiles, o *clientes*, someten cambios hechos en copias locales de datos fuente y reciben los cambios realizados en datos fuente (contenidos en el servidor remoto) desde la última vez que se sincronizaron los datos.

sincronización. Véase *sincronización de datos portátiles*.

sincronización de datos. Véase *sincronización de datos portátiles*.

sincronización de datos portátiles. Proceso en dos etapas en el que los usuarios de portátiles, o *clientes*, someten cambios hechos en copias locales de datos fuente y reciben los cambios realizados en datos fuente (contenidos en una base de datos remota) desde la última vez que se sincronizaron los datos.

sistema de gestión de bases de datos (database management system, DBMS). Programa informático de gestión de datos, que proporciona control centralizado de servicios, independencia de los datos y estructuras físicas complejas para conseguir un acceso eficaz, integridad, recuperación, control de concurrencia, privacidad y seguridad.

sistema de nivel medio. Máquina donde está instalado el DB2 Everyplace Sync Server. En una configuración de la sincronización con dos niveles, los términos sistema de nivel medio y sistema fuente designan una misma máquina.

SQL. Véase *Lenguaje de consulta estructurada*.

suscripción. Especificación de cómo debe duplicarse la información de una base de datos fuente en una base de datos destino. Una suscripción le permite definir qué subconjuntos de datos y archivos se pueden copiar desde la base de datos fuente. Puede crear dos tipos de suscripciones: suscripciones para archivos almacenados en el servidor fuente y suscripciones para tablas contenidas en la base de datos fuente.

T

tabla destino. Tabla en la que se copian datos de una tabla fuente. Las tablas reflejo del servidor de nivel medio son destinos, como también lo son las tablas DB2 Everyplace del dispositivo portátil.

tabla fuente. Tabla que contiene los datos que se deben copiar en una tabla destino. La tabla fuente debe ser una tabla fuente de duplicación. Compárese con *tabla destino*.

tabla temporal. Tabla creada durante el proceso de una sentencia de SQL para que contenga resultados intermedios.

U

unión. Operación relacional que permite recuperar datos de dos o más tablas mediante la asociación de valores coincidentes de columnas.

V

vinculación. En SQL, proceso por el cual la salida del precompilador SQL se convierte en una estructura utilizable llamada plan de acceso. Durante este proceso, se seleccionan vías de acceso a los datos y se realiza una cierta comprobación de autorización.

vista. Tabla lógica que consta de datos generados por una consulta.

Índice

A

actualización posicional de columnas por fila 173
API C de IBM Sync Client
  clave para las descripciones de funciones 321
  comparación de versión 315
  resumen 317, 318
  resumen de funciones 317
  tipos de datos 319
API de .NET 307
API de IBM Java Sync, las 19
API de IBM Sync Client, las
  Cliente Java Sync para Cloudscape
    visión general 28
  ISync Client nativo
    visión general 20
  Java ISync Client
    visión general 27
  MIDP ISync Client
    implementación 27
    visión general 27
API de ISync.Net
  ubicaciones de archivo 53
API de ISync.NET
  código de ejemplo 56
API de Java para Cloudscape Sync Client
  visión general 28
API de Java para ISync Client
  implementación
    JNI, en dispositivos Symbian para Nokia 22
    JNI, en Win32 22
    JNI, en Windows CE 23
API de Java para ISync Client nativo
  visión general 20
API de Java para J2ME MIDP ISync Client
  implementación 27
  visión general 27
API de Java para Java Sync Client
  visión general 27
API de JDBC 287
aplicación cliente 138
Aplicación de sincronización de ejemplo de MIDP 125
aplicación de sincronización de ejemplo GoISyncConsole, ejecución 132
aplicación isync4j para MIDP
  desarrollo con la línea de mandatos de Sun Wireless Toolkit 131
  desarrollo con Sun Wireless Toolkit 129
aplicación ISyncSample.java 121
aplicación JSP
  transferir a un dispositivo Windows CE 40
aplicaciones de Cliente de sincronización
  desarrollo utilizando Java 19
aplicaciones de ejemplo
  C/C++ 93
aplicaciones de ejemplo *(continuación)*
  Java 95
    ejecución 105
  JSP 117
  sincronización de Java
    GoISyncConsole 132
  sincronización MIDP de Java 125
  sincronización nativa 121
  Sync Client C/C++ 119
  Visual Basic 111
    visión general 111
aplicaciones JSP
  ejecución 41
  ejecución en un dispositivo Windows CE 43
  ejecución en una estación de trabajo Windows 43
archivos de cabecera 9
asignación de descriptores de contexto 200
atributos, tipo de datos 177
atributos de tipos de datos 177
aumento de la eficiencia, utilizando el objeto PreparedStatement 296
AUTOCOMMIT 154
ayuda en línea 368

B

base de datos
  establecimiento de una conexión 83
base de datos de DB2 Everyplace
  conectarse a 66
biblioteca de sistema operativo 9
bit de modificación
  activar manualmente 175
  concepto de 278
  descripción 278
  errores, al activar 175
  estados 278
  valores, obtener 278, 279
Blob, clase en Java 288
Blob, interfaz 288
BLOB, tipo de datos 145

C

C/C++
  herramientas de desarrollo soportadas 10
caracteres DBCS
  en nombres de columna 145
  en nombres de tabla 144
catálogo 138
CD-ROM, ejecución de DB2 Everyplace desde 68
cifrado
  ejemplo de utilización DB2eCLP 85
  otorgar, instrucciones para sentencia de SQL 156
cifrado *(continuación)*
  visión general 81
clase, DB2eConnection 291
clase, DB2eStatement 304
clase de controlador, en Java 295
Clase de DB2eConnection 291
clases, .NET 307
clases .NET
  soportadas 307
CLI
  utilización para la recuperación gradual de datos 69
CLI de DB2
  diferencias entre estándar y DB2 Everyplace 194
  funciones, lista de 194
  SQLSTATE 181
Cliente de sincronización
  Visión general de la API de Java 19
Cliente Java Sync para Cloudscape
  visión general 28
Cloudscape Sync Client 28
codificación de caracteres 363
columnas
  actualizar valores de fila, sentencia UPDATE 171
  insertar valores, sentencia INSERT 158
comportamiento del cursor 79
condición de búsqueda
  con DELETE, selección de filas 151
  con SELECT, selección de filas 168
  con UPDATE, aplicar cambios 174
Conectar, función 211
conectar con base de datos, utilizando Java 290
conexión
  establecimiento 83
conexión, base de datos 66
conexiones
  comportamiento del cursor en 79
Configuración de CLDC de J2ME 98
configuración de jclCldc, utilización 98
configuración de jclXtr, utilización 98
Configuración personalizada de JCL Extreme Palm 98
confirmación
  comportamiento del cursor 80
conflictos, denominación 66
conflictos de denominación, manejo 66
conflictos de lectura grabación 79
Connection interfaz 290
Consulta Remota 138
controladores de interfaz, registro 288, 295
conversión de datos 76, 285
CREATE INDEX, sentencia 141, 142
CREATE TABLE, sentencia 143
cuentas de bytes 149

© Copyright IBM Corp. 1999,2003

cursor de lectura
comportamiento en los conflictos de grabación 79
cursor desplazable
comportamiento en los conflictos de grabación 79

CH

CHAR, tipo de datos 145
CHARACTER, tipo de datos 145
chips ROM, ejecución de DB2 Everyplace desde 68

D

database_enabler_cldc.jar 99
DatabaseMetaData, interfaz 292
DATE, tipo de datos 145
datos
cifrado
conexión con la base de datos 83
creación de una tabla 84
ejemplo de utilización DB2eCLP 85
gestión de los privilegios de usuario 84
otorgar privilegios de usuario 83
visión general 81
recuperación gradual 69
datos locales
cifrado 81
DB2 Everyplace
conjunto de información 367
límites 357
palabras reservadas 359
DB2 Everyplace, catálogo de 138
DB2eAppl.java
compilación y ejecución en dispositivos no Palm OS 102
compilación y ejecución en Palm OS 97
ejecución en QNX Neutrino o Linux incorporado 108
ejecución en Symbian 108
ejecución en un emulador de Palm OS 100
ejecución en Win32 105
ejecución en Windows CE 106
importación a WSDD para destinos de Palm OS 99
importación a WSDD para destinos no Palm OS 104
no para Palm
añadir db2ejdbc.jar a la vía de acceso de creación 104
creación de un proyecto de WSDD 103
para Palm
añadir el controlador JDBC a la vía de acceso de creación 99
creación de un proyecto de WSDD utilizando la configuración de jclCldc 98
DB2eAppl.java *(continuación)*
para Palm *(continuación)*
creación de un proyecto de WSDD utilizando la configuración de jclXtr 98
DB2eCLP
cifrado utilizando 85
DB2eCommand 308
DB2eCommandBuilder 307
DB2eConnection 309
DB2eDataAdapter 309
DB2eDataReader 310
DB2eError 312
DB2eException 312
DB2eJDBC_Cldc_maps.jar 99
DB2eParameter 312
DB2ePLANTABLE
columnas 155
utilizando la sentencia EXPLAIN 155
DB2eStatement, clase 304
DB2eSYSCOLUMNS 355
DB2eSYSRELS 356
DB2eSYSTABLES 355
DB2eSYSUSERS 356
DB2eType 314
DECIMAL, tipo de datos 145
definición del preprocesador 12
DELETE, estado del bit de modificación 278
DELETE, sentencia 150
errores al ejecutar 153
fila múltiple 153
supresión lógica de registros 153
desarrollo de aplicaciones DB2 Everyplace
para el Sync Client 19, 20, 26
registro del ID de creador de la aplicación 10
utilización de .NET
enumeración de DB2eType 314
Miembros de DB2eCommand 308
miembros de DB2eCommandBuilder 307
Miembros de DB2eConnection 309
Miembros de DB2eDataAdapter 309
Miembros de DB2eDataReader 310
Miembros de DB2eError 312
Miembros de DB2eException 312
Miembros de DB2eParameter 312
utilizando C/C++
aplicación de ejemplo 93
archivos de biblioteca necesarios 12
archivos de cabecera 11
archivos necesarios para prueba 13, 14
compilación de ejemplos 11
definición del preprocesador 12
herramientas de desarrollo soportadas 10
para el Cliente de sincronización 15
pila de ejecución de programas, tamaño para Palm OS 12
desarrollo de aplicaciones DB2 Everyplace *(continuación)*
utilizando C/C++ *(continuación)*
preparación, compilación y enlace de proyectos 11
probar aplicación 13
sistemas operativos soportados 11
soporte para UNICODE 12
visión general 9
utilizando Java 17
aplicaciones de ejemplo 95
aplicaciones de ejemplo, ejecución 105
interfaces en el paquete java.sql 287
interfaces en el paquete javax.sql 305
programas de ejemplo 97, 102
sistemas operativos soportados 17
visión general 17
utilizando JavaServer Pages
aplicaciones de ejemplo 117, 121
configurando el mini servidor de Web HTTP en Win32 41
ejecución de aplicaciones JSP 41
ejecución de una aplicación en un dispositivo Windows CE 43
ejecución de una aplicación en una estación de trabajo Windows 43
probar 33
sistemas operativos soportados 33
subconjuntos JSP 1.1 soportados 45
utilizando códigos de personalización de IBM 48
visión general 33
utilizando Visual Basic
aplicaciones de ejemplo 111
aplicaciones de ejemplo, visión general 111
pasos básicos 31
prueba del programa de ejemplo 114
sistemas operativos soportados 32
SQLAllocHandle, función 202
visión general 31
desarrollo de aplicaciones de Cliente de sincronización de DB2 Everyplace
utilizando Java 19
Desconectar, función 222
Describir atributos de columna, función 220
descriptor de conexión
asignación 200
ficticio 201
liberar 242
descriptor de contexto, liberar 242
descriptor de contexto de descriptor
asignación 200
descriptor de entorno
asignación 200
liberar 242
descriptor de sentencia
asignación 200

descriptor de sentencia *(continuación)*
descriptor 202
liberar 242
múltiple 201
diagnósticos, obtener varios campos 254
dispositivo portátil
utilización de habilitadores de idioma 363
Driver, interfaz 295
DROP, sentencia
errores al ejecutar 154
finalidad 153

E

eficiencia, aumentar utilizando el objeto PreparedStatement 296
Ejecutar sentencia, función 227
ejecutar sentencia de SQL 303
Ejecutar sentencia directamente, función 225
Enlazar columna, función 203
Enlazar un almacenamiento intermedio con un marcador de parámetro, función 207
entorno de ejecución J9
instalación en un dispositivo Windows CE 37
errores
al ejecutar sentencia DROP 154
al ejecutar sentencias DELETE 153
al ejecutar sentencias UPDATE 175
Establecer opciones de conexión, función 272
Establecer opciones de sentencia, función 276
executeUpdate(String sql), método 95
EXPLAIN, sistemas operativos soportados para la sentencia 154

F

FAR, punteros 198
fila
actualizar valores de columna, sentencia UPDATE 171
insertar en una tabla 157
insertar valores, sentencia INSERT 158
restricciones para insertar valores 159
suprimir, reglas para sentencia de SQL 150
FROM, cláusula en sentencia DELETE 151
función de sincronización
iscConfigClose() 327
iscConfigCloseCursor() 330
iscConfigDisableSubsSet() 333
iscConfigEnableSubsSet() 332
iscConfigGetNextSubsSet() 331
iscConfigGetSubsSetStatus() 337
iscConfigOpen() 326
iscConfigOpenCursor() 328
iscConfigPurge() 328
iscConfigResetSubsSet() 334
función de sincronización *(continuación)*
iscConfigSubsSetIsEnabled() 335
iscConfigSubsSetIsReset() 336
iscEngineClose() 339
iscEngineGetInfo() 340
iscEngineGetPref() 350
iscEngineListenerPF 342
iscEngineOpen() 338
iscEngineSetListener() 341
iscEngineSetPref() 348
iscEngineSync() 351
iscEngineSyncConfig() 352
iscGetVersion() 321
iscServiceClose() 325
iscServiceOpen() 322
iscServiceOpenEx() 324
función en desuso
SQLAllocConnect 199
SQLAllocEnv 200
SQLAllocStmt 203
SQLError 225
SQLFreeConnect 241
SQLFreeEnv 242
SQLFreeStmt 244
funciones de CLI para DB2, por categoría 194

G

GRANT, sistemas operativos soportados para la sentencia 156

H

habilitadores 363
herramientas de desarrollo de aplicaciones
para EPOC R5 10
para Linux y Linux incorporado 10
para Palm OS 10
para QNX Neutrino 10
para Symbian OS Versión 6.0 10
para Symbian OS Versión 7.0 10
para Windows 2000 10
para Windows CE 10
para Windows NT 10
HISCCONF, tipo de datos 319
HISCCSR, tipo de datos 319
HISCENG, tipo de datos 319
HISCSERV, tipo de datos 319

I

IBDB 10
identificadores delimitados
uso para nombres de columna 144
uso para nombres de tabla 144
idioma, habilitadores de 363
INDEX, cláusula en sentencia DROP 153
índice
crear, bit de modificación 142
crear, instrucciones para sentencia de SQL 141
duplicado, descripción 142
exploración bidireccional 142
exploración por prefijo 142
índice *(continuación)*
limitaciones al crear 142
ordenación 142
supresión, mediante la sentencia DROP 153
INSERT, estado del bit de modificación 278
INSERT, restricciones de la cláusula que provocan error 159
INSERT, sentencia 157
INTEGER, tipo de datos 145
interfaz, CallableStatement 288
interfaz, DataSource 306
interfaz Blob 288
interfaz CallableStatement 288
interfaz Connection 290
interfaz DatabaseMetaData 292
Interfaz DataSource 306
interfaz de CLI/ODBC 9, 31, 33
interfaz de JDBC.*Ver también* desarrollo de aplicaciones DB2 Everyplace, utilizando Java 17
interfaz Driver 295
interfaz PreparedStatement 296
interfaz ResultSet 297
interfaz ResultSetMetaData 301
interfaz Statement 303
INTO, cláusula
INSERT, especificación de la tabla en la sentencia 158
restricciones de utilización 158
iscConfigClose(), función de sincronización 327
iscConfigCloseCursor(), función de sincronización 330
iscConfigDisableSubsSet(), función de sincronización 333
iscConfigEnableSubsSet(), función de sincronización 332
iscConfigGetNextSubsSet(), función de sincronización 331
iscConfigGetSubsSetStatus(), función de sincronización 337
iscConfigOpen() , función de sincronización 326
iscConfigOpenCursor(), función de sincronización 328
iscConfigPurge(), función de sincronización 328
iscConfigResetSubsSet(), función de sincronización 334
iscConfigSubsSetIsEnabled(), función de sincronización 335
iscConfigSubsSetIsReset(), función de sincronización 336
iscEngineClose(), función de sincronización 339
iscEngineGetInfo(), función de sincronización 340
iscEngineGetPref(), función de sincronización 350
iscEngineListenerPF, función de sincronización 342
iscEngineOpen(), función de sincronización 338
iscEngineSetListener(), función de sincronización 341

iscEngineSetPref(), función de sincronización 348
iscEngineSync(), función de sincronización 351
iscEngineSyncConfig(), función de sincronización 352
ISCEVT, tipo de datos 319
iscGetVersion(), función de sincronización 321
ISCLISTENARG, tipo de datos 320
ISCLISTENCOLUMN, tipo de datos 320
ISCLISTENCONFLICT, tipo de datos 320
iscServiceClose(), función de sincronización 325
iscServiceOpen(), función de sincronización 322
iscServiceOpenEx(), función de sincronización 324
ISCSTATE, tipo de datos 319
isy_BOOL, tipo de datos 319
isy_BYTE, tipo de datos 319
isy_DWORD, tipo de datos 319
isy_INT, tipo de datos 319
isy_INT16, tipo de datos 319
isy_INT32, tipo de datos 319
isy_TCHAR, tipo de datos 319
isy_UINT, tipo de datos 319
isy_UINT16, tipo de datos 319
isy_UINT32, tipo de datos 319
isy_ULONG, tipo de datos 319
isy_VOID, tipo de datos 319
isy_WORD, tipo de datos 319
ISync Client nativo
  visión general 20
isync4j 19, 20, 26
isync4j para PalmOS 24

J

J2ME MIDP ISync Client
  implementación 27
  visión general 27
Java, aplicaciones
  utilización de UNICODE 363
Java, métodos 287
Java ISync Client
  visión general 27
java.sql 99
java.sql, paquete 95
  interfaces soportadas 287
JavaServer Pages, aplicaciones *Ver* desarrollo de aplicaciones DB2 Everyplace, JavaServer Pages 33
JDBC
  sistemas operativos soportados 17
JSP
  códigos de personalización de IBM 48
JSP *Ver* desarrollo de aplicaciones DB2 Everyplace, JavaServer Pages 33

L

Liberar recursos de descriptor de contexto, función 242
limitaciones en las aplicaciones de ejemplo Visual Basic 111
límites 357
Línea de mandatos de Sun Wireless Toolkit 131
Linux
  uso con C/C++ 11
  uso con Java 17
  uso con la sentencia EXPLAIN 154
Linux incorporado. *Ver* Linux 11

M

manuales 367
marcadores de parámetro
  ejemplo de ADO.NET 74
  ejemplo de CLI 72
  ejemplo de JDBC 74
  visión general 71
marcadores de parámetros
  no tipificados 76
  restricciones 76, 77
mensaje de límite excedido de SQL o del producto, en SQLState 181
mensajes de aviso, en SQLState 180
mensajes de característica no soportada, en SQLState 180
mensajes de DDL para Java, en SQLState 181
mensajes de diversos errores de SQL o del producto, en SQLState 181
mensajes de error
  CLI 180
  SQL 180
mensajes de error de sintaxis o de violación de regla de acceso, en SQLState 181
mensajes de error de SQL dinámico, en SQLState 180
mensajes de error del Administrador de DB2 Everyplace, en SQLState 181
mensajes de especificación no válida de autorización, en SQLState 181
mensajes de estado no válido de aplicación, en SQLState 181
mensajes de estado no válido de la transacción, en SQLState 181
mensajes de estado no válido del cursor, en SQLState 181
mensajes de excepción de acción desencadenada, en SQLState 180
mensajes de excepción de conexión, en SQLState 180
mensajes de excepción de datos, en SQLState 181
mensajes de excepción de función externa, en SQLState 181
mensajes de excepción de llamada a función externa, en SQLState 181
mensajes de identificador no válido de sentencia de SQL, en SQLState 181
mensajes de no hay datos, en SQLState 180
mensajes de nombre no válido de conexión, en SQLState 181
mensajes de nombre no válido de cursor, en SQLState 181
mensajes de objeto no encontrado en estado previo necesario, en SQLState 181
mensajes de recurso erróneo del sistema, en SQLState 181
mensajes de recurso no disponible o intervención del operador, en SQLState 181
mensajes de retrotracción de transacción, en SQLState 181
mensajes de símbolo no válido, en SQLState 180
mensajes de SQLState
  CLI 185
  códigos de clase 180
  JDBC 193
mensajes de terminación no válida de transacción, en SQLState 181
mensajes de terminación satisfactoria no calificada, en SQLState 180
mensajes de violación de la cardinalidad, en SQLState 180
mensajes de violación de opción con comprobación, en SQLState 181
mensajes de violación de restricción, en SQLState 181
mensajes en SQLState 180, 181
método Java
  clase, DB2eConnecton 291
  clase, DB2eStatement 304
  interfaz, CallableStatement 288
  interfaz, DataSource 306
  interfaz Blob 288
  interfaz Connection 290
  interfaz DatabaseMetaData 292
  interfaz Driver 295
  interfaz PreparedStatement 296
  interfaz ResultSet 297
  interfaz ResultSetMetaData 301
  interfaz Statement 303
métodos JDBC
  soportados 287
Metrowerks CodeWarrior 10
Microsoft eMbedded Visual Tools 10
MIDP ISync Client
  implementación 27
  visión general 27
mini servidor de Web HTTP 34
mini servidor de Web HTTP, configuración para JavaServer Pages en Win32 41
MiniHttpConfig.properties, ejemplo de archivo para JavaServer Pages para Win32 41
modalidad de confirmación automática
  comportamiento del cursor 80

N

next(), método 95
nombre-columna, en sentencia CREATE TABLE 144
nombre-tabla, en sentencia CREATE TABLE 144
Número de columnas resultantes, función 265

O

Obtener columnas de clave foránea, función 238
Obtener columnas de clave primaria, función 268
Obtener datos, función 250
Obtener el valor actual de un atributo de conexión, función 246
Obtener el valor de una sentencia, función 260
Obtener información, función 257
obtener información, para sentencia SELECT 154
Obtener información de columna para una tabla, función 216
Obtener información de tabla, función 283
Obtener nombre de cursor, función 248
Obtener número de columnas resultantes, función 265
Obtener número de filas, función 271
Obtener número de parámetros en una sentencia SQL, función 264
Obtener varios campos del registro de diagnóstico, función 254
On Error Resume Next, sentencia 31
opciones de columna, en sentencia CREATE TABLE 146

P

palabras reservadas 359
Palm OS
  uso con C/C++ 11
  uso con Java 17
  uso con la sentencia GRANT 156
paquete javax.sql
  interfaces soportadas 305
paquete JDBC 95
parámetros, vincular 76
PDF 367
pila de ejecución de programas, tamaño para Palm OS 12
Preparar sentencia, función 266
PreparedStatement, interfaz 296
privilegios
  usuario
    gestionar bases de datos cifradas 84
    otorgar para bases de datos cifradas 83
privilegios de cifrado
  gestionar 84
  otorgar 83
privilegios de usuario
  gestionar bases de datos cifradas 84
  Otorgar para bases de datos cifradas 83
procedimiento almacenado
  llamar, instrucciones para sentencia de SQL 138
procesador JSP 34
procesador MIPS 12
procesador SH3 12
procesador SH4 12
programas de ejemplo
  Java 97, 102
  sentencia CALL 139
propiedades de columna, búsqueda en objeto ResultSet 301
proveedor de datos .NET
  utilización 57
  visión general 57
proveedor de sincronización basado en JNI
  instalación 23
proveedor de sincronización nativa basada en JNI, instalación 21
proveedor de sincronización nativa basada en la detección
  instalación 24
proveedores de sincronización
  visión general 19
proveedores de sincronización de Java 26
proveedores de sincronización nativa
  instalación 20
punteros FAR 198

Q

QNX Neutrino
  uso con C/C++ 11
  uso con Java 17
  uso con Metrowerks CodeWarrior 10

R

recuperación gradual de datos 69
Recuperar, función 229
Recuperar conjunto de filas y Devolver datos, función 231
recursos, liberar 201
registro de controladores de interfaz 288, 295
REORG TABLE, sentencia
  finalidad 160
  invocar internamente 160
requisitos de hardware 4
restricciones referenciales
  en sentencia CREATE TABLE 147
ResultSet, interfaz 297
ResultSetMetaData, interfaz 301
retrotracción
  comportamiento del cursor 80

S

seguridad 81
selección de la vía de acceso
  script de ejemplo 155
  utilizando la sentencia EXPLAIN 154
SELECT, sentencia 162
sentencia CALL 138
sentencia de SQL
  CALL 137, 138
  CREATE INDEX 137, 141
  CREATE TABLE 137, 143
  DELETE 137, 150
  DROP 137, 153
  ejecutar 303
sentencia de SQL *(continuación)*
  estática 303
  EXPLAIN
    DB2ePLANTABLE, columnas de la tabla 155
    DB2ePLANTABLE, creación de la tabla 155
    finalidad 154
    lista 137
  GRANT 156
  INSERT
    finalidad 157
    lista 137
    restricciones 160
  limitación de longitud 138
  precompilada 296
  preparada 296
  REORG TABLE
    consideraciones 160
    finalidad 160
    invocar internamente 160
    lista 137
  REVOKE 161
  SELECT 137, 162
  SQLExecute, función 138
  SQLPrepare, función 138
  UPDATE 137, 171
  visión general 137
serialización, conexiones 67
serialización de conexiones 67
SET, cláusula de sentencia UPDATE 173
sitio Web de DB2 Everyplace 10
SMALLINT, tipo de datos 145
Software Developer's Kit de GNU 10
Software Developer's Kit de Java 17, 287
soporte de idioma
  codificación de caracteres en aplicaciones Java 363
  por sistema operativo 361
  UNICODE 364
  utilización de habilitadores de idioma 363
  visión general 361
soporte de idioma nacional
  codificación de caracteres en aplicaciones Java 363
  por sistema operativo 361
  UNICODE 364
  utilización de habilitadores de idioma 363
  visión general 361
soporte de JSP 34
  configuración en un dispositivo Windows CE
    visión general 36
  verificación en una estación de trabajo Windows 36
soporte de la sentencia de SQL 137
soporte de sólo lectura, ejecución de DB2 Everyplace desde 68
Soporte NLS
  codificación de caracteres en aplicaciones Java 363
  por sistema operativo 361
  UNICODE 364
  utilización de habilitadores de idioma 363

Soporte NLS *(continuación)*
visión general 361
soporte para UNICODE 12
SQL
límites 357
SQL, tipo de datos 194
SQL, tipos de datos
atributos 177
simbólicos y por omisión 177
SQLAllocConnect, función en desuso 199
SQLAllocEnv, función en desuso 200
SQLAllocStmt, función en desuso 203
SQLAllocHandle, función 200
SQLAllocHandleVer, función interna 31
SQLBindCol, función 203
SQLBindParameter, función 76, 207
SQLColumns, función 216
SQLConnect, función 211
SQLDescribeCol, función 220
SQLDisconnect, función 222
SQLEndTran, función 223
SQLError, función en desuso 225
SQLExecDirect, función 76, 225
SQLExecute, función 76, 227
SQLFetch, función 229
SQLFetchScroll, función 231
SQLForeignKeys, función 238
SQLFreeConnect, función en desuso 241
SQLFreeEnv, función en desuso 242
SQLFreeHandle, función 242
SQLFreeStmt, función en desuso 244
SQLGetConnectAttr, función 246
SQLGetCursorName, función
descripción 248
SQLGetData, función 250
SQLGetDiagRec, función 254
SQLGetInfo, función 257
SQLGetStmtAttr, función 260
SQLNumParams, función 264
SQLNumResultCols, función 265
SQLPrepare, función 266
SQLPrimaryKeys, función 268
SQLRowCount, función 271
SQLSetConnectAttr, función 272
SQLSetStmtAttr, función 276
SQLSTATE 138, 180
SQLTables, función 283
Statement, interfaz 303
subconjuntos de JSP Versión 1.1, soportados 45
Sun Wireless Toolkit 129
supresión de objetos SQL 153
Symbian
implementaciones basadas en JNI 22
Symbian OS
uso con C/C++ 11
Symbian OS/EPOC
uso con la sentencia GRANT 156
Sync Client
aplicaciones de ejemplo
C/C++ 119

T

tabla
actualización por fila y columna, sentencia UPDATE 171
compresión
con sentencia de SQL 160
invocar internamente 160
creación de cifrado 84
crear, en base de datos corporativa 149
crear, instrucciones para sentencia de SQL 143
inserción de filas
con sentencia de SQL 157
supresión, mediante la sentencia DROP 153
tablas
base del catálogo del sistema, descripción 355
límites para DB2 Everyplace 357
visión general de DB2 Everyplace 65
tablas base del catálogo del sistema, descripción 355
tablas definidas por el usuario
manejo de conflictos de denominación 66
TABLE, cláusula en sentencia DROP 153
TIME, tipo de datos 145
TIMESTAMP, tipo de datos 145
tipo de datos
BLOB 145
compatibilidad 176
compatibles 176
conversiones 76, 194
CHAR 145
DATE 145
DECIMAL 145
en lenguaje C 194
HISCCONF 319
HISCCSR 319
HISCENG 319
HISCSERV 319
INT 145
INTEGER 145
ISCEVT 319
ISCLISTENARG 320
ISCLISTENCOLUMN 320
ISCLISTENCONFLICT 320
ISCSTATE 319
isy_BOOL 319
isy_BYTE 319
isy_DWORD 319
isy_INT 319
isy_INT16 319
isy_INT32 319
isy_TCHAR 319
isy_UINT 319
isy_UINT16 319
isy_UINT32 319
isy_ULONG 319
isy_VOID 319
isy_WORD 319
operandos, de 176
para las API C de IBM Sync Client 319
SMALLINT 145
SQL 194
tipo de datos *(continuación)*
TIME 145
TIMESTAMP 145
VARCHAR 145
tipos de avisos 180
tipos de columna, búsqueda en objeto ResultSet 301
tipos de datos por omisión y simbólicos, SQL 177

U

UID de aplicación
para EPOC R5 10
para Palm OS 10
para Symbian OS Versión 6.0 10
para Symbian OS Versión 7.0 10
UNICODE
soporte en DB2 Everyplace 364
utilización en aplicaciones Java 363
UPDATE, estado del bit de modificación 278
UPDATE, sentencia
finalidad 171

V

VALUES, cláusula
INSERT, carga de una fila en sentencia 158
número de valores, reglas para 158
VARCHAR, tipo de datos 145
variable de lenguaje principal, insertar en filas 158
vinculación de parámetros 76

W

WCE Tooling
instalación para destinos no Palm 103
instalación para destinos Palm 97
WHERE, cláusula
DELETE, selección de filas para la sentencia 151
SELECT, selección de filas para la sentencia 168
UPDATE, búsqueda condicional en sentencia 174
Windows 2000
implementaciones basadas en JNI 22
uso con C/C++ 11
uso con Java 17
uso con JavaServer Pages 33
uso con la sentencia EXPLAIN 154
uso con Visual Basic 32
Windows CE
implementaciones basadas en JNI 23
uso con C/C++ 11
uso con Java 17
uso con JavaServer Pages 33
uso con la sentencia GRANT 156
uso con Visual Basic 32
Windows NT
implementaciones basadas en JNI 22
uso con C/C++ 11

Windows NT *(continuación)*
uso con Java 17
uso con JavaServer Pages 33
uso con la sentencia EXPLAIN 154
uso con la sentencia GRANT 156
uso con Visual Basic 32
WSDD
creación de un proyecto para DB2eAppl.java para destinos de Palm 98
creación de un proyecto para DB2eAppl.java para destinos no Palm 103
importación de DB2eAppl.java para destinos no Palm 104
importación de DB2eAppl.java para destinos Palm 99
instalación de WCE Tooling para destinos no Palm 103
instalación de WCE Tooling para destinos Palm 97

Cómo ponerse en contacto con IBM

Para obtener información o para solicitar cualquiera de los productos de DB2 Everyplace, consulte a un representante de IBM en una sucursal local o a un distribuidor autorizado de software IBM.

Si vive en los EE.UU., puede llamar a uno de los números siguientes:

- 1-800-237-5511 para obtener asistencia técnica
- 1-888-426-4343 para conocer las opciones de servicio técnico disponibles

Información sobre productos

Si vive en los EE.UU., puede llamar a uno de los números siguientes:

- 1-800-IBM-CALL (1-800-426-2255) ó 1-800-3IBM-OS2 (1-800-342-6672) para solicitar productos u obtener información general.
- 1-800-879-2755 para solicitar publicaciones.

http://www.ibm.com/software/data/db2/everyplace/
Las páginas Web de DB2 Everyplace proporcionan información actual sobre noticias, descripciones de productos, programas de formación, etc. referente a DB2 Everyplace.

http://www.ibm.com/software/data/db2/everyplace/library.html
La Biblioteca Técnica de DB2 Everyplace proporciona acceso a las preguntas más frecuentes (FAQ), arreglos de programa, manuales e información técnica actualizada sobre DB2 Everyplace.

Nota: Puede que esta información sólo esté disponible en inglés.

http://www.ibm.com/software/data/
Las páginas Web de DB2 proporcionan información actual sobre noticias, descripciones de productos, programas de formación, etc. referente a DB2.

http://www.ibm.com/software/data/db2/library/
La Biblioteca Técnica sobre Productos y Servicios de DB2 proporciona acceso a las preguntas más frecuentes (FAQ), arreglos de programa, manuales e información técnica actualizada sobre DB2.

Nota: Puede que esta información sólo esté disponible en inglés.

http://www.elink.ibmlink.ibm.com/pbl/pbl/
El sitio Web para solicitar Publicaciones Internacionales proporciona información sobre cómo hacer pedidos de libros.

http://www.ibm.com/education/certify/
El Professional Certification Program del sitio Web de IBM proporciona información sobre pruebas de homologación para diversos productos de IBM, incluido DB2.

ftp.software.ibm.com
Conéctese como usuario anónimo (anonymous). En el directorio `/ps/products/db2`, puede encontrar programas de demostración, arreglos de programa, información y herramientas referentes a DB2 y a muchos otros productos.

© Copyright IBM Corp. 1999,2003

comp.databases.ibm-db2, bit.listserv.db2-l
En estos foros de discusión de Internet los usuarios pueden hablar sobre sus experiencias con productos DB2.

En Compuserve: GO IBMDB2
Especifique este mandato para acceder a los foros sobre la familia de productos IBM DB2. Todos los productos DB2 están soportados mediante estos foros.

Para obtener información sobre cómo ponerse en contacto con IBM desde fuera de los Estados Unidos, consulte el Apéndice A del manual *IBM Software Support Handbook* . Para acceder a este documento, vaya a la siguiente página Web: http://www.ibm.com/support/, y luego seleccione el enlace IBM Software Support Handbook, cerca de la parte inferior de la página.

Nota: En algunos países, los concesionarios autorizados de IBM deben ponerse en contacto con su estructura de soporte de concesionarios, en lugar de acudir al Centro de Soporte de IBM.

Número de Programa: 5724–D04

SC10-3942-01