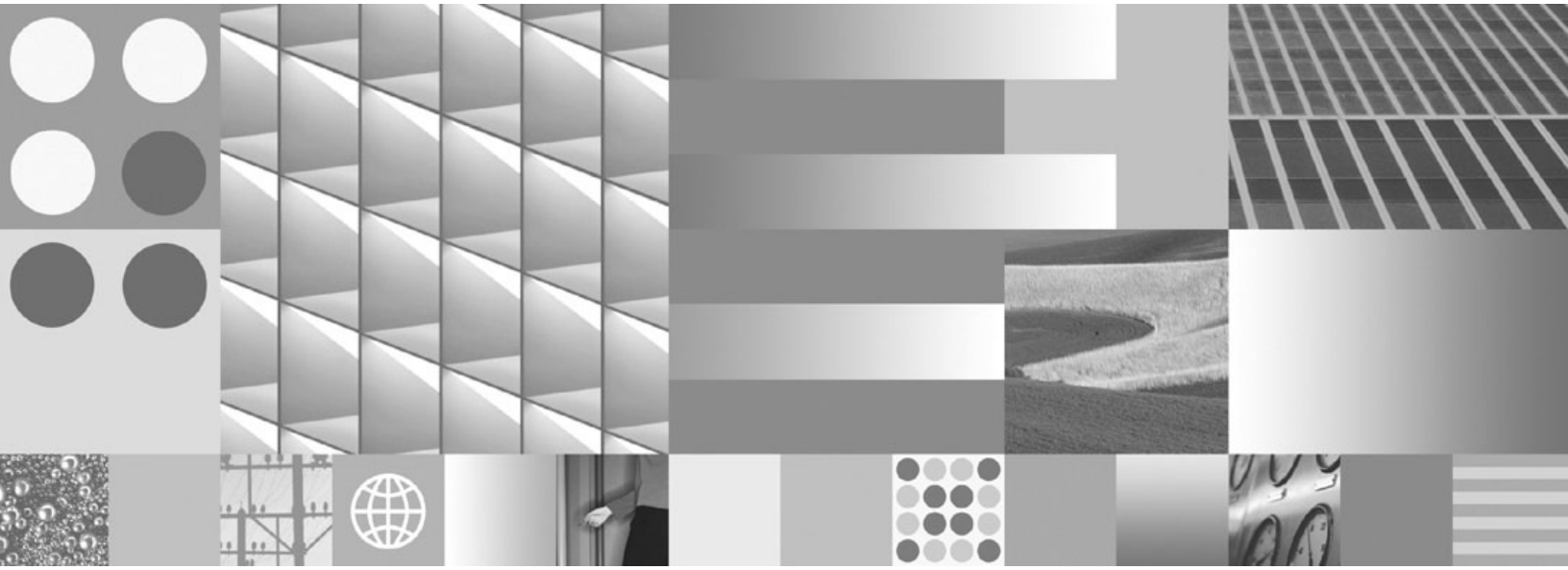
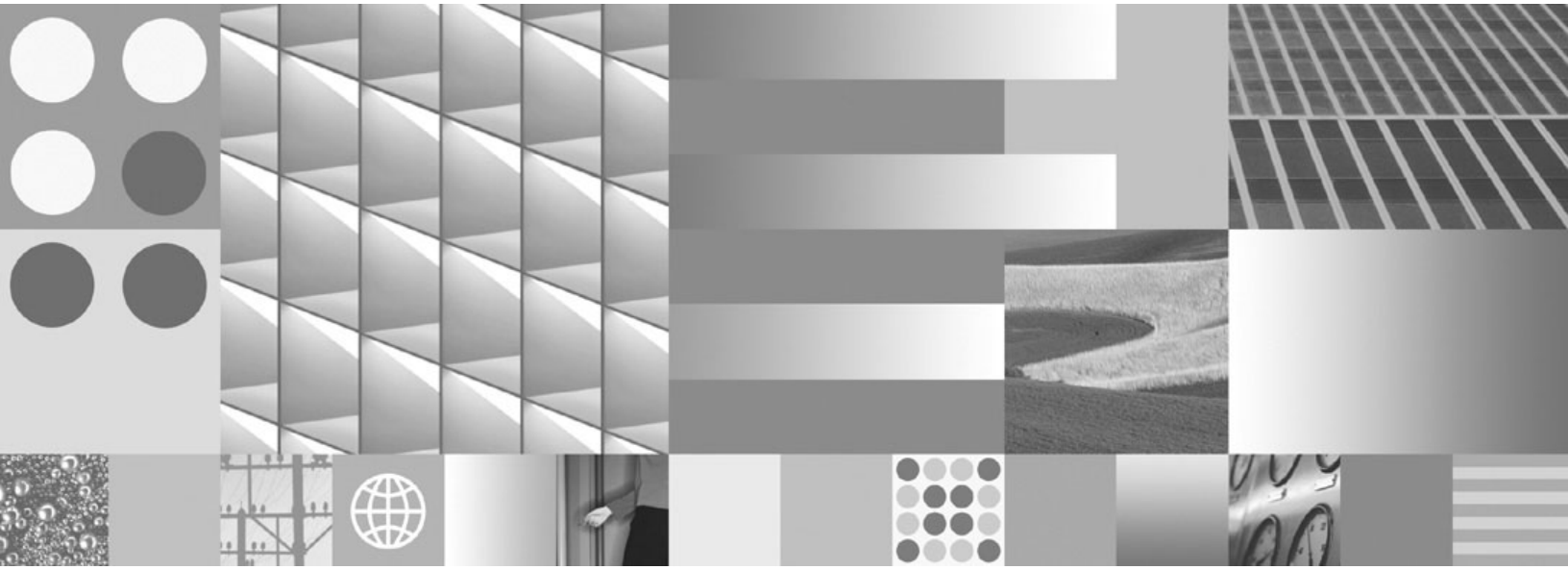
DB2 Versión 9.5
para Linux, UNIX y Windows



Consulta de las API administrativas
Actualizado en abril de 2009

SC11-3505-02

DB2 Versión 9.5
para Linux, UNIX y Windows



Consulta de las API administrativas
Actualizado en abril de 2009

SC11-3505-02

Nota

Antes de utilizar esta información y el producto al que da soporte, lea la información general contenida en el Apéndice D, "Avisos", en la página 637.

Nota de edición

Este manual es la traducción del original en inglés *DB2 Version 9.5 for Linux, UNIX, and Windows Administrative API Reference* (SC23-5842-02). Este documento contiene información propiedad de IBM. Se proporciona según un acuerdo de licencia y está protegido por la ley de la propiedad intelectual. La información contenida en esta publicación no incluye ninguna garantía de producto, por lo que ninguna declaración proporcionada en este manual deberá interpretarse como tal.

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

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

Para realizar pedidos de publicaciones de DB2 desde DB2 Marketing and Sales, en los EE.UU. o en Canadá, llame al 1-800-IBM-4YOU (426-4968).

Cuando envía información a IBM, otorga a IBM un derecho no exclusivo a utilizar o distribuir dicha información en la forma en que IBM considere adecuada, sin contraer por ello ninguna obligación con el remitente.

Contenido

Acerca de este manual

Este manual proporciona información sobre cómo utilizar las interfaces de programación de aplicaciones (API) para ejecutar funciones administrativas de las bases de datos. Presenta información detallada acerca del uso de llamadas a la API del gestor de bases de datos en aplicaciones escritas en los lenguajes de programación siguientes:

- C
- C++
- COBOL
- FORTRAN
- REXX

Para un lenguaje compilado, es necesario que esté disponible un precompilador para procesar las sentencias. Se proporcionan precompiladores para todos los lenguajes soportados.

Quién debe utilizar este manual

Se supone que el lector tiene conocimientos sobre la programación de aplicaciones y la administración de bases de datos, así como conocimientos sobre:

- Structured Query Language (SQL)
- Los lenguajes de programación C, C++, COBOL, FORTRAN, y/o REXX
- El diseño de programas de aplicación

Cómo esta estructurado este manual

Este manual proporciona la información de consulta necesaria para utilizar las API administrativas en el desarrollo de aplicaciones.

Las principales áreas temáticas tratadas en los capítulos de este manual son las siguientes:

Visión genera de las API administrativas y estructuras de datos

- El capítulo 1, “DB2 API,” incluye tablas que muestran las API administrativas, los archivos de inclusión y los programas de ejemplo.
- El capítulo 2, “API y estructuras de datos modificadas” utiliza tablas para mostrar las API soportadas y no soportadas y las estructuras de datos que se han cambiado.
- El capítulo 3, “Cómo están organizadas las descripciones de las API,” describe cómo están organizadas las descripciones de las API y muestra los archivos de inclusión para las aplicaciones de API DB2.

API

- El capítulo 4, “API administrativas” muestra, en orden alfabético, las API administrativas de DB2.
- El capítulo 5, “Llamada a las API de DB2 en REXX,” describe cómo llamar a las API de DB2 desde una aplicación REXX.

- El capítulo 6, “API de gestión de transacciones dudosas” presenta un conjunto de API proporcionadas para que los escritores de herramientas lleven a cabo funciones heurísticas en transacciones dudosas.
- El capítulo 7, “Aplicaciones con hebras con acceso simultáneo,” describe las API de DB2 que pueden utilizarse en aplicaciones de hebras.

API de plugins

- El capítulo 8, “Plugins del sistema de base de datos de DB2 para la personalización de la gestión de las bases de datos ” presenta las API de plugins de seguridad, copia de seguridad, restauración, archivado de anotaciones y compresión/descompresión para las imágenes de copia de seguridad.

Estructuras de datos

- El capítulo 9, “Estructuras de datos utilizadas por las API,” describe las estructuras de datos utilizadas por las API.

Apéndices

- El Apéndice A, “API de personalización del precompilador,” proporciona un enlace con lugares donde obtener información sobre un conjunto de API documentadas que permiten que otras herramientas de desarrollo de aplicaciones implementen el soporte del precompilador para DB2 directamente en sus productos.
- El Apéndice B, “Registros de anotaciones de DB2,” describe la estructura de los distintos registros de anotaciones de DB2.

Convenios de resaltado

En este manual se utilizan los siguientes convenios de resaltado.

Negrita	Indica mandatos, palabras clave y otros elementos cuyos nombres ha predefinido el sistema.
Cursiva	Indica uno de los casos siguientes: • Nombres o valores (variables) que el usuario debe proporcionar • Énfasis general • La introducción de un término nuevo • Una referencia a otra fuente de información
`Monoespaciado`	Indica uno de los casos siguientes: • Archivos y directorios • Información que el usuario debe escribir en una solicitud de mandatos o en una ventana • Ejemplos de valores de datos concretos • Ejemplo de texto similares a la forma en que es posible que el sistema los visualice • Ejemplos de mensajes de texto • Ejemplos de código de programación

Capítulo 1. Las API de DB2

Las tablas siguientes muestran las API de DB2 con los programas de ejemplo de DB2. La primera tabla lista las API de DB2 agrupadas según la categoría funcional, sus respectivos archivos de inclusión y los programas de ejemplo que muestran su utilización (consulte la nota incluida a continuación de la tabla para obtener más información sobre los archivos de inclusión). La segunda tabla lista los programas de ejemplo de C/C++ y muestra las API de DB2 ejemplificadas en cada programa de C/C++. La tercera tabla muestra los programas de ejemplo de COBOL y las API de DB2 ejemplificadas en cada programa COBOL.

Las API de DB2, archivos de inclusión y programas de ejemplo
Tabla 1.

Programas de ejemplo de C/C++ con las API de DB2
Tabla 2 en la página 15.

Programas de ejemplo de COBOL con las API de DB2
Tabla 3 en la página 18.

Tabla 1. Las API de DB2, archivos de inclusión y programas de ejemplo

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
Las API de control de bases de datos	"db2DatabaseQuiesce - Inmovilizar la base de datos" en la página 68	db2ApiDf	n/d
Las API de control de bases de datos	"db2DatabaseUnquiesce - Movilizar base de datos" en la página 73	db2ApiDf	n/d
Las API de control de bases de datos	"db2DatabaseRestart - Reiniciar base de datos" en la página 70	db2ApiDf	C: dbconn.sqc C++: dbconn.sqC
Las API de control de bases de datos	"sqlecrea - Crear una base de datos" en la página 336	sqlenv	C: dbcreate.c dbrecov.sqc dbsample.sqc C++: dbcreate.C dbrecov.sq COBOL: db_udcs.cbl dbconf.cbl ebcdicdb.cbl
Las API de control de bases de datos	"sqlecran - Crear una base de datos en un servidor de particiones de base de datos" en la página 335	sqlenv	n/d
Las API de control de bases de datos	"sqledrpd - Descartar base de datos" en la página 350	sqlenv	C: dbcreate.c C++: dbcreate.C COBOL: dbconf.cbl
Las API de control de bases de datos	"sqledpan - Descartar una base de datos de un servidor de particiones de base de datos" en la página 349	sqlenv	n/d
Las API de control de bases de datos	"sqlemgdb - Migrar la versión anterior de la base de datos DB2 a la versión actual" en la página 368	sqlenv	C: dbmigrat.c C++: dbmigrat.C COBOL: migrate.cbl
Las API de control de bases de datos	"db2XaListIndTrans - Listar transacciones dudosas" en la página 407	db2ApiDf	n/d
Las API de control de bases de datos	"sqle_activate_db - Activar base de datos" en la página 319	sqlenv	n/d

Tabla 1. Las API de DB2, archivos de inclusión y programas de ejemplo (continuación)

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
Las API de control de bases de datos	"sqle_deactivate_db - Desactivar base de datos" en la página 321	sqlenv	n/d
Las API de control de bases de datos	"sqlcspqy - Listar transacciones dudosas DRDA" en la página 318	sqlxa	n/d
Las API de control de bases de datos	"db2SetWriteForDB - Suspender o reanudar las escrituras de E/S para la base de datos" en la página 275	db2ApiDf	n/d
Las API de control de bases de datos	"sqlefrce - Desconectar usuarios y aplicaciones del sistema" en la página 355	sqlenv	C: dbconn.sqc dbsample.sqc instart.c C++: dbconn.sqC instart.C COBOL: dbstop.cbl
Las API de control de instancias	"db2InstanceStart - Iniciar instancia" en la página 149	db2ApiDf	C: instart.c C++: instart.C
Las API de control de instancias	"db2InstanceStop - Detener instancia" en la página 154	db2ApiDf	C: instart.c C++: instart.C
Las API de control de instancias	"db2InstanceQuiesce - Inmovilizar instancia" en la página 147	db2ApiDf	n/d
Las API de control de instancias	"db2InstanceUnquiesce - Movilizar instancia" en la página 157	db2ApiDf	n/d
Las API de control de instancias	"sqleatin - Conectar a instancia" en la página 327	sqlenv	C: inattach.c utilapi.c C++: inattach.C utilapi.C COBOL: dbinst.cbl
Las API de control de instancias	"sqleatcp - Conectar a instancia y cambiar contraseña" en la página 325	sqlenv	C: inattach.c C++: inattach.C COBOL: dbinst.cbl
Las API de control de instancias	"sqledtin - Desconectar de instancia" en la página 353	sqlenv	C: inattach.c utilapi.c C++: inattach.C utilapi.C COBOL: dbinst.cbl
Las API de control de instancias	"sqlegins - Obtener instancia actual" en la página 365	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dbinst.cbl
Las API de control de instancias	"sqluadau - Obtener autorizaciones del usuario actual" en la página 390 **Nota:** Esta API está en desuso para DB2 V9.5.	sqlutil	n/d
Las API de control de instancias	"db2UtilityControl - Establecer el nivel de prioridad de los programas de utilidad en ejecución" en la página 293	db2ApiDf	n/d
Las API del gestor de bases de datos y de configuración de bases de datos	"db2CfgGet - Obtener los parámetros de configuración del gestor de bases de datos o de la base de datos" en la página 57	db2ApiDf	C: dbinfo.c dbrecov.sqc ininfo.c tscreate.sqc C++: dbinfo.C dbrecov.sqC ininfo.C tscreate.sqC

Tabla 1. Las API de DB2, archivos de inclusión y programas de ejemplo (continuación)

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
Las API del gestor de bases de datos y de configuración de bases de datos	"db2CfgSet - Definir los parámetros de configuración del gestor de bases de datos o de la base de datos" en la página 60	db2ApiDf	C: dbinfo.c dbrecov.sqc ininfo.c C++: dbinfo.C dbrecov.sqC ininfo.C
Las API del gestor de bases de datos y de configuración de bases de datos	"db2AutoConfig - Acceder al Asesor de configuración" en la página 41	db2AuCfg	C: dbcfg.sqc C++: dbcfg.sqC
Las API del gestor de bases de datos y de configuración de bases de datos	"db2AutoConfigFreeMemory - Liberar la memoria asignada por la API db2AutoConfig" en la página 45	db2AuCfg	C: dbcfg.sqc C++: dbcfg.sqC
Las API de supervisión de bases de datos	"db2GetSnapshotSize - Calcular el tamaño del almacenamiento intermedio de salida necesario para la API db2GetSnapshot" en la página 105	db2ApiDf	n/d
Las API de supervisión de bases de datos	"db2AddSnapshotRequest - Añadir una petición de instantánea" en la página 36	db2ApiDf	n/d
Las API de supervisión de bases de datos	"db2MonitorSwitches - Obtener o actualizar los valores de los conmutadores del supervisor" en la página 202	db2ApiDf	C: utilsnap.c C++: utilsnap.C
Las API de supervisión de bases de datos	"db2GetSnapshot - Obtener una instantánea del estado operacional del gestor de bases de datos" en la página 102	db2ApiDf	C: utilsnap.c C++: utilsnap.C
Las API de supervisión de bases de datos	"db2ResetMonitor - Restaurar los datos del supervisor del sistema de base de datos" en la página 234	db2ApiDf	n/d
Las API de supervisión de bases de datos	"db2ConvMonStream - Convertir la corriente de supervisor en el formato anterior a la versión 6" en la página 64	db2ApiDf	n/d
Las API de supervisión de bases de datos	"db2Inspect - Inspeccionar la base de datos para comprobar la integridad de la arquitectura" en la página 140	db2ApiDf	n/d
Las API de supervisión de estado de bases de datos	"db2AddContact - Añadir un contacto al que se pueden enviar mensajes de notificación" en la página 33	db2ApiDf	n/d
Las API de supervisión de estado de bases de datos	"db2AddContactGroup - Añadir un grupo de contactos al que se pueden enviar mensajes de notificación" en la página 34	db2ApiDf	n/d
Las API de supervisión de estado de bases de datos	"db2DropContact - Eliminar un contacto de la lista de contactos a los que se pueden enviar mensajes de notificación" en la página 80	db2ApiDf	n/d

Tabla 1. Las API de DB2, archivos de inclusión y programas de ejemplo (continuación)

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
Las API de supervisión de estado de bases de datos	"db2DropContactGroup - Eliminar un grupo de contactos de la lista de contactos a los que se pueden enviar mensajes de notificación" en la página 81	db2ApiDf	n/d
Las API de supervisión de estado de bases de datos	"db2GetAlertCfg - Obtener los valores de configuración de alertas para los indicadores de salud" en la página 89	db2ApiDf	n/d
Las API de supervisión de estado de bases de datos	"db2GetAlertCfgFree - Liberar la memoria asignada por la API db2GetAlertCfg" en la página 93	db2ApiDf	n/d
Las API de supervisión de estado de bases de datos	"db2GetContactGroup - Obtener la lista de contactos de un solo grupo de contactos al que se puedan enviar mensajes de notificación" en la página 94	db2ApiDf	n/d
Las API de supervisión de estado de bases de datos	"db2GetContactGroups - Obtener la lista de grupos de contactos a los que se puedan enviar mensajes de notificación" en la página 95	db2ApiDf	n/d
Las API de supervisión de estado de bases de datos	"db2GetContacts - Obtener la lista de contactos a los que se pueden enviar mensajes de notificación" en la página 96	db2ApiDf	n/d
Las API de supervisión de estado de bases de datos	"db2GetHealthNotificationList - Obtener la lista de contactos a los que se puedan enviar notificaciones de alerta de salud" en la página 98	db2ApiDf	n/d
Las API de supervisión de estado de bases de datos	"db2ResetAlertCfg - Restablecer la configuración de alertas de los indicadores de salud" en la página 232	db2ApiDf	n/d
Las API de supervisión de estado de bases de datos	"db2UpdateAlertCfg - Actualizar los valores de configuración de alertas para los indicadores de salud" en la página 281	db2ApiDf	n/d
Las API de supervisión de estado de bases de datos	"db2UpdateContact - Actualizar los atributos de un contacto" en la página 288	db2ApiDf	n/d
Las API de supervisión de estado de bases de datos	"db2UpdateContactGroup - Actualizar los atributos de un grupo de contactos" en la página 290	db2ApiDf	n/d
Las API de supervisión de estado de bases de datos	"db2UpdateHealthNotificationList - Actualizar la lista de contactos a los que se puedan enviar notificaciones de alerta de salud" en la página 291	db2ApiDf	n/d

Tabla 1. Las API de DB2, archivos de inclusión y programas de ejemplo (continuación)

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
Las API de supervisión de estado de bases de datos	"db2GetSnapshot - Obtener una instantánea del estado operacional del gestor de bases de datos" en la página 102	db2ApiDf	C: utilsnap.c C++: utilsnap.C
Las API de supervisión de estado de bases de datos	"db2GetSnapshotSize - Calcular el tamaño del almacenamiento intermedio de salida necesario para la API db2GetSnapshot" en la página 105	db2ApiDf	n/d
Las API de supervisión de estado de bases de datos	"db2GetRecommendations - Obtener recomendaciones para resolver un indicador de salud en estado de alerta" en la página 99	db2ApiDf	n/d
Las API de supervisión de estado de bases de datos	"db2GetRecommendationsFree - Liberar la memoria asignada por la API db2GetRecommendations" en la página 101	db2ApiDf	n/d
Las API de traslado de datos	"db2Export - Exportar datos de una base de datos" en la página 82	sqlutil	C: tbmove.sqc C++: tbmove.sqC COBOL: expsamp.sqb impexp.sqb tload.sqb
Las API de traslado de datos	"db2Import - Importar datos a una tabla, jerarquía, apodo o vista" en la página 125	db2ApiDf	C: dtformat.sqc tbmove.sqc C++: tbmove.sqC COBOL: expsamp.sqb impexp.sqb
Las API de traslado de datos	"db2Load - Cargar datos en una tabla" en la página 172	db2ApiDf	C: dtformat.sqc tbload.sqc tbmove.sqc C++: tbmove.sqC
Las API de traslado de datos	"db2LoadQuery - Obtener el estado de una operación de carga" en la página 194	db2ApiDf	C: tbmove.sqc C++: tbmove.sqC COBOL: loadqry.sqb
Las API de recuperación	"db2Backup - Hacer copia de seguridad de una base de datos o un espacio de tablas" en la página 46	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
Las API de recuperación	"db2Restore - Restaurar una base de datos o un espacio de tablas" en la página 236	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
Las API de recuperación	"db2Recover - Restaurar y avanzar una base de datos" en la página 219	db2ApiDf	n/d
Las API de recuperación	"db2Rollforward - Avanzar una base de datos" en la página 252	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
Las API de recuperación	"db2HistoryOpenScan - Iniciar una exploración del archivo histórico" en la página 118	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
Las API de recuperación	"db2HistoryGetEntry - Obtener la entrada siguiente del archivo histórico" en la página 116	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
Las API de recuperación	"db2HistoryCloseScan - Finalizar la exploración del archivo histórico" en la página 115	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
Las API de recuperación	"db2Prune - Suprimir las entradas del archivo histórico o archivos de anotaciones cronológicas de la vía de acceso de anotación cronológica activa" en la página 205	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC

Tabla 1. Las API de DB2, archivos de inclusión y programas de ejemplo (continuación)

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
Las API de recuperación	"db2HistoryUpdate - Actualizar una entrada de archivo histórico" en la página 122	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
Las API de recuperación	"db2ArchiveLog - Archivar el archivo de anotaciones cronológicas activo" en la página 39	db2ApiDf	n/d
Las API de HADR (High Availability Disaster Recovery)	"db2HADRStart - Iniciar operaciones de HADR (high availability disaster recovery)" en la página 109	db2ApiDf	n/d
Las API de HADR (High Availability Disaster Recovery)	"db2HADRStop - Detener operaciones de HADR (high availability disaster recovery)" en la página 111	db2ApiDf	n/d
Las API de HADR (High Availability Disaster Recovery)	"db2HADRTakeover - Dar instrucciones a una base de datos para que se convierta en la base de datos primaria de HADR (high availability disaster recovery)" en la página 112	db2ApiDf	n/d
Las API de gestión de directorios de bases de datos y de DCS	"sqlecadb - Catalogar una base de datos del directorio de bases de datos del sistema" en la página 329	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dbcat.cbl
Las API de gestión de directorios de bases de datos y de DCS	"sqleuncd - Descatalogar una base de datos del directorio de bases de datos del sistema" en la página 383	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dbcat.cbl
Las API de gestión de directorios de bases de datos y de DCS	"sqlegdad - Catalogar una base de datos en el directorio de DCS (Database Connection Services)" en la página 357	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cbl
Las API de gestión de directorios de bases de datos y de DCS	"sqlegdel - Descatalogar una base de datos del directorio de DCS (Database Connection Services)" en la página 359	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cbl
Las API de gestión de directorios de bases de datos y de DCS	"sqledcgd - Cambiar un comentario de base de datos en el directorio de bases de datos locales o del sistema" en la página 347	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dbcmt.cbl
Las API de gestión de directorios de bases de datos y de DCS	"db2DbDirOpenScan - Iniciar una exploración del directorio de bases de datos locales o del sistema" en la página 78	db2ApiDf	C: ininfo.c C++: ininfo.C COBOL: dbcat.cbl dbcmt.cbl

Tabla 1. Las API de DB2, archivos de inclusión y programas de ejemplo (continuación)

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
Las API de gestión de directorios de bases de datos y de DCS	"db2DbDirGetNextEntry - Obtener la entrada siguiente del directorio de bases de datos locales o del sistema" en la página 75	db2ApiDf	C: ininfo.c C++: ininfo.C COBOL: dbcat.cbl dbcmt.cbl
Las API de gestión de directorios de bases de datos y de DCS	"db2DbDirCloseScan - Finalizar una exploración del directorio de bases de datos locales o del sistema" en la página 74	db2ApiDf	C: ininfo.c C++: ininfo.C COBOL: dbcat.cbl dbcmt.cbl
Las API de gestión de directorios de bases de datos y de DCS	"sqlegdcl - Iniciar una exploración del directorio de DCS (Database Connection Services)" en la página 364	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cbl
Las API de gestión de directorios de bases de datos y de DCS	"sqlegdgt - Obtener entradas del directorio de DCS (servicios de conexión de base de datos)" en la página 362	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cbl
Las API de gestión de directorios de bases de datos y de DCS	"sqlegdcl - Finalizar una exploración del directorio de DCS (Database Connection Services)" en la página 359	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cbl
Las API de gestión de directorios de bases de datos y de DCS	"sqlegdge - Obtener una entrada específica del directorio de DCS (Database Connection Services)" en la página 361	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cbl
Las API de gestión de directorios de bases de datos y de DCS	"db2UpdateAlternateServerForDB - Actualizar el servidor alternativo para un alias de base de datos en el directorio de bases de datos del sistema" en la página 287	db2ApiDf	n/d
Las API de gestión de cliente/servidor	"sqleqryc - Consultar valores de conexión del cliente" en la página 374	sqlenv	C: cli_info.c C++: cli_info.C COBOL: client.cbl
Las API de gestión de cliente/servidor	"sqleqryi - Consultar información sobre el cliente" en la página 375	sqlenv	C: cli_info.c C++: cli_info.C
Las API de gestión de cliente/servidor	"sqlesetc - Establecer valores de conexión del cliente" en la página 379	sqlenv	C: cli_info.c dbcfg.sqc dbmcon.sqc C++: cli_info.C dbcfg.sqC dbmcon.sqC COBOL: client.cbl
Las API de gestión de cliente/servidor	"sqleseti - Establecer información sobre el cliente" en la página 381	sqlenv	C: cli_info.c C++: cli_info.C
Las API de gestión de cliente/servidor	"sqlesact - Establecer serie de contabilidad" en la página 376	sqlenv	COBOL: setact.cbl

Tabla 1. Las API de DB2, archivos de inclusión y programas de ejemplo (continuación)

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
Las API de gestión de cliente/servidor	"db2DatabasePing - Sondear la base de datos para probar el tiempo de respuesta de la red" en la página 67	db2ApiDf	n/d
Las API de gestión de cliente/servidor	"sqleisig - Instalar manejador de señales" en la página 367	sqlenv	COBOL: dbcmt.cbl
Las API de gestión de cliente/servidor	"sqleintr - Interrumpir peticiones de aplicaciones" en la página 366	sqlenv	n/d
Las API de gestión de directorios de LDAP (Lightweight Directory Access Protocol)	"db2LdapRegister - Registrar el servidor DB2 en el servidor LDAP" en la página 162	db2ApiDf	n/d
Las API de gestión de directorios de LDAP (Lightweight Directory Access Protocol)	"db2LdapUpdate - Actualizar los atributos del servidor DB2 en el servidor LDAP" en la página 168	db2ApiDf	n/d
Las API de gestión de directorios de LDAP (Lightweight Directory Access Protocol)	"db2LdapDeregister - Desregistrar el servidor DB2 y las bases de datos catalogadas del servidor LDAP" en la página 161	db2ApiDf	n/d
Las API de gestión de directorios de LDAP (Lightweight Directory Access Protocol)	"db2LdapCatalogNode - Proporcionar un alias para el nombre de nodo en el servidor LDAP" en la página 160	db2ApiDf	n/d
Las API de gestión de directorios de LDAP (Lightweight Directory Access Protocol)	"db2LdapUncatalogNode - Suprimir alias para nombre de nodo del servidor LDAP" en la página 167	db2ApiDf	n/d
Las API de gestión de directorios de LDAP (Lightweight Directory Access Protocol)	"db2LdapCatalogDatabase - Registrar la base de datos en el servidor LDAP" en la página 158	db2ApiDf	n/d

Tabla 1. Las API de DB2, archivos de inclusión y programas de ejemplo (continuación)

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
Las API de gestión de directorios de LDAP (Lightweight Directory Access Protocol)	"db2LdapUncatalogDatabase - Desregistrar base de datos del servidor LDAP" en la página 166	db2ApiDf	n/d
Las API de gestión de directorios de LDAP (Lightweight Directory Access Protocol)	"db2LdapUpdateAlternateServerForDB - Actualizar el servidor alternativo de la base de datos en el servidor LDAP" en la página 171	db2ApiDf	n/d
Las API de programación y preparación de aplicaciones	"sqlaintp - Obtener mensaje de error" en la página 297	sql	C: dbcfg.sqc utilapi.c C++: dbcfg.sqC utilapi.C COBOL: checkerr.cbl
Las API de programación y preparación de aplicaciones	"sqlogstt - Obtener el mensaje de SQLSTATE" en la página 388	sql	C: utilapi.c C++: utilapi.C COBOL: checkerr.cbl
Las API de programación y preparación de aplicaciones	"sqleisig - Instalar manejador de señales" en la página 367	sqlenv	COBOL: dbcmt.cbl
Las API de programación y preparación de aplicaciones	"sqleintr - Interrumpir peticiones de aplicaciones" en la página 366	sqlenv	n/d
Las API de programación y preparación de aplicaciones	"sqlaprep - Precompilar programa de aplicación" en la página 299	sql	C: dbpkg.sqc C++: dbpkg.sqC
Las API de programación y preparación de aplicaciones	"sqlabndx - Programa de aplicación de vinculación para crear un paquete" en la página 294	sql	C: dbpkg.sqc dbsample.sqc C++: dbpkg.sqC
Las API de programación y preparación de aplicaciones	"sqlarbnd - Volver a vincular paquete" en la página 301	sql	C: dbpkg.sqc C++: dbpkg.sqC COBOL: rebind.sqb
Las API específicas de aplicaciones COBOL, FORTRAN y REXX	"sqlgaddr - Obtener la dirección de una variable" en la página 386	sqlutil	n/d

Tabla 1. Las API de DB2, archivos de inclusión y programas de ejemplo (continuación)

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
Las API específicas de aplicaciones COBOL, FORTRAN y REXX	"sqlgdref - Eliminar la referencia de una dirección" en la página 387	sqlutil	n/d
Las API específicas de aplicaciones COBOL, FORTRAN y REXX	"sqlgmcpy - Copiar datos de un área de memoria a otra" en la página 387	sqlutil	n/d
Las API de gestión de espacios de tablas y gestión de tablas	"sqlbtcq - Obtener los datos de la consulta para todos los contenedores de espacios de tablas" en la página 317	sqlutil	C: dbrecov.sqc tsinfo.sqc C++: dbrecov.sqC tsinfo.sqC COBOL: tabscont.sqb tspace.sqb
Las API de gestión de espacios de tablas y gestión de tablas	"sqlbotcq - Abrir una consulta de contenedor de espacio de tablas" en la página 310	sqlutil	C: tsinfo.sqc C++: tsinfo.sqC COBOL: tabscont.sqb tspace.sqb
Las API de gestión de espacios de tablas y gestión de tablas	"sqlbftcq - Captar los datos de la consulta para filas de un contenedor de espacio de tablas" en la página 305	sqlutil	C: tsinfo.sqc C++: tsinfo.sqC COBOL: tabscont.sqb tspace.sqb
Las API de gestión de espacios de tablas y gestión de tablas	"sqlbctcq - Cerrar una consulta de contenedor de espacio de tablas" en la página 304	sqlutil	C: tsinfo.sqc C++: tsinfo.sqC COBOL: tabscont.sqb tspace.sqb
Las API de gestión de espacios de tablas y gestión de tablas	"sqlbstsc - Definir contenedores de espacios de tablas" en la página 315	sqlutil	C: dbrecov.sqc C++: dbrecov.sqC COBOL: tabscont.sqb tspace.sqb
Las API de gestión de espacios de tablas y gestión de tablas	"sqlbmtsq - Obtener los datos de la consulta para todos los espacios de tablas" en la página 308	sqlutil	C: dbrecov.sqc tsinfo.sqc C++: dbrecov.sqC tsinfo.sqC COBOL: tabspace.sqb tspace.sqb
Las API de gestión de espacios de tablas y gestión de tablas	"sqlbstpq - Obtener información sobre un espacio de tablas individual" en la página 313	sqlutil	C: tsinfo.sqc C++: tsinfo.sqC COBOL: tabspace.sqb tspace.sqb
Las API de gestión de espacios de tablas y gestión de tablas	"sqlbotsq - Abrir una consulta de espacio de tablas" en la página 312	sqlutil	C: tsinfo.sqc C++: tsinfo.sqC COBOL: tabspace.sqb tspace.sqb

Tabla 1. Las API de DB2, archivos de inclusión y programas de ejemplo (continuación)

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
Las API de gestión de espacios de tablas y gestión de tablas	"sqlbftpq - Captar los datos de la consulta para filas de un espacio de tablas" en la página 306	sqlutil	C: tsinfo.sqc C++: tsinfo.sqC COBOL: tabspace.sqb tspace.sqb
Las API de gestión de espacios de tablas y gestión de tablas	"sqlbctsq - Cerrar una consulta de espacio de tablas" en la página 304	sqlutil	C: tsinfo.sqc C++: tsinfo.sqC COBOL: tabspace.sqb tspace.sqb
Las API de gestión de espacios de tablas y gestión de tablas	"sqlbgtss - Obtener estadísticas de utilización del espacio de tablas" en la página 307	sqlutil	C: tsinfo.sqc C++: tsinfo.sqC COBOL: tabspace.sqb tspace.sqb
Las API de gestión de espacios de tablas y gestión de tablas	"sqluvqdp - Inmovilizar espacios de tablas para una tabla" en la página 399	sqlutil	C: tbmove.sqc C++: tbmove.sqC COBOL: tload.sqb
Las API de gestión de espacios de tablas y gestión de tablas	"db2Runstats - Actualizar estadísticas para tablas e índices" en la página 263	db2ApiDf	C: tbreorg.sqc C++: tbreorg.sqC COBOL: dbstat.sqb
Las API de gestión de espacios de tablas y gestión de tablas	"db2Reorg - Reorganizar un índice o una tabla" en la página 225	db2ApiDf	C: tbreorg.sqc C++: tbreorg.sqC COBOL: dbstat.sqb
Las API de gestión de espacios de tablas y gestión de tablas	"sqlefmem - Liberar la memoria asignada por las API sqlbtcq y sqlbmtsq" en la página 354	sqlenv	C: dbrecov.sqc tsinfo.sqc C++: dbrecov.sqC tsinfo.sqC COBOL: tabscont.sqb tabspace.sqb tspace.sqb
Las API de gestión de directorios de nodos	"sqlectnd - Catalogar una entrada en el directorio de nodos" en la página 344	sqlenv	C: ininfo.c C++: ininfo.C COBOL: nodecat.cbl
Las API de gestión de directorios de nodos	"sqleuncn - Descatalogar una entrada del directorio de nodos" en la página 385	sqlenv	C: ininfo.c C++: ininfo.C COBOL: nodecat.cbl
Las API de gestión de directorios de nodos	"sqlenops - Iniciar una exploración del directorio de nodos" en la página 372	sqlenv	C: ininfo.c C++: ininfo.C COBOL: nodecat.cbl
Las API de gestión de directorios de nodos	"sqlengne - Obtener la entrada siguiente del directorio de nodos" en la página 371	sqlenv	C: ininfo.c C++: ininfo.C COBOL: nodecat.cbl

Tabla 1. Las API de DB2, archivos de inclusión y programas de ejemplo (continuación)

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
Las API de gestión de directorios de nodos	"sqlencl - Finalizar una exploración del directorio de nodos" en la página 370	sqlenv	C: ininfo.c C++: ininfo.C COBOL: nodecat.cbl
Las API de gestión de directorios de nodos	"db2UpdateAlternateServerForDB - Actualizar el servidor alternativo para un alias de base de datos en el directorio de bases de datos del sistema" en la página 287	db2ApiDf	n/d
Las API de sincronización de satélites	"db2GetSyncSession - Obtener un identificador de sesión de sincronización de satélites" en la página 108	db2ApiDf	n/d
Las API de sincronización de satélites	"db2QuerySatelliteProgress - Obtener el estado de una sesión de sincronización de satélites" en la página 207	db2ApiDf	n/d
Las API de sincronización de satélites	"db2SetSyncSession - Establecer sesión de sincronización de satélites" en la página 274	db2ApiDf	n/d
Las API de sincronización de satélites	"db2SyncSatellite - Iniciar sincronización de satélites" en la página 279	db2ApiDf	n/d
Las API de sincronización de satélites	"db2SyncSatelliteStop - Pausar sincronización de satélites" en la página 280	db2ApiDf	n/d
Las API de sincronización de satélites	"db2SyncSatelliteTest - Probar si se puede sincronizar un satélite" en la página 281	db2ApiDf	n/d
Las API de archivos de anotaciones de lectura	"db2ReadLog - Extraer registros de anotaciones cronológicas" en la página 209	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
Las API de archivos de anotaciones de lectura	"db2ReadLogNoConn - Leer las anotaciones cronológicas de la base de datos sin una conexión de base de datos" en la página 213	db2ApiDf	n/d
Las API de archivos de anotaciones de lectura	"db2ReadLogNoConnInit - Inicializar la lectura de las anotaciones cronológicas de la base de datos sin una conexión de base de datos" en la página 216	db2ApiDf	n/d
Las API de archivos de anotaciones de lectura	"db2ReadLogNoConnTerm - Terminar la lectura de las anotaciones cronológicas de la base de datos sin una conexión de base de datos" en la página 218	db2ApiDf	n/d
Las API de gestión de transacciones dudosas	"db2XaListIndTrans - Listar transacciones dudosas" en la página 407	db2ApiDf	n/d

Tabla 1. Las API de DB2, archivos de inclusión y programas de ejemplo (continuación)

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
Las API de gestión de transacciones dudosas	"sqlxhfrg - Olvidar estado de transacción" en la página 412	sqlxa	n/d
Las API de gestión de transacciones dudosas	"sqlxphcm - Confirmar una transacción dudosa" en la página 412	sqlxa	n/d
Las API de gestión de transacciones dudosas	"sqlxphrl - Retrotraer una transacción dudosa" en la página 413	sqlxa	n/d
Las API de gestión de transacciones dudosas	"sqlcspqy - Listar transacciones dudosas DRDA" en la página 318	sqlxa	n/d
Las API para obtener acceso simultáneo a una base de datos	"sqleAttachToCtx - Conectar a contexto" en la página 415	sql	C: dbthrds.sqc C++: dbthrds.sqC
Las API para obtener acceso simultáneo a una base de datos	"sqleBeginCtx - Crear y conectar a un contexto de aplicación" en la página 415	sql	C: dbthrds.sqc C++: dbthrds.sqC
Las API para obtener acceso simultáneo a una base de datos	"sqleDetachFromCtx - Desconectar de contexto" en la página 416	sql	C: dbthrds.sqc C++: dbthrds.sqC
Las API para obtener acceso simultáneo a una base de datos	"sqleEndCtx - Desconectar y liberar la memoria asociada a un contexto de aplicación" en la página 417	sql	n/d
Las API para obtener acceso simultáneo a una base de datos	"sqleGetCurrentCtx - Obtener contexto actual" en la página 418	sql	n/d
Las API para obtener acceso simultáneo a una base de datos	"sqleInterruptCtx - Interrumpir contexto" en la página 419	sql	n/d
Las API para obtener acceso simultáneo a una base de datos	"sqleSetTypeCtx - Definir el tipo de contexto de aplicación" en la página 420	sql	C: dbthrds.sqc C++: dbthrds.sqC
Las API de gestión de particiones de base de datos	"sqleaddn - Añadir un servidor de particiones de base de datos al entorno de bases de datos particionadas" en la página 323	sqlenv	n/d

Tabla 1. Las API de DB2, archivos de inclusión y programas de ejemplo (continuación)

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
Las API de gestión de particiones de base de datos	"sqledrpn - Comprobar si se puede descartar un servidor de particiones de base de datos" en la página 352	sqlenv	n/d
Las API de gestión de particiones de base de datos	"sqlecran - Crear una base de datos en un servidor de particiones de base de datos" en la página 335	sqlenv	n/d
Las API de gestión de particiones de base de datos	"sqledpan - Descartar una base de datos de un servidor de particiones de base de datos" en la página 349	sqlenv	n/d
Las API de gestión de particiones de base de datos	"sqlesdeg - Establecer el nivel o grado máximo de paralelismo intrapartición para la ejecución de sentencias de SQL" en la página 377	sqlenv	C: ininfo.c C++: ininfo.C
Las API de gestión de particiones de base de datos	"qlugtpi - Obtener información de distribución de tablas" en la página 398	sqlutil	n/d
Las API de gestión de particiones de base de datos	"sqlugrpn - Obtener el número de servidor de particiones de base de datos para una fila" en la página 395	sqlutil	n/d
API diversas	"db2AdminMsgWrite - Escribir mensajes de anotación cronológica para la función de administración y duplicación" en la página 37	db2ApiDf	n/d
API diversas	"db2XaGetInfo - Obtener información para un gestor de recursos" en la página 406	sqlxa	n/d

Nota: Las extensiones de los archivos de inclusión varían según el lenguaje de programación. Los archivos de inclusión de C/C++ tienen la extensión de archivo `.h`. Los archivos de inclusión de COBOL tienen la extensión de archivo `.cbl`. Los archivos de inclusión se pueden encontrar en los directorios siguientes:

C/C++ (UNIX):

```
sqllib/include
```

C/C++ (Windows):

```
sqllib\include
```

COBOL (UNIX):

```
sqllib/include/cobol_a

sqllib/include/cobol_i

sqllib/include/cobol_mf
```

COBOL (Windows):

```
sqllib\include\cobol_a

sqllib\include\cobol_i

sqllib\include\cobol_mf
```

Tabla 2. Programas de ejemplo de C/C++ con las API de DB2

Programa de ejemplo	**Las API incluidas**
cli_info.c, cli_info.C	• API sqlesetc - Establecer valores de conexión del cliente • API sqleseti - Establecer información sobre el cliente • API sqleqryc - Consultar valores de conexión del cliente • API sqleqryi - Consultar información sobre el cliente
dbcfg.sqc, dbcfg.sqC	• API db2AutoConfig - Acceder al Asesor de configuración • API db2AutoConfigFreeMemory - Liberar la memoria asignada por la API db2AutoConfig • API sqlesetc - Establecer valores de conexión del cliente • API sqlaintp - Obtener mensaje de error
dbconn.sqc, dbconn.sqC	• API db2DatabaseRestart - Reiniciar base de datos • API sqlefrc - Desconectar usuarios y aplicaciones del sistema
dbcreate.c, dbcreate.C	• API sqlecrea - Crear base de datos • API sqledrpd - Descartar base de datos
dbinfo.c, dbinfo.C	• API db2CfgGet - Obtener los parámetros de configuración del gestor de bases de datos o de la base de datos • API db2CfgSet - Definir los parámetros de configuración del gestor de bases de datos o de la base de datos
dbmcon.sqc, dbmcon.sqC	• API sqlesetc - Establecer valores de conexión del cliente
dbmigrat.c, dbmigrat.C	• API sqlemgdb - Migrar la versión anterior de la base de datos DB2 a la versión actual
dbpkg.sqc, dbpkg.sqC	• API sqlaprep - Precompilar programa de aplicación • API sqlabndx - Vincular programa de aplicación para crear un paquete • API sqlarbnd - Volver a vincular paquete

Tabla 2. Programas de ejemplo de C/C++ con las API de DB2 (continuación)

Programa de ejemplo	Las API incluidas
dbrecov.sqc, dbrecov.sqC	• API db2HistoryCloseScan - Finalizar la exploración del archivo histórico • API db2HistoryGetEntry - Obtener la entrada siguiente del archivo histórico • API db2HistoryOpenScan - Iniciar una exploración del archivo histórico • API db2HistoryUpdate - Actualizar una entrada de archivo histórico • API db2Prune - Suprimir las entradas del archivo histórico o archivos de anotaciones cronológicas de la vía de acceso de anotación cronológica activa • API db2CfgGet - Obtener los parámetros de configuración del gestor de bases de datos o de la base de datos • API db2CfgSet - Definir los parámetros de configuración del gestor de bases de datos o de la base de datos • API sqlbmtsq - Obtener los datos de la consulta para todos los espacios de tabla • API sqlbstsc - Definir contenedores de espacios de tablas • API sqlbtcq - Obtener los datos de la consulta para todos los contenedores de espacios de tablas • API sqlecrea - Crear base de datos • API sqledrpd - Descartar base de datos • API sqlefmem - Liberar la memoria asignada por las API sqlbtcq y sqlbmtsq • API db2Backup - Hacer copia de seguridad de una base de datos o un espacio de tablas • API db2Restore - Restaurar una base de datos o un espacio de tablas • API db2ReadLog - Lectura asíncrona de anotación cronológica • API db2ReadLogNoConn - Leer anotación cronológica sin una conexión de base de datos • API db2Rollforward - Avanzar una base de datos
dbsample.sqc	• API db2DatabaseRestart - Reiniciar base de datos • API sqlecrea - Crear base de datos • API sqlefrc - Desconectar usuarios y aplicaciones del sistema • API sqlabndx - Vincular programa de aplicación para crear un paquete
dbthrds.sqc, dbthrds.sqC	• API sqleAttachToCtx - Conectar a contexto • API sqleBeginCtx - Crear y conectar a un contexto de aplicación • API sqleDetachFromCtx - Desconectar de contexto • API sqleSetTypeCtx - Definir tipo de contexto de aplicación
dtformat.sqc	• API db2Load - Cargar datos en una tabla • API db2Import - Importar datos a una tabla, jerarquía, apodo o vista

Tabla 2. Programas de ejemplo de C/C++ con las API de DB2 (continuación)

Programa de ejemplo	**Las API incluidas**
inattach.c, inattach.C	• API sqleatcp - Conectar a instancia y cambiar contraseña • API sqleatin - Conectar a instancia • API sqledtin - Desconectar de instancia
ininfo.c, ininfo.C	• API db2CfgGet - Obtener los parámetros de configuración del gestor de bases de datos o de la base de datos • API db2CfgSet - Definir los parámetros de configuración del gestor de bases de datos o de la base de datos • API sqlegins - Obtener instancia actual • API sqlectnd - Catalogar una entrada en el directorio de nodos • API sqlenops - Iniciar una exploración del directorio de nodos • API sqlengne - Obtener la entrada siguiente del directorio de nodos • API sqlencls - Finalizar una exploración del directorio de nodos • API sqleuncn - Descatalogar una entrada del directorio de nodos • API sqlecadb - Catalogar una base de datos del directorio de bases de datos del sistema • API db2DbDirOpenScan - Iniciar una exploración del directorio de bases de datos locales o del sistema • API db2DbDirGetNextEntry - Obtener la entrada siguiente del directorio de bases de datos locales o del sistema • API sqledcgd - Cambiar un comentario de base de datos en el directorio de bases de datos locales o del sistema • API db2DbDirCloseScan - Finalizar una exploración del directorio de bases de datos locales o del sistema • API sqleuncd - Descatalogar una base de datos del directorio de bases de datos del sistema • API sqlegdad - Catalogar una base de datos en el directorio de DCS (servicios de conexión de base de datos) • API sqlegdcl - Iniciar una exploración del directorio de DCS (servicios de conexión de base de datos) • API sqlegdge - Obtener una entrada específica del directorio de DCS (servicios de conexión de base de datos) • API sqlegdgt - Obtener entradas del directorio de DCS (servicios de conexión de base de datos) • API sqlegdcl - Finalizar una exploración del directorio de DCS (servicios de conexión de base de datos) • API sqlegdel - Descatalogar una base de datos del directorio de DCS (servicios de conexión de base de datos) • API sqlesdeg - Establecer el nivel o grado máximo de paralelismo intrapartición para la ejecución de sentencias de SQL
instart.c, instart.C	• API sqlefrc - Desconectar usuarios y aplicaciones del sistema • API db2InstanceStart - Iniciar instancia • API db2InstanceStop - Detener instancia

Tabla 2. Programas de ejemplo de C/C++ con las API de DB2 (continuación)

Programa de ejemplo	Las API incluidas
tbmove.sqc, tbmove.sqC	• API db2Export - Exportar datos de una base de datos • API db2Import - Importar datos a una tabla, jerarquía, apodo o vista • API sqluvqdp - Inmovilizar espacios de tablas para una tabla • API db2Load - Cargar datos en una tabla • API db2LoadQuery - Obtener el estado de una operación de carga
tbreorg.sqc, tbreorg.sqC	• API db2Reorg - Reorganizar un índice o una tabla • API db2Runstats - Actualizar estadísticas sobre las características de una tabla e índices asociados
tscreate.sqc, tscreate.sqC	• API db2CfgGet - Obtener los parámetros de configuración del gestor de bases de datos o de la base de datos
tsinfo.sqc, tsinfo.sqC	• API sqlbstpq - Obtener información sobre un espacio de tablas individual • API sqlbgtss - Obtener estadísticas de utilización del espacio de tablas • API sqlbmtsq - Obtener los datos de la consulta para todos los espacios de tabla • API sqlefmem - Liberar la memoria asignada por las API sqlbtcq y sqlbmtsq • API sqlbotsq - Abrir una consulta de espacios de tablas • API sqlbftpq - Captar los datos de la consulta para filas de un espacio de tablas • API sqlbctsq - Cerrar una consulta de espacios de tablas • API sqlbtcq - Obtener los datos de la consulta para todos los contenedores de espacios de tablas • API sqlbotcq - Abrir una consulta de contenedor de espacio de tablas • API sqlbftcq - Captar los datos de la consulta para filas de un contenedor de espacio de tablas • API sqlbctcq - Cerrar una consulta de contenedor de espacio de tablas
utilapi.c, utilapi.C	• API sqlaintp - Obtener mensaje de error • API sqlogstt - Obtener el mensaje de SQLSTATE • API sqleatin - Conectar a instancia • API sqledtin - Desconectar de instancia
utilsnap.c, utilsnap.C	• API db2GetSnapshot - Obtener una instantánea del estado operativo del gestor de bases de datos • API db2MonitorSwitches - Obtener o actualizar los valores de los conmutadores del supervisor

Tabla 3. Programas de ejemplo de COBOL con las API de DB2

Programa de ejemplo	Las API incluidas
checkerr.cbl	• API sqlaintp - Obtener mensaje de error • API sqlogstt - Obtener el mensaje de SQLSTATE

Tabla 3. Programas de ejemplo de COBOL con las API de DB2 (continuación)

Programa de ejemplo	Las API incluidas
client.cbl	• API sqleqryc - Consultar valores de conexión del cliente • API sqlesetc - Establecer valores de conexión del cliente
db_udcs.cbl	• API sqleatin - Conectar a instancia • API sqlecrea - Crear base de datos • API sqledrpd - Descartar base de datos
dbcat.cbl	• API sqlecadb - Catalogar una base de datos del directorio de bases de datos del sistema • API db2DbDirCloseScan - Finalizar una exploración del directorio de bases de datos locales o del sistema • API db2DbDirGetNextEntry - Obtener la entrada siguiente del directorio de bases de datos locales o del sistema • API db2DbDirOpenScan - Iniciar una exploración del directorio de bases de datos locales o del sistema • API sqleuncd - Descatalogar una base de datos del directorio de bases de datos del sistema
dbcmt.cbl	• API sqledcgd - Cambiar un comentario de base de datos en el directorio de bases de datos locales o del sistema • API db2DbDirCloseScan - Finalizar una exploración del directorio de bases de datos locales o del sistema • API db2DbDirGetNextEntry - Obtener la entrada siguiente del directorio de bases de datos locales o del sistema • API db2DbDirOpenScan - Iniciar una exploración del directorio de bases de datos locales o del sistema • API sqleisig - Instalar manejador de señales
dbinst.cbl	• API sqleatcp - Conectar a instancia y cambiar contraseña • API sqleatin - Conectar a instancia • API sqledtin - Desconectar de instancia • API sqlegins - Obtener instancia actual
dbstat.sqb	• API db2Reorg - Reorganizar un índice o una tabla • API db2Runstats - Actualizar estadísticas sobre las características de una tabla e índices asociados
dcscat.cbl	• API sqlegdad - Catalogar una base de datos en el directorio de DCS (servicios de conexión de base de datos) • API sqlegdcl - Finalizar una exploración del directorio de DCS (servicios de conexión de base de datos) • API sqlegdel - Descatalogar una base de datos del directorio de DCS (servicios de conexión de base de datos) • API sqlegdge - Obtener una entrada específica del directorio de DCS (servicios de conexión de base de datos) • API sqlegdgt - Obtener entradas del directorio de DCS (servicios de conexión de base de datos) • API sqlegdcl - Iniciar una exploración del directorio de DCS (servicios de conexión de base de datos)

Tabla 3. Programas de ejemplo de COBOL con las API de DB2 (continuación)

Programa de ejemplo	Las API incluidas
ebcdicdb.cbl	• API sqleatin - Conectar a instancia • API sqlecrea - Crear base de datos • API sqledrpd - Descartar base de datos
expsamp.sqb	• API db2Export - Exportar datos de una base de datos • API db2Import - Importar datos a una tabla, jerarquía, apodo o vista
impexp.sqb	• API db2Export - Exportar datos de una base de datos • API db2Import - Importar datos a una tabla, jerarquía, apodo o vista
loadqry.sqb	• API db2LoadQuery - Obtener el estado de una operación de carga
migrate.cbl	• API sqlemgdb - Migrar la versión anterior de la base de datos DB2 a la versión actual
nodecat.cbl	• API sqlectnd - Catalogar una entrada en el directorio de nodos • API sqlencls - Finalizar una exploración del directorio de nodos • API sqlengne - Obtener la entrada siguiente del directorio de nodos • API sqlenops - Iniciar una exploración del directorio de nodos • API sqleuncn - Descatalogar una entrada del directorio de nodos
rebind.sqb	• API sqlarbnd - Volver a vincular paquete
tabscont.sqb	• API sqlbctcq - Cerrar una consulta de contenedor de espacio de tablas • API sqlbftcq - Captar los datos de la consulta para filas de un contenedor de espacio de tablas • API sqlbotcq - Abrir una consulta de contenedor de espacio de tablas • API sqlbtcq - Obtener los datos de la consulta para todos los contenedores de espacios de tablas • API sqlefmem - Liberar la memoria asignada por las API sqlbtcq y sqlbmtsq
tabspace.sqb	• API sqlbctsq - Cerrar una consulta de espacios de tablas • API sqlbftpq - Captar los datos de la consulta para filas de un espacio de tablas • API sqlbgtss - Obtener estadísticas de utilización del espacio de tablas • API sqlbmtsq - Obtener los datos de la consulta para todos los espacios de tabla • API sqlbotsq - Abrir una consulta de espacios de tablas • API sqlbstpq - Obtener información sobre un espacio de tablas individual • API sqlefmem - Liberar la memoria asignada por las API sqlbtcq y sqlbmtsq
tload.sqb	• API db2Export - Exportar datos de una base de datos • API sqluvqdp - Inmovilizar espacios de tablas para una tabla

Tabla 3. Programas de ejemplo de COBOL con las API de DB2 (continuación)

Programa de ejemplo	**Las API incluidas**
tspace.sqb	• API sqlbctcq - Cerrar una consulta de contenedor de espacio de tablas • API sqlbctsq - Cerrar una consulta de espacios de tablas • API sqlbftcq - Captar los datos de la consulta para filas de un contenedor de espacio de tablas • API sqlbftpq - Captar los datos de la consulta para filas de un espacio de tablas • API sqlbgtss - Obtener estadísticas de utilización del espacio de tablas • API sqlbmtsq - Obtener los datos de la consulta para todos los espacios de tabla • API sqlbotcq - Abrir una consulta de contenedor de espacio de tablas • API sqlbotsq - Abrir una consulta de espacios de tablas • API sqlbstpq - Obtener información sobre un espacio de tablas individual • API sqlbstsc - Definir contenedores de espacios de tablas • API sqlbtcq - Obtener los datos de la consulta para todos los contenedores de espacios de tablas • API sqlefmem - Liberar la memoria asignada por las API sqlbtcq y sqlbmtsq
setact.cbl	• API sqlesact - Establecer serie de contabilidad

Capítulo 2. Cambios en las API y estructuras de datos

Tabla 4. Interfaces API y estructuras de datos de nivel anterior que siguen siendo válidas

API o estructura de datos (Versión)	Nombre descriptivo	Nueva API o estructura de datos (Versión)
sqlbftsq (V2)	Recuperar consulta de espacios de tablas	sqlbftpq (V5)
sqlbstsq (V2)	Consulta de espacio de tablas simple	sqlbstpq (V5)
sqlbtsq (V2)	Consulta de espacios de tablas	sqlbmtsq (V5)
sqlectdd (V2)	Catalogar base de datos	sqlecadb (V5)
sqledosd (V8.1)	Abrir exploración del directorio de bases de datos	db2DbDirOpenScan (V8.2)
sqledgne (V8.1)	Obtener entrada siguiente del directorio de bases de datos	db2DbDirGetNextEntry (V8.2)
sqledcls (V8.1)	Cerrar exploración de directorio de bases de datos	db2DbDirCloseScan (V8.2)
sqlepstart (V5)	Iniciar gestor de bases de datos	db2InstanceStart (V8)
sqlepstp (V5)	Detener gestor de bases de datos	db2InstanceStop (V8)
sqlepstr (V2)	Iniciar el gestor de bases de datos (DB2 Parallel Edition Versión 1.2)	db2InstanceStart (V8)
sqlestar (V2)	Iniciar gestor de bases de datos (DB2 Versión 2)	db2InstanceStart (V8)
sqlestop (V2)	Detener gestor de bases de datos	db2InstanceStop (V8)
sqlerstd (V5)	Reiniciar base de datos	db2DatabaseRestart (V6)
sqlfddb (V7)	Obtener valores por omisión de la configuración de la base de datos	db2CfgGet (V8)
sqlfdsys (V7)	Obtener valores por omisión de la configuración del gestor de bases de datos	db2CfgGet (V8)
sqlfrdb (V7)	Restaurar configuración de la base de datos	db2CfgSet (V8)
sqlfrsys (V7)	Restaurar configuración del gestor de bases de datos	db2CfgSet (V8)
sqlfudb (V7)	Actualizar configuración de la base de datos	db2CfgSet (V8)
sqlfusys (V7)	Actualizar configuración del gestor de bases de datos	db2CfgSet (V8)
sqlfxdb (V7)	Obtener configuración de la base de datos	db2CfgGet (V8)
sqlfxsys (V7)	Obtener configuración de la base de datos	db2CfgGet (V8)
sqlmon (V6)	Obtener/actualizar conmutadores de supervisor	db2MonitorSwitches (V7)
sqlmonss (V5)	Obtener instantánea	db2GetSnapshot (V6)

Tabla 4. Interfaces API y estructuras de datos de nivel anterior que siguen siendo válidas (continuación)

API o estructura de datos (Versión)	Nombre descriptivo	Nueva API o estructura de datos (Versión)
sqlmonsz (V6)	Calcular tamaño necesario para almacenamiento intermedio de salida de sqlmonss()	db2GetSnapshotSize (V7)
sqlmrset (V6)	Restaurar supervisor	db2ResetMonitor (V7)
sqluadau (V8)	Obtener autorizaciones	Función de tabla AUTH_LIST_AUTHORITIES_FOR_AUTHID (V9.5)
sqlubkp (V5)	Hacer copia de seguridad de base de datos	db2Backup (V8)
sqlubkup (V2)	Hacer copia de seguridad de base de datos	db2Backup (V8)
sqluexpr	Exportar	db2Export (V8)
sqlugrpi (V2)	Obtener información sobre el particionamiento de filas (DB2 Parallel Edition Versión 1.x)	sqlugrpn (V5)
sqluhcls (V5)	Cerrar exploración del archivo histórico de recuperación	db2HistoryCloseScan (V6)
sqluhget (V5)	Recuperar información sobre DDL del archivo histórico	db2HistoryGetEntry (V6)
sqluhgne (V5)	Obtener entrada siguiente del archivo histórico de recuperación	db2HistoryGetEntry (V6)
sqluhops (V5)	Abrir exploración del archivo histórico de recuperación	db2HistoryOpenScan (V6)
sqluhprn (V5)	Eliminar archivo histórico de recuperación	db2Prune (V6)
sqluhupd (V5)	Actualizar archivo histórico de recuperación	db2HistoryUpdate (V6)
sqluimpr	Importar	db2Import (V8)
sqluload (V7)	Cargar	db2Load (V8)
sqluqry (V5)	Cargar consulta	db2LoadQuery (V6)
sqlureot (V7)	Reorganizar tabla	db2Reorg (V8)
sqlurestore (V7)	Restaurar base de datos	db2Restore (V8)
sqlurlog (V7)	Archivo de anotaciones de lectura asíncrona	db2ReadLog (V8)
sqluroll (V7)	Avanzar base de datos	db2Rollforward (V8)
sqlursto (V2)	Restaurar base de datos	sqlurst (V5)
sqlustat (V7)	Ejecutar estadísticas	db2Runstats (V8)
sqlxhcom (V2)	Confirmar transacción dudosa	sqlxphcm (V5)
sqlxhqry (V2)	Listar transacciones dudosas	sqlxphqr (V5)
sqlxhrol (V2)	Retrotraer transacción dudosa	sqlxphrl (V5)
SQL-AUTHORIZATIONS (V8)	Estructura de las autorizaciones	none
SQLB-TBSQRY-DATA (V2)	Estructura de datos del espacio de tablas.	SQLB-TBSPQRY-DATA (V5)

Tabla 4. Interfaces API y estructuras de datos de nivel anterior que siguen siendo válidas (continuación)

API o estructura de datos (Versión)	Nombre descriptivo	Nueva API o estructura de datos (Versión)
SQLE-START-OPTIONS (V7)	Iniciar estructura de datos del gestor de bases de datos	db2StartOptionsStruct (V8)
SQLEDBSTOPOPT (V7)	Iniciar estructura de datos del gestor de bases de datos	db2StopOptionsStruct (V8)
SQLEDBSTRTOPT (V2)	Iniciar estructura de datos del gestor de bases de datos (DB2 Parallel Edition Versión 1.2)	db2StartOptionsStruct (V8)
SQLEDINFO (v8.1)	Obtener estructura de datos siguiente para la entrada del directorio de bases de datos	db2DbDirInfo (V8.2)
SQLUEXPT-OUT	Exportar estructura de salida	db2ExportOut (V8.2)
SQLUHINFO y SQLUHADM (V5)	Estructuras de datos del archivo histórico	db2HistData (V6)
SQLUIMPT-IN	Importar estructura de entrada	db2ImportIn (V8.2)
SQLUIMPT-OUT	Importar estructura de salida	db2ImportOut (V8.2)
SQLULOAD-IN (V7)	Cargar estructura de entrada	db2LoadIn (V8)
SQLULOAD-OUT (V7)	Cargar estructura de salida	db2LoadOut (V8)
db2DbDirInfo (V8.2)	Obtener estructura de datos siguiente para la entrada del directorio de bases de datos	db2DbDirInfoV9 (V9.1)
db2DbDirNextEntryStruct (V8.2)	Obtener estructura de datos siguiente para la entrada del directorio de bases de datos	db2DbDirNextEntryStructV9 (V9.1)
db2gDbDirNextEntryStruct (V8.2)	Obtener estructura de datos siguiente para la entrada del directorio de bases de datos	db2gDbDirNextEntryStrV9 (V9.1)

Tabla 5. Interfaces API y estructuras de datos de nivel anterior que ya no son válidas

Nombre	Nombre descriptivo	API o estructura de datos permitida en V9
sqlufrol/sqlgfrol	Avance de base de datos (DB2 Versión 1.1)	db2Rollforward
sqluprfw	Avance de bases de datos (DB2 Parallel Edition Versión 1.x)	db2Rollforward
sqlurfwd/sqlgrfwd	Avanzar base de datos (DB2 Versión 1.2)	db2Rollforward
sqlurllf/sqlgrfwd	Avance de base de datos (DB2 Versión 2)	db2Rollforward
sqlxphqr	Listar transacción dudosa	db2XaListIndTrans
SQLXA-RECOVER	Estructura de la API de transacción	db2XaRecoverStruct

Capítulo 3. Cómo están organizadas las descripciones de las API

Algunas o todas las secciones siguientes están precedidas por una breve descripción de cada API.

Ámbito

Ámbito de operación de la API dentro de la instancia. En un entorno de bases de datos de una sola partición, el ámbito es esa partición de base de datos solamente. En un entorno de bases de datos particionadas, el ámbito puede ser el conjunto de todos los servidores de particiones lógicas de base de datos definidos en el archivo de configuración de nodos (`db2nodes.cfg`) o la partición de base de datos desde la que se invoca la API.

Autorización

Autorización necesaria para invocar la API satisfactoriamente.

Conexión necesaria

Una de las siguientes: base de datos, instancia, ninguna o establece una conexión. Indica si la función necesita una conexión de base de datos, una conexión de instancia o ninguna conexión para operar satisfactoriamente.

Ninguna significa que no es necesaria ninguna conexión de base de datos para que la API trabaje satisfactoriamente. *Establece una conexión* significa que la API establecerá una conexión con la base de datos cuando se invoque la API.

Puede ser necesaria una conexión explícita con la base de datos o una conexión con la instancia para poder invocar una API determinada. Las API que necesitan una conexión de base de datos o una conexión de instancia se pueden ejecutar de forma local o remota. Las API que no necesitan ninguna de las dos conexiones no se pueden ejecutar de forma remota; cuando estas API se invocan desde el cliente, afectan al entorno del cliente solamente.

Archivo de inclusión de la API

Nombre del archivo de inclusión donde están contenidos el prototipo de API, y las constantes y parámetros predefinidos que sean necesarios.

Nota: Las extensiones de los archivos de inclusión varían según el lenguaje de programación. Los archivos de inclusión de C/C++ tienen la extensión de archivo `.h`. Los archivos de inclusión de COBOL tienen la extensión de archivo `.cbl`. Los archivos de inclusión se pueden encontrar en los directorios siguientes:

C/C++ (UNIX):
```
sqllib/include
```

C/C++ (Windows):
```
sqllib\include
```

COBOL (UNIX):
```
sqllib/include/cobol_a
```

```
sqllib/include/cobol_i

sqllib/include/cobol_mf
```

COBOL (Windows):

```
sqllib\include\cobol_a

sqllib\include\cobol_i

sqllib\include\cobol_mf
```

Sintaxis de la API para C

Sintaxis de la llamada a la API para el lenguaje C.

A partir de la Versión 6, se aplica un nuevo estándar a las API administrativas de DB2. La aplicación de las nuevas definiciones de API se realiza de forma gradual. A continuación sigue una breve visión general de los cambios:

- Los nuevos nombres de las API contienen el prefijo ″db2″, seguido de una serie, en mayúsculas y minúsculas, que sea descriptiva (por ejemplo, db2LoadQuery). Las API que son afines entre sí tienen nombres que permiten agruparlas según criterios lógicos. Por ejemplo:

  ```
  db2HistoryCloseScan
  db2HistoryGetEntry
  db2HistoryOpenScan
  db2HistoryUpdate
  ```

- Las API genéricas tienen nombres que contienen el prefijo ″db2g″, seguido de una serie que coincide con el nombre de la API para C. Las estructuras de datos utilizadas por las API genéricas tienen nombres que también contienen el prefijo ″db2g″.
- El primer parámetro dentro de la función, el número de versión, (*númeroVersión*) representa la versión, el release o nivel de PTF para el que se debe compilar el código. Este número de versión se utiliza para especificar el nivel de la estructura que se pasa como segundo parámetro.
- El segundo parámetro dentro de la función es un puntero nulo que apunta a la estructura de interfaz primaria de la API. Cada elemento de la estructura es un tipo atómico (por ejemplo, db2Long32) o un puntero. Cada nombre de parámetro se ajusta a los siguientes convenios de denominación:

  ```
  piCamelCase  - puntero para datos de entrada
  poCamelCase  - puntero para datos de salida
  pioCamelCase - puntero para datos de entrada o de salida
  iCamelCase   - datos de entrada
  ioCamelCase  - datos de entrada/salida
  oCamelCase   - datos de salida
  ```

- El tercer parámetro es un puntero a la SQLCA, y es obligatorio.

Sintaxis de la API genérica

Sintaxis de la llamada a la API para los lenguajes de programación COBOL y FORTRAN.

Atención: Proporcione un byte adicional para cada serie transferida a una API; de lo contrario, se pueden producir errores inesperados. Este byte adicional es modificado por el gestor de bases de datos.

Parámetros de la API

Descripción de cada parámetro de la API y de sus valores. Aparecen listados los valores predefinidos junto con los símbolos apropiados. Los valores reales de los símbolos se pueden obtener a partir de los archivos de inclusión del lenguaje apropiado. Los programadores de COBOL debe utilizar un guión (-) en lugar de un signo de subrayado (_) en todos los símbolos. Para obtener más información sobre los tipos de datos de parámetros en cada lenguaje principal, consulte los programas de ejemplo.

Nota: Las aplicaciones que invocan a las API del gestor de bases de datos deben comprobar debidamente si existen condiciones de error examinando códigos de retorno y la estructura SQLCA. La mayoría de las API del gestor de bases de datos devuelven el código de retorno 0 cuando se ejecutan satisfactoriamente. En general, un código de retorno distinto de cero indica que el mecanismo de manejo de errores secundarios, la estructura SQLCA, puede estar dañado. En este caso, la API llamada no se ejecuta. Una posible razón de que se dañe la estructura SQLCA es que se haya pasado una dirección no válida para la estructura.

La información de error se devuelve en los campos SQLCODE y SQLSTATE de la estructura SQLCA, que se actualiza después de la mayoría de las llamadas a las API del gestor de bases de datos. Los archivos fuente que invocan a las API del gestor de bases de datos pueden proporcionar una o más estructuras SQLCA; sus nombres son arbitrarios. Un valor de SQLCODE igual a 0 significa ejecución satisfactoria (con posibles condiciones de aviso SQLWARN). Un valor positivo significa que la sentencia se ha ejecutado satisfactoriamente, pero con un aviso, tal como el truncamiento de una variable de lenguaje principal. Un valor negativo significa que se ha producido una condición de error.

Un campo adicional, SQLSTATE, contiene un código de error estandarizado que es coherente con otros productos de bases de datos IBM y con gestores de bases de datos compatibles con SQL92. Utilice los campos SQLSTATE cuando sea necesario asegurar la portabilidad, pues los SQLSTATE son comunes a muchos gestores de bases de datos.

El campo SQLWARN contiene una matriz de indicadores de aviso, incluso si SQLCODE es cero.

Sintaxis de la API de REXX

Sintaxis para REXX de la llamada a la API, cuando sea apropiado.

La interfaz SQLDB2 permite la invocación de una API desde REXX. La interfaz SQLDB2 se ha creado para dar soporte en REXX a nuevas API o las API que no estaban soportadas anteriormente cuya única información de salida es la SQLCA. Invocar un mandato mediante la interfaz SQLDB2 es sintácticamente lo mismo que invocar el mandato mediante el procesador de línea de mandatos (CLP), excepto que el símbolo `call db2` se sustituye por `CALL SQLDB2`. Utilizar `CALL SQLDB2` desde REXX tiene las ventajas siguientes respecto a llamar directamente al CLP:

- La variable compuesta SQLCA de REXX está definida
- Por omisión, todos los mensajes de salida del CLP están desactivados.

Parámetros de la API de REXX

Descripción de cada parámetro de la API de REXX y de sus valores, cuando sea apropiado.

Archivos de inclusión para aplicaciones de las API de DB2

A continuación se describen los archivos de inclusión que están pensados para ser utilizados en aplicaciones C, C++, COBOL y FORTRAN para invocar las API de DB2:

Archivos de inclusión para C y C++

DB2APIDF (db2ApiDf.h)
Este archivo define estructuras, constantes y prototipos para casi todas las API de DB2 cuyos nombres comiencen por 'db2'.

DB2AUCFG (db2AuCfg.h)
Este archivo define estructuras, constantes y prototipos para las API de DB2, db2AutoConfig y db2AutoConfigFreeMemory.

DB2SECPLUGIN (db2secPlugin.h)
Este archivo define estructuras, constantes y prototipos para las API utilizadas para desarrollar plugins de seguridad personalizados con fines de autentificación y búsqueda de miembros de grupos.

SQL (sql.h)
Este archivo incluye prototipos específicos del lenguaje para el vinculador, el precompilador y las API de recuperación de mensajes de error. También define constantes del sistema.

SQLAPREP (sqlaprep.h)
Este archivo contiene definiciones necesarias para escribir su propio precompilador.

SQLENV (sqlenv.h)
Este archivo define llamadas específicas del lenguaje para las API del entorno de bases de datos y las estructuras, constantes y códigos de retorno correspondientes a dichas interfaces.

SQLMON (sqlmon.h)
Este archivo define llamadas específicas del lenguaje para las API del supervisor del sistema de bases de datos y las estructuras, constantes y códigos de retorno correspondientes a dichas interfaces.

SQLUTIL (sqlutil.h)
Este archivo define las llamadas específicas del lenguaje correspondientes a las API de programas de utilidad y las estructuras, constantes y códigos necesarios para dichas interfaces.

SQLUVEND (sqluvend.h)
Este archivo define estructuras, constantes y prototipos correspondientes a las API que utilizarán los proveedores de gestión de almacenamiento.

SQLXA (sqlxa.h)
Este archivo contiene prototipos y constantes de funciones utilizados por las aplicaciones que usan la Interfaz XA de X/Open.

Archivos de inclusión para COBOL

SQL (sql.cbl)
Este archivo incluye prototipos específicos del lenguaje para el vinculador, el precompilador y las API de recuperación de mensajes de error. También define constantes del sistema.

SQLAPREP (sqlaprep.cbl)
Este archivo contiene definiciones necesarias para escribir su propio precompilador.

SQLENV (sqlenv.cbl)
Este archivo define llamadas específicas del lenguaje para las API del entorno de bases de datos y las estructuras, constantes y códigos de retorno correspondientes a dichas interfaces.

SQLMON (sqlmon.cbl)
Este archivo define llamadas específicas del lenguaje para las API del supervisor del sistema de bases de datos y las estructuras, constantes y códigos de retorno correspondientes a dichas interfaces.

SQLMONCT (sqlmonct.cbl)
Este archivo contiene definiciones de constantes y definiciones de estructuras de datos locales necesarias para llamar a las API del Supervisor del sistema de bases de datos.

SQLUTIL (sqlutil.cbl)
Este archivo define las llamadas específicas del lenguaje correspondientes a las API de programas de utilidad y las estructuras, constantes y códigos necesarios para dichas interfaces.

Archivos de inclusión para FORTRAN

SQL (sql.f)
Este archivo incluye prototipos específicos del lenguaje para el vinculador, el precompilador y las API de recuperación de mensajes de error. También define constantes del sistema.

SQLAPREP (sqlaprep.f)
Este archivo contiene definiciones necesarias para escribir su propio precompilador.

SQLENV (sqlenv.f)
Este archivo define llamadas específicas del lenguaje para las API del entorno de bases de datos y las estructuras, constantes y códigos de retorno correspondientes a dichas interfaces.

SQLMON (sqlmon.f)
Este archivo define llamadas específicas del lenguaje para las API del supervisor del sistema de bases de datos y las estructuras, constantes y códigos de retorno correspondientes a dichas interfaces.

SQLUTIL (sqlutil.f)
Este archivo define las llamadas específicas del lenguaje correspondientes a las API de programas de utilidad y las estructuras, constantes y códigos necesarios para dichas interfaces.

Capítulo 4. Las API administrativas

db2AddContact - Añadir un contacto al que se pueden enviar mensajes de notificación

Añade un contacto a la lista de contactos. Los contactos son usuarios a los que se pueden enviar mensajes de notificación. Los contactos se pueden definir localmente en el sistema o en una lista global. El valor del parámetro de configuración del Servidor de administración de DB2 (DAS), contact_host, determina si la lista es local o global.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2AddContact (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2AddContactData
{
   char *piUserid;
   char *piPassword;
   char *piName;
   db2Uint32 iType;
   char *piAddress;
   db2Uint32 iMaxPageLength;
   char *piDescription;
} db2AddContactData;
```

Parámetros de la API db2AddContact

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2AddContactData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2AddContactData

piUserid
Entrada. Nombre del usuario.

piPassword
Entrada. Contraseña del ID de usuario especificado por el parámetro piUserid.

piName
Entrada. Nombre del contacto.

iType Entrada. Especifica el tipo de contacto. Los valores válidos son:
- DB2CONTACT_EMAIL
- DB2CONTACT_PAGE

piAddress
Entrada. Dirección de correo electrónico o buscapersonas del parámetro iType.

iMaxPageLength
Entrada. Longitud máxima del mensaje cuando el valor de iType es DB2CONTACT_PAGE.

piDescription
Entrada. Descripción del contacto proporcionada por el usuario.

Notas de uso

Esta API no está soportada en UNIX y Linux. No obstante, podrá acceder a la misma función utilizando la interfaz de SQL.

db2AddContactGroup - Añadir un grupo de contactos al que se pueden enviar mensajes de notificación

Añade un nuevo grupo de contactos a la lista de grupos de contacto. Un grupo de contactos contiene una lista de usuarios a los que se pueden enviar mensajes de notificación. Los grupos de contactos se pueden definir localmente en el sistema o en una lista global. El valor del parámetro de configuración contact_host del Servidor de administración de DB2 (DAS) determina si la lista es local o global.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2AddContactGroup (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2AddContactGroupData
{
   char *piUserid;
   char *piPassword;
```

```
   char *piGroupName;
   char *piDescription;
   db2Uint32 iNumContacts;
   struct db2ContactTypeData *piContacts;
} db2AddContactGroupData;

typedef SQL_STRUCTURE db2ContactTypeData
{
   db2Uint32 contactType;
   char *pName;
} db2ContactTypeData;
```

Parámetros de la API db2AddContactGroup

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2AddContactGroupData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2AddContactGroupData

piUserid
Entrada. Nombre del usuario.

piPassword
Entrada. Contraseña de piUserid.

piGroupName
Entrada. Nombre del grupo que se debe recuperar.

piDescription
Entrada. Descripción del grupo.

iNumContacts
Entrada. Número de piContacts.

piContacts
Puntero a la estructura db2ContactTypeData.

Parámetros de la estructura de datos db2ContactTypeData

contactType
Especifica el tipo de contacto. Los valores válidos son:

- DB2CONTACT_SINGLE
- DB2CONTACT_GROUP

pName
Nombre del grupo de contactos, o nombre del contacto si contactType es DB2CONTACT_SINGLE.

Notas de uso

Esta API no está soportada en UNIX y Linux. No obstante, podrá acceder a la misma función utilizando la interfaz de SQL.

db2AddSnapshotRequest - Añadir una petición de instantánea

Esta API prepara la corriente de petición de instantánea para db2GetSnapshotSize y db2GetSnapshot.

Ámbito

Prepara la corriente de petición de instantánea para las API db2GetSnapshotSize y db2GetSnapshot. Los datos de salida (una petición de instantánea generada por la API db2AddSnapshotRequest) se pasan a las API db2GetSnapshotSize y db2GetSnapshot. Cada petición de instantánea contiene el tipo de petición y la información de identificación.

Autorización

Ninguna.

Conexión necesaria

Ninguna.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2AddSnapshotRequest (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2AddSnapshotRqstData
{
   void *pioRequestData;
   db2Uint32 iRequestType;
   db2int32 iRequestFlags;
   db2Uint32 iQualType;
   void *piQualData;
} db2AddSnapshotRqstData;

SQL_API_RC SQL_API_FN
  db2gAddSnapshotRequest (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gAddSnapshotRqstData
{
   void *pioRequestData;
   db2Uint32 iRequestType;
   db2int32 iRequestFlags;
   db2Uint32 iQualType;
   void *piQualData;
   db2Uint32 iQualDataLen;
} db2gAddSnapshotRqstData;
```

Parámetros de la API db2AddSnapshotRequest

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura

transferida como segundo parámetro pParmStruct. Para utilizar la estructura db2AddSnapshotData tal como se describe más arriba, especifique db2Versio910. Si desea utilizar una versión diferente de esta estructura, vea la lista completa de versiones soportadas en el archivo de cabecera db2ApiDf, en el directorio de include. Debe utilizar la versión de la estructura db2AddSnapshotRequestData correspondiente al número de versión que especifique.

pParmStruct
Entrada y/o salida. Puntero a la estructura db2AddSnapshotRequestData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2AddSnapshotRqstData

pioRequestData
Entrada/salida. Datos de petición que la API db2AddSnapshotRequest debe construir. Inicialmente, el valor de esta parámetro es NULL. La memoria necesaria para pioRequestData se asignará mediante la API db2AddSnapshotRequest. Hay que liberar pioRequestData al final de su utilización (por ejemplo, después de la llamada a la API db2GetSnapshot).

iRequestType
Entrada. Tipo de petición de instantánea; por ejemplo, SQLMA_DB2.

iRequestFlags
Entrada. Distintivos de acción con mapa de bits; los valores son SQLM_INSTREAM_ADD_REQUEST, SQLM_INSTREAM_ADD_QUAL o SQLM_INSTREAM_ADD_REQQUAL. Si el llamador no establece iRequestFlags:

- si se establece iRequestType, la API activa el bit SQLM_INSTREAM_ADD_REQUEST de iRequestFlags.
- si el puntero piQualifierData no es nulo, la API activa SQLM_INSTREAM_ADD_QUAL.

Con la llamada a la API, iRequestType, iQualifierType, iRequestFlags y piQualifierData se restablecen igual a 0.

iQualType
Entrada. Tipo del calificador unido a la petición de instantánea; por ejemplo, SQLM_INSTREAM_ELM_DBNAME.

piQualData
Entrada. Datos que describen el calificador. En un puntero a una serie terminada en nulo.

Parámetros específicos de la estructura de datos db2gAddSnapshotRqstData

iQualDataLen
Entrada. Longitud de los datos del calificador en el parámetro piQualData.

db2AdminMsgWrite - Escribir mensajes de anotación cronológica para la función de administración y duplicación

Proporciona un mecanismo para que los usuarios y la función de Duplicación escriban información en el archivo db2diag.log, y en el archivo de registro de notificaciones de administración.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2AdminMsgWrite (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef struct db2AdminMsgWriteStruct
{
   db2Uint32 iMsgType;
   db2Uint32 iComponent;
   db2Uint32 iFunction;
   db2Uint32 iProbeID;
   char *piData_title;
   void *piData;
   db2Uint32 iDataLen;
   db2Uint32 iError_type;
} db2AdminMsgWriteStruct;
```

Parámetros de la API db2AdminMsgWrite

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2AdminMsgWriteStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2AdminMsgWriteStruct

iMsgType
Entrada. Especifique el tipo de datos que se deben registrar. Los valores válidos son BINARY_MSG para datos binarios y STRING_MSG para datos de tipo serie.

iComponent
Entrada. Especifique cero.

iFunction
Entrada. Especifique cero.

iProbeID
Entrada. Especifique el punto de sondeo numérico. El punto de sondeo numérico es un identificador interno exclusivo que se utiliza para localizar el punto del código fuente que notificó el mensaje.

piData_title
Entrada. Puntero a la serie de título que describe los datos que se van a registrar cronológicamente. Su valor se puede establecer en NULL si no es necesario un título.

piData
Entrada. Puntero a los datos que se van a registrar cronológicamente. Su valor se puede establecer en NULL si no es necesario registrar anotaciones para los datos.

iDataLen
Entrada. Número de bytes de datos binarios que se deben utilizar para el registro de anotaciones si iMsgType es BINARY_MSG. No se utiliza si iMsgType es STRING_MSG.

iError_type
Entrada. Los valores válidos son:

- `DB2LOG_SEVERE_ERROR`: (1) Se ha producido un error grave
- `DB2LOG_ERROR`: (2) Se ha producido un error
- `DB2LOG_WARNING`: (3) Se ha producido un aviso
- `DB2LOG_INFORMATION`: (4) Informativo

Notas de uso

Esta API registra anotaciones en el registro de notificaciones de administración solamente si el tipo de error especificado es menor o igual que el valor del parámetro de configuración notifylevel del gestor de bases de datos. La API registra anotaciones en db2diag.log solamente si el tipo de error especificado es menor o igual que el valor del parámetro de configuración diaglevel del gestor de bases de datos. Pero toda la información escrita en el registro de notificaciones de administración se duplica en db2diag.log a menos que el parámetro diaglevel se establezca en cero.

db2ArchiveLog - Archivar el archivo de anotaciones cronológicas activo

Cierra y trunca el archivo de anotaciones cronológicas activo para una base de datos recuperable. Si está habilitada una salida de usuario, también emite una petición de archivado.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint
- dbadm

Conexión necesaria

Esta API establece automáticamente una conexión a la base de datos especificada. Si ya existe una conexión a la base de datos especificada, la API devolverá un error.

Archivo de inclusión de la API

db2ApiDf.h

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2ArchiveLog (
       db2Uint32 versionNumber,
       void * pDB2ArchiveLogStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ArchiveLogStruct
{
   char *piDatabaseAlias;
   char *piUserName;
   char *piPassword;
   db2Uint16 iAllNodeFlag;
   db2Uint16 iNumNodes;
   SQL_PDB_NODE_TYPE *piNodeList;
   db2Uint32 iOptions;
} db2ArchiveLogStruct;

SQL_API_RC SQL_API_FN
  db2gArchiveLog (
       db2Uint32 versionNumber,
       void * pDB2ArchiveLogStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gArchiveLogStruct
{
   db2Uint32 iAliasLen;
   db2Uint32 iUserNameLen;
   db2Uint32 iPasswordLen;
   char *piDatabaseAlias;
   char *piUserName;
   char *piPassword;
   db2Uint16 iAllNodeFlag;
   db2Uint16 iNumNodes;
   SQL_PDB_NODE_TYPE *piNodeList;
   db2Uint32 iOptions;
} db2gArchiveLogStruct;
```

Parámetros de la API db2ArchiveLog

versionNumber
Entrada. Especifica la versión y el nivel de release de la variable transferida como segundo parámetro, pDB2ArchiveLogStruct.

pDB2ArchiveLogStruct
Entrada. Puntero a la estructura db2ArchiveLogStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2ArchiveLogStruct

piDatabaseAlias
Entrada. Serie que contiene el alias (tal como está catalogado en el directorio de bases de datos del sistema) de la base de datos para la que se debe archivar el archivo de anotaciones activo.

piUserName
Entrada. Serie que contiene el nombre de usuario que se debe utilizar al intentar establecer una conexión.

piPassword
Entrada. Serie que contiene la contraseña que se debe utilizar al intentar establecer una conexión.

iAllNodeFlag
Sólo aplicable a un entorno de bases de datos particionadas. Entrada. Distintivo que indica si la operación se debe aplicar a todos los nodos listados en el archivo db2nodes.cfg. Los valores válidos son:

DB2ARCHIVELOG_NODE_LIST
Aplicar a los nodos contenidos en la lista de nodos que se proporciona en piNodeList.

DB2ARCHIVELOG_ALL_NODES
Aplicar a todos los nodos. piNodeList debe ser NULL. Es el valor por omisión.

DB2ARCHIVELOG_ALL_EXCEPT
Aplicar a todos los nodos excepto los contenidos en la lista de nodos que se proporciona en piNodeList.

iNumNodes
Sólo para entornos de bases de datos particionadas. Entrada. Especifica el número de nodos en la matriz piNodeList.

piNodeList
Sólo para entornos de bases de datos particionadas. Entrada. Puntero a una matriz de números de nodo a los que se debe aplicar la operación de archivar archivo de anotaciones.

iOptions
Entrada. Reservado para una utilización futura.

Parámetros específicos de la estructura de datos db2gArchiveLogStruct

iAliasLen
Entrada. Número entero sin signo, de 4 bytes, que representa la longitud, expresada en bytes, del alias de base de datos.

iUserNameLen
Entrada. Número entero sin signo de 4 bytes que representa la longitud en bytes del nombre de usuario. Se establece en un valor cero si no se utiliza ningún nombre de usuario.

iPasswordLen
Entrada. Número entero sin signo, de 4 bytes, que representa la longitud en bytes de la contraseña. El valor se establece en cero si no se proporciona ninguna contraseña.

db2AutoConfig - Acceder al Asesor de configuración

Permite que los programas de aplicación accedan al Asesor de configuración en el Centro de control. La ayuda en línea del Centro de control proporciona información detallada sobre este asesor.

Ámbito

En un entorno de bases de datos particionadas, las recomendaciones del asesor para las bases de datos se aplican por omisión a todas las particiones de la base de

datos. El distintivo DB2_SG_APPLY_ON_ONE_NODE del parámetro iApply de la estructura de datos db2AutoConfigInterface hace que los cambios estén limitados solamente a la partición de coordinador. Observe que los cambios en la agrupación de almacenamientos intermedios se aplican siempre a los catálogos del sistema (DB2_SG_APPLY_ON_ONE_NODE no afecta a las recomendaciones sobre la agrupación de almacenamientos intermedios), por tanto, la API afecta a todas las particiones de base de datos.

Autorización

sysadm

Conexión necesaria

Base de datos

Archivo de inclusión de la API

db2AuCfg.h

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
db2AutoConfig(
  db2Uint32 db2VersionNumber,
  void * pAutoConfigInterface,
  struct sqlca * pSqlca);

typedef struct {
  db2int32 iProductID;
  char iProductVersion[DB2_SG_PROD_VERSION_SIZE+1];
  char iDbAlias[SQL_ALIAS_SZ+1];
  db2int32 iApply;
  db2AutoConfigInput iParams;
  db2AutoConfigOutput oResult;
} db2AutoConfigInterface;

typedef struct {
  db2int32 token;
  db2int32 value;
} db2AutoConfigElement;

typedef struct {
  db2Uint32 numElements;
  db2AutoConfigElement *  pElements;
} db2AutoConfigArray;
typedef db2AutoConfigArray  db2AutoConfigInput;
typedef db2AutoConfigArray  db2AutoConfigDiags;

typedef struct {
  db2Uint32 numElements;
  struct db2CfgParam * pConfigs;
  void * pDataArea;
} db2ConfigValues;

typedef struct {
  char * pName;
  db2int32 value;
} db2AutoConfigNameElement;

typedef struct {
  db2Uint32 numElements;
  db2AutoConfigNameElement * pElements;
} db2AutoConfigNameArray;
typedef db2AutoConfigNameArray db2BpValues;
```

```
typedef struct {
  db2ConfigValues oOldDbValues;
  db2ConfigValues oOldDbmValues;
  db2ConfigValues oNewDbValues;
  db2ConfigValues oNewDbmValues;
  db2AutoConfigDiags oDiagnostics;
  db2BpValues oOldBpValues;
  db2BpValues oNewBpValues;
} db2AutoConfigOutput;
```

Parámetros de la API db2AutoConfig

db2VersionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro, pAutoConfigInterface.

pAutoConfigInterface
Entrada. Puntero a la estructura db2AutoConfigInterface.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2AutoConfigInterface

iProductID
Entrada. Especifica un identificador de producto exclusivo. Los valores válidos para el parámetro iProductID (definidos en db2AuCfg.h, ubicado en el directorio de inclusión) son:

- DB2_SG_PID_DEFAULT
- DB2_SG_PID_WEBSPHERE_COMMERCE_SUITE
- DB2_SG_PID_SAP
- DB2_SG_PID_WEBSPHERE_ADVANCED_SERVER
- DB2_SG_PID_SIEBEL
- DB2_SG_PID_PS_EPM
- DB2_SG_PID_PS_ONLINE
- DB2_SG_PID_PS_BATCH
- DB2_SG_PID_PS
- DB2_SG_PID_LOTUS_DOMINO
- DB2_SG_PID_CONTENT_MANAGER

iProductVersion
Entrada. Serie de 16 bytes que especifica la versión del producto.

iDbAlias
Entrada. Serie que especifica un alias de base de datos.

iApply
Entrada. Actualiza automáticamente la configuración. Los valores válidos para el parámetro iApply (definidos en db2AuCfg.h, ubicado en el directorio de inclusión) son:

DB2_SG_NOT_APPLY
No aplicar ninguna recomendación

DB2_SG_APPLY
Aplicar todas las recomendaciones

DB2_SG_APPLY_DB
Aplicar solamente las recomendaciones para la base de datos (y la agrupación de almacenamientos intermedios)

DB2_SG_APPLY_ON_ONE_NODE
Aplicar las recomendaciones para la base de datos (solamente válido con DB2_SG_APPLY y DB2_SG_APPLY_DB) solamente en la partición de base de datos actual. Por omisión, las recomendaciones para la base de datos se aplican a todas las particiones.

iParams
Entrada. Pasa parámetros al asesor.

oResult
Salida. Incluye todos los resultados del asesor.

Parámetros de la estructura de datos db2AutoConfigElement

token Entrada o salida. Especifica el valor de configuración para los parámetros de entrada y los diagnósticos de salida.

value Entrada o salida. Contiene los datos especificados por token.

Parámetros de la estructura de datos db2AutoConfigArray

numElements
Entrada o salida. Número de elementos de la matriz.

pElements
Entrada o salida. Puntero a la matriz de elementos.

Parámetros de la estructura de datos db2ConfigValues

numElements
Entrada o salida. Número de elementos de la matriz.

pConfigs
Entrada o salida. Puntero a una matriz de estructura db2CfgParam.

pDataArea
Entrada o salida. Puntero al área de datos que contiene los valores de la configuración.

Parámetros de la estructura de datos db2AutoConfigNameElement

pName
Salida. Nombre de la agrupación de almacenamientos intermedios de salida.

value Entrada o salida. Contiene el tamaño (en páginas) de la agrupación de almacenamientos intermedios especificada en el nombre.

Parámetros de la estructura de datos db2AutoConfigNameArray

numElements
Entrada o salida. Número de elementos de la matriz.

pElements
Entrada o salida. Puntero a la matriz de elementos.

Parámetros de la estructura de datos db2AutoConfigOutput

oOldDbValues
Salida. Si se define el valor de iApply para actualizar la configuración de la base de datos o todas las configuraciones, este valor representa el valor de configuración de la base de datos existente antes de utilizar el asesor. En caso contrario, es el valor por omisión.

oOldDbmValues
Salida. Si se define el valor de iApply para actualizar todas las configuraciones, este valor representa el valor de configuración del gestor de bases de datos existente antes de utilizar el asesor. En caso contrario, es el valor por omisión.

oNewDbValues
Salida. Si se define el valor de iApply para actualizar la configuración de la base de datos o todas las configuraciones, este valor representa el valor de configuración actual de la base de datos. En caso contrario, es el valor recomendado para el asesor.

oNewDbmValues
Salida. Si se define el valor de iApply para actualizar todas las configuraciones, este valor representa el valor de configuración del gestor de bases de datos actual. En caso contrario, es el valor recomendado para el asesor.

oDiagnostics
Salida. Contiene diagnósticos devueltos por el asesor.

oOldBpValues
Salida. Si se define el valor de iApply para actualizar la configuración de la base de datos o todas las configuraciones, este valor representa los tamaños de la agrupación de almacenamientos intermedios, en páginas, existentes antes de utilizar el asesor. En caso contrario, es el valor actual.

oNewBpValues
Salida. Si se define el valor de iApply para actualizar la configuración de la base de datos o todas las configuraciones, este valor representa los tamaños actuales de la agrupación de almacenamientos intermedios. En caso contrario, es el valor recomendado para el asesor.

Notas de uso

Para liberar la memoria asignada por la API db2AutoConfig, invoque la API db2AutoConfigFreeMemory.

Puesto que los parámetros de configuración `maxagents` y `maxcagents` han quedado obsoletos, el comportamiento de la API db2AutoConfig dependerá del valor de db2VersionNumber que se pase a la API. Si la versión es DB2 v9.5 o anterior, no se devolverá `maxagents`, pero sí se devolverá para versiones anteriores a esta. En un futuro release, es posible que estos parámetros de configuración se eliminen por completo.

db2AutoConfigFreeMemory - Liberar la memoria asignada por la API db2AutoConfig

Libera la memoria asignada por la API db2AutoConfig.

Autorización

sysadm

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
db2AuCfg.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
db2AutoConfigFreeMemory(
  db2Uint32 db2VersionNumber,
  void * pAutoConfigInterface,
  struct sqlca * pSqlca);
```

Parámetros de la API db2AutoConfigFreeMemory

db2VersionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro, pAutoConfigInterface.

pAutoConfigInterface
Entrada. Puntero a la estructura db2AutoConfigInterface.

pSqlca
Salida. Puntero a la estructura sqlca.

db2Backup - Hacer copia de seguridad de una base de datos o un espacio de tablas

Crea una copia de seguridad de una base de datos o de un espacio de tablas.

Ámbito

En un entorno de bases de datos particionadas, por omisión esta API sólo afecta a la partición de base de datos en la que se ejecuta.

Si se especifica la opción de realizar una copia de seguridad particionada, sólo se puede llamar al mandato en el nodo de catálogo. Si la opción especifica que se debe hacer copia de seguridad de todos los servidores de particiones de base de datos, afecta a todos los servidores de particiones de base de datos listados en el archivo `db2nodes.cfg`. De lo contrario, afecta a los servidores de particiones de base de datos que se especifican en la API.

Autorización

Una de las siguientes:

- *sysadm*
- *sysctrl*
- *sysmaint*

Conexión necesaria

Base de datos. Esta API establece automáticamente una conexión a la base de datos especificada.

La conexión se interrumpe cuando finaliza la copia de seguridad.

Archivo de inclusión de la API

db2ApiDf.h

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2Backup (
       db2Uint32 versionNumber,
       void * pDB2BackupStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2BackupStruct
{
  char *piDBAlias;
  char oApplicationId[SQLU_APPLID_LEN+1];
  char oTimestamp[SQLU_TIME_STAMP_LEN+1];
  struct db2TablespaceStruct *piTablespaceList;
  struct db2MediaListStruct *piMediaList;
  char *piUsername;
  char *piPassword;
  void *piVendorOptions;
  db2Uint32 iVendorOptionsSize;
  db2Uint32 oBackupSize;
  db2Uint32 iCallerAction;
  db2Uint32 iBufferSize;
  db2Uint32 iNumBuffers;
  db2Uint32 iParallelism;
  db2Uint32 iOptions;
  db2Uint32 iUtilImpactPriority;
  char *piComprLibrary;
  void *piComprOptions;
  db2Uint32 iComprOptionsSize;
  db2int32 iAllNodeFlag;
  db2int32 iNumNodes;
  db2NodeType *piNodeList;
  db2int32 iNumMPPOutputStructs;
  struct db2BackupMPPOutputStruct *poMPPOutputStruct;
} db2BackupStruct;

typedef SQL_STRUCTURE db2TablespaceStruct
{
  char                        **tablespaces;
  db2Uint32 numTablespaces;
} db2TablespaceStruct;

typedef SQL_STRUCTURE db2MediaListStruct
{
  char                        **locations;
  db2Uint32 numLocations;
  char locationType;
} db2MediaListStruct;

typedef SQL_STRUCTURE db2BackupMPPOutputStruct
{
  db2NodeType nodeNumber;
  db2Uint64 backupSize;
  struct sqlca  sqlca;
} db2BackupMPPOutputStruct;
```

```
SQL_API_RC SQL_API_FN
  db2gBackup (
       db2Uint32 versionNumber,
       void * pDB2gBackupStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gBackupStruct
{
  char *piDBAlias;
  db2Uint32 iDBAliasLen;
  char *poApplicationId;
  db2Uint32 iApplicationIdLen;
  char *poTimestamp;
  db2Uint32 iTimestampLen;
  struct db2gTablespaceStruct *piTablespaceList;
  struct db2gMediaListStruct *piMediaList;
  char *piUsername;
  db2Uint32 iUsernameLen;
  char *piPassword;
  db2Uint32 iPasswordLen;
  void *piVendorOptions;
  db2Uint32 iVendorOptionsSize;
  db2Uint32 oBackupSize;
  db2Uint32 iCallerAction;
  db2Uint32 iBufferSize;
  db2Uint32 iNumBuffers;
  db2Uint32 iParallelism;
  db2Uint32 iOptions;
  db2Uint32 iUtilImpactPriority;
  char *piComprLibrary;
  db2Uint32 iComprLibraryLen;
  void *piComprOptions;
  db2Uint32 iComprOptionsSize;
  db2int32 iAllNodeFlag;
  db2int32 iNumNodes;
  db2NodeType *piNodeList;
  db2int32 iNumMPPOutputStructs;
  struct db2gBackupMPPOutputStruct *poMPPOutputStruct;
} db2gBackupStruct;

typedef SQL_STRUCTURE db2gTablespaceStruct
{
  struct db2Char *tablespaces;
  db2Uint32 numTablespaces;
} db2gTablespaceStruct;

typedef SQL_STRUCTURE db2gMediaListStruct
{
  struct db2Char *locations;
  db2Uint32 numLocations;
  char locationType;
} db2gMediaListStruct;

typedef SQL_STRUCTURE db2gBackupMPPOutputStruct
{
  db2NodeType nodeNumber;
  db2Uint64 backupSize;
  struct sqlca  sqlca;
} db2gBackupMPPOutputStruct;

typedef SQL_STRUCTURE db2Char
{
   char *pioData;
   db2Uint32 iLength;
   db2Uint32 oLength;
} db2Char;
```

Parámetros de la API db2Backup

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro **pDB2BackupStruct**.

pDB2BackupStruct
Entrada. Puntero a la estructura db2BackupStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2BackupStruct

piDBAlias
Entrada. Serie que contiene el alias de base de datos (tal como está catalogado en el directorio de bases de datos del sistema) de la base de datos que se debe copiar.

oApplicationId
Salida. La API devuelve una serie que identifica al agente que presta servicio a la aplicación. Se puede utilizar para obtener información sobre el progreso de la operación de copia de seguridad utilizando el supervisor de bases de datos.

oTimestamp
Salida. La API devolverá la indicación de fecha y hora de la imagen de copia de seguridad.

piTablespaceList
Entrada. Lista de espacios de tablas de los que debe hacerse copia de seguridad. Necesario solamente para la copia de seguridad de nivel de espacios de tablas. Debe ser NULL para una copia de seguridad de nivel de base de datos. Vea la estructura db2TablespaceStruct.

piMediaList
Entrada. Esta estructura permite al llamador especificar el destino de la operación de copia de seguridad. Para obtener más información, consulte la estructura db2MediaListStruct.

piUsername
Entrada. Serie que contiene el nombre de usuario que se debe utilizar al intentar establecer una conexión. Puede ser NULL.

piPassword
Entrada. Serie que contiene la contraseña que se debe utilizar con el nombre de usuario. Puede ser NULL.

piVendorOptions
Entrada. Se utiliza para pasar información desde la aplicación a las funciones de proveedor. Esta estructura de datos debe ser plana; es decir, no se admite ningún nivel de direccionamiento indirecto. Tenga en cuenta que no se realiza la inversión de bytes y no se buscan estos datos en la página de códigos.

iVendorOptionsSize
Entrada. Longitud del campo **piVendorOptions**, que no puede tener más de 65535 bytes.

oBackupSize
Salida. Tamaño de la imagen de copia de seguridad (en MB).

iCallerAction

Entrada. Especifica la acción que se debe realizar. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2BACKUP_BACKUP
Iniciar la copia de seguridad.

DB2BACKUP_NOINTERRUPT
Iniciar la copia de seguridad. Especifica que la copia de seguridad se ejecutará de forma desatendida y que en las situaciones que normalmente requieran la intervención del usuario se intentarán sin primero devolver el control al llamador o bien se producirá un error. Utilice esta acción de llamador, por ejemplo, si se sabe que se han montado todos los soportes de almacenamiento necesarios para la copia de seguridad y no se desea que aparezcan mensajes de solicitud de programas de utilidad.

DB2BACKUP_CONTINUE
Continuar la copia de seguridad después de que el usuario haya realizado alguna acción solicitada por el programa de utilidad (por ejemplo, montar una nueva cinta).

DB2BACKUP_TERMINATE
Interrumpir la copia de seguridad después de que el usuario no haya realizado alguna acción solicitada por el programa de utilidad.

DB2BACKUP_DEVICE_TERMINATE
Eliminar un determinado dispositivo de la lista de dispositivos utilizados por la copia de seguridad. Cuando un determinado soporte de almacenamiento está lleno, la copia de seguridad devuelve un aviso al llamador (y continúa el proceso utilizando los dispositivos restantes). Invoque de nuevo la copia de seguridad con esta acción de llamador para eliminar de la lista de dispositivos utilizados el dispositivo que produjo el aviso.

DB2BACKUP_PARM_CHK
Se utiliza para validar parámetros sin realizar una copia de seguridad. Esta opción no interrumpe la conexión con la base de datos después de finalizar la ejecución de la llamada. Después de finalizar satisfactoriamente esta llamada, el usuario debe emitir una llamada con SQLUB_CONTINUE para proseguir con la acción.

DB2BACKUP_PARM_CHK_ONLY
Se utiliza para validar parámetros sin realizar una copia de seguridad. Antes de que concluya esta llamada, se interrumpe la conexión con la base de datos establecida por esta llamada y no es necesaria ninguna llamada más.

iBufferSize

Entrada. Tamaño del almacenamiento intermedio de copia de seguridad expresado en unidades de asignación de 4 KB (páginas). El mínimo es de 8 unidades.

iNumBuffers

Entrada. Especifica el número de almacenamientos intermedios de copia de seguridad que se utilizarán. El mínimo es 2. El máximo está limitado por la memoria.

iParallelism
Entrada. Grado de paralelismo (número de manipuladores de almacenamientos intermedios). El mínimo es 1. El máximo es 1024.

iOptions
Entrada. Mapa de bits de propiedades de copia de seguridad. Las opciones se deben combinar utilizando el operador de bits OR para producir un valor para **iOptions**. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2BACKUP_OFFLINE
Este valor proporciona una conexión exclusiva con la base de datos.

DB2BACKUP_ONLINE
Este valor permite que otras aplicaciones accedan a la base de datos mientras se realiza la copia de seguridad.

Nota: Las operaciones de copia de seguridad ONLINE pueden parecer estar bloqueadas si los usuarios mantienen bloqueos sobre datos LOB de SMS.

DB2BACKUP_DB
Copia de seguridad de la base de datos completa.

DB2BACKUP_TABLESPACE
Copia de seguridad de nivel de espacios de tablas. Para una copia de seguridad de nivel de espacios de tablas, proporcione una lista de los espacios de tablas en el parámetro **piTablespaceList**.

DB2BACKUP_INCREMENTAL
Especifica una imagen de copia de seguridad acumulativa (incremental). Una imagen de copia de seguridad incremental es una copia de todos los datos de la base de datos que se han modificado desde la operación de copia de seguridad completa más reciente que ha resultado satisfactoria.

DB2BACKUP_DELTA
Especifica una imagen de copia de seguridad (delta) no acumulativa. Una imagen de copia de seguridad delta es una copia de todos los datos de la base de datos que se han modificado desde la operación de copia de seguridad de cualquier tipo más reciente que ha resultado satisfactoria.

DB2BACKUP_COMPRESS
Especifica que la copia de seguridad se debe comprimir.

DB2BACKUP_INCLUDE_COMPR_LIB
Especifica que la biblioteca utilizada para comprimir la copia de seguridad se debe incluir en la imagen de copia de seguridad.

DB2BACKUP_EXCLUDE_COMPR_LIB
Especifica que la biblioteca utilizada para comprimir la copia de seguridad no se debe incluir en la imagen de copia de seguridad.

DB2BACKUP_INCLUDE_LOGS
Especifica que la imagen de copia de seguridad debe también incluir el rango de archivos de anotaciones necesarios para restaurar y llevar esta imagen hasta un punto en el tiempo coherente. El gestor de bases de datos incluye anotaciones cronológicas de base de datos en imágenes de copia de seguridad a

menos que especifique el parámetro EXCLUDE LOGS. El comportamiento por omisión no se aplica a una copia de seguridad no SSV de una base de datos particionada. Esta opción no es válida para una copia de seguridad fuera de línea. Por omisión, los archivos de anotaciones cronológicas se incluyen en los siguientes escenarios de copia de seguridad:

- Copia de seguridad en línea de una base de datos con una sola partición
- Copia de seguridad de una vista de un solo sistema (SSV) en línea o fuera de línea en una base de datos con varias particiones.
- Copia de seguridad instantánea en línea o fuera de línea

DB2BACKUP_EXCLUDE_LOGS
Especifica que la imagen de copia de seguridad no debe incluir ningún archivo de anotaciones cronológicas.

Nota: Al realizar una operación de copia de seguridad fuera de línea, las anotaciones cronológicas se excluyen tanto si se ha especificado esta opción como si no, a excepción de las copias de seguridad instantáneas en las que INCLUDE es el valor por omisión. Esta opción es el comportamiento por omisión para una base de datos no SSV de una base de datos particionada.
Por omisión, los archivos de anotaciones cronológicas se excluyen en los siguientes escenarios de copia de seguridad:

- Copia de seguridad fuera de línea de una base de datos con una sola partición
- Copia de seguridad en línea o fuera de línea de una base de datos con varias particiones, cuando no se utiliza una copia de seguridad de vista de un solo sistema

DB2BACKUP_MPP
Realiza una copia de seguridad de manera adecuada para una base de datos particionada.

iUtilImpactPriority
Entrada. Especifica el valor de prioridad a utilizar durante una copia de seguridad.

- Si este valor es distinto a cero, el programa de utilidad se ejecutará estrangulado. De lo contrario, el programa de utilidad no se ejecutará estrangulado.
- Si hay múltiples programas de utilidad en ejecución a la vez, este parámetro se utiliza para determinar una prioridad relativa entre las tareas estranguladas. Por ejemplo, imagine dos copias de seguridad concurrentes, una con prioridad 2 y otra con prioridad 4. Ambas estarán estranguladas, pero la que tenga prioridad 4 tendrá más recursos asignados. Establecer las prioridades en 2 y 4 no es distinto a establecerlas en 5 y 10 o 30 y 60. Los valores de las prioridades son puramente relativos.

piComprLibrary
Entrada. Indica el nombre de la biblioteca externa que se utilizará para realizar la compresión de la imagen de copia de seguridad. El nombre debe ser una vía de acceso totalmente calificada que haga referencia a un archivo del servidor. Si el valor es un puntero nulo o un puntero a una

serie vacía, DB2 utilizará la biblioteca por omisión para la compresión. Si no se encuentra la biblioteca especificada, no se realizará la copia de seguridad.

piComprOptions
Entrada. Describe un bloque de datos binarios que se pasará a la rutina de inicialización en la biblioteca de compresión. DB2 pasará esta serie directamente del cliente al servidor, de modo que los posibles problemas de inversión de bytes o de conversión de página de códigos los deberá manejar la biblioteca de compresión. Si el primer carácter del bloque de datos es '@', DB2 interpretará los datos restantes como el nombre de un archivo que se encuentra en el servidor. DB2 sustituirá entonces el contenido de **piComprOptions** y **iComprOptionsSize** por el contenido y tamaño de este archivo respectivamente y pasará estos nuevos valores a la rutina de inicialización.

iComprOptionsSize
Entrada. Valor entero de cuatro bytes y sin signo que representa el tamaño del bloque de datos pasado como **piComprOptions**. **SiComprOptionsSize** será cero solamente si **piComprOptions** es un puntero nulo.

iAllNodeFlag
Entrada. Entornos de bases de datos particionadas solamente. Indica si la operación de copia de seguridad se debe aplicar a todos o a algunos de los servidores de particiones de base de datos definidos en db2nodes.cfg. Los valores válidos son:

DB2_NODE_LIST
Aplicar a servidores de particiones de base de datos de una lista que se pasa en **piNodeList**.

DB2_ALL_NODES
Aplicar a todos los servidores de particiones de base de datos. **piNodeList** debe ser NULL. Es el valor por omisión.

DB2_ALL_EXCEPT
Aplicar a todos los servidores de particiones de base de datos excepto a los de una lista que se pasa en **piNodeList**.

iNumNodes
Entrada. Especifica el número de servidores de particiones de base de datos de la matriz **piNodeList**.

piNodeList
Entrada. Puntero a una matriz de números de servidor de particiones de base de datos en los que se debe realizar la copia de seguridad.

iNumMPPOutputStructs
Entrada. Especifica el número de elementos de la matriz **piMPPOutputStruct**. Debe ser igual o mayor que el número de servidores de particiones de base de datos que participan en esta operación de copia de seguridad.

piMPPOutputStruct
Salida. Puntero a una matriz de estructuras db2BackupMPPOutputStruct que especifican parámetros de salida para servidores de particiones de base de datos determinados.

Parámetros específicos de la estructura de datos db2TablespaceStruct

tablespaces
Entrada. Puntero a la lista de espacios de tablas de los que debe hacerse copia de seguridad. Para C, la lista son series de terminación nula. En el caso genérico, es una lista de estructuras db2Char.

numTablespaces
Entrada. Número de entradas en el parámetro **tablespaces**.

Parámetros de la estructura de datos db2MediaListStruct

locations
Entrada. Puntero a la lista de ubicaciones de soporte de almacenamiento. Para C, la lista son series de terminación nula. En el caso genérico, es una lista de estructuras db2Char.

numLocations
Entrada. Número de entradas del parámetro locations.

locationType
Entrada. Carácter que indica el tipo de soporte de almacenamiento. Los valores válidos (definidos en el archivo de cabecera sqlutil, ubicado en el directorio de inclusión) son:

SQLU_LOCAL_MEDIA: 'L'
Dispositivos locales (cintas, discos, disquetes o conexiones con nombre).

SQLU_XBSA_MEDIA: 'X'
Interfaz de XBSA.

SQLU_TSM_MEDIA: 'A'
Tivoli Storage Manager.

SQLU_OTHER_MEDIA: 'O'
Biblioteca de proveedor.

SQLU_SNAPSHOT_MEDIA: 'F'
Especifica que se debe realizar una copia de seguridad instantánea.

No puede utilizar SQLU_SNAPSHOT_MEDIA con ninguno de los siguientes:

- DB2BACKUP_COMPRESS
- DB2BACKUP_TABLESPACE
- DB2BACKUP_INCREMENTAL
- **iNumBuffers**
- **iBufferSize**
- **iParallelism**
- **piComprOptions**
- **iUtilImpactPriority**
- El campo **numLocations** de esta estructura debe ser 1 para la restauración instantánea

El comportamiento por omisión para una copia de seguridad instantánea es una copia de seguridad FULL DATABASE OFFLINE de todas las vías de acceso que componen la base de datos incluyendo todos los contenedores, el directorio de volúmenes local, la vía de acceso de base de datos (DBPATH) y las vías de

acceso de anotaciones cronológicas primaria y de reflejos (INCLUDE LOGS es el valor por omisión de todas las copias de seguridad instantáneas a menos que se indique EXCLUDE LOGS explícitamente).

En IBM Data Server se integra un controlador de API ACS de DB2 para el hardware de almacenamiento siguiente:

- IBM TotalStorage SAN Volume Controller
- IBM Enterprise Storage Server Model 800
- IBM System Storage DS6000
- IBM System Storage DS8000
- IBM System Storage N Series
- NetApp V-series
- NetApp FAS

Parámetro de las estructuras de datos db2BackupMPPOutputStruct y db2gBackupMPPOutputStruct

nodeNumber
Servidor de particiones de base de datos al que se aplica la opción.

backupSize
Tamaño de la copia de seguridad de la partición de base de datos especificada, en kilobytes.

sqlca sqlca de la partición de base de datos especificada.

Parámetros específicos de la estructura de datos db2gBackupStruct

iDBAliasLen
Entrada. Número entero sin signo, de 4 bytes, que representa la longitud, expresada en bytes, del alias de base de datos.

iApplicationIdLen
Entrada. Número entero sin signo, de 4 bytes, que representa la longitud en bytes del almacenamiento intermedio **poApplicationId**. Debe ser equivalente a SQLU_APPLID_LEN+1 (definido en sqlutil.h).

iTimestampLen
Entrada. Número entero sin signo, de 4 bytes, que representa la longitud en bytes del almacenamiento intermedio **poTimestamp**. Debe ser equivalente a SQLU_TIME_STAMP_LEN+1 (definido en sqlutil.h).

iUsernameLen
Entrada. Número entero sin signo de 4 bytes que representa la longitud en bytes del nombre de usuario. El valor se establece en cero si no se proporciona ningún nombre de usuario.

iPasswordLen
Entrada. Número entero sin signo, de 4 bytes, que representa la longitud en bytes de la contraseña. El valor se establece en cero si no se proporciona ninguna contraseña.

iComprLibraryLen
Entrada. Número entero sin signo, de 4 bytes, que representa la longitud, en bytes, del nombre de la biblioteca especificada en **piComprLibrary**. El valor se establece en cero si no se proporciona ningún nombre de biblioteca.

Parámetros de la estructura de datos db2Char

pioData
Puntero a un almacenamiento intermedio de datos de caracteres. Si el valor es NULL, no se devolverán datos.

iLength
Entrada. Tamaño del almacenamiento intermedio **pioData**.

oLength
Salida. Número de caracteres válidos de datos contenidos en el almacenamiento intermedio **pioData**.

Notas de uso

Sólo se puede realizar una copia de seguridad instantánea con **versionNumber** db2Version950 o superior. Si especifica el tipo de soporte SQLU_SNAPSHOT_MEDIA con **versionNumber** inferior a db2Version950, la base de datos DB2 devolverá un error.

Esta función está exenta de todas las reglas de control del acceso basadas en etiquetas (reglas LBAC). Realiza una copia de seguridad de todos los datos, incluso de los datos protegidos. Además, los datos de la propia copia de seguridad no están protegidos por LBAC. Cualquier usuario que disponga de la copia de seguridad y un lugar donde restaurarla puede acceder a los datos.

Puesto que deberá realizar una copia de seguridad de la base de datos regularmente, es posible que acumule imágenes de copia de seguridad de base de datos de gran tamaño, numerosas anotaciones cronológicas de base de datos e imágenes de copia de carga que pueden ocupar una gran cantidad de espacio en disco. Consulte la tarea "Gestión de objetos de recuperación" para obtener información sobre cómo gestionar estos objetos de recuperación.

Notas sobre el uso para la copia de seguridad de una vista de un solo sistema (SSV) en un entorno de bases de datos particionadas

- Para realizar una copia de seguridad SSV, especifique **iOptions** DB2BACKUP_MPP y DB2BACKUP_DB o DB2BACKUP_TABLESPACE.
- Sólo se puede realizar una copia de seguridad SSV con **versionNumber** db2Version950 o superior. Si especifica **iOptions** DB2BACKUP_MPP con **versionNumber** inferior a db2Version950, la base de datos DB2 devolverá un error. Si especifica otras opciones relacionadas con la copia de seguridad SSV con **versionNumber** inferior a db2Version950, la base de datos DB2 pasará por alto estas opciones.
- Los valores para **piMediaList** especificados directamente en db2BackupStruct se utilizarán como valores por omisión en todos los nodos.
- El valor de **oBackupSize** devuelto en db2BackupStruct es la suma de todos los tamaños de copia de seguridad de todos los nodos. El valor de **backupSize** devuelto en db2BackupMPPOutputStruct es el tamaño de la copia de seguridad en la partición de base de datos especificada.
- **iAllNodeFlag**, **iNumNodes** y **piNodeList** funcionan igual que los elementos denominados de forma similar en db2RollforwardStruct, a excepción de que no hay ningún valor CAT_NODE_ONLY para **iAllNodeFlag**.

- Las copias de seguridad SSV realizadas con la acción de llamador DB2BACKUP_BACKUP se realizan como si se hubiese especificado la acción del llamador DB2BACKUP_NOINTERRUPT.
- ***poMPPOutputStruct** apunta a memoria asignada por el llamador que contiene como mínimo tantos elementos como hay en las particiones de base de datos que participan en la copia de seguridad.

db2CfgGet - Obtener los parámetros de configuración del gestor de bases de datos o de la base de datos

Devuelve los valores de entradas individuales contenidas en un determinado archivo de configuración de base de datos o de configuración de gestor de bases de datos.

Ámbito

Se devuelve información sobre un determinado archivo de configuración de base de datos solamente para la partición de base de datos desde donde se ejecuta la API.

Autorización

Ninguna

Conexión necesaria

Para obtener el valor activo actual de un parámetro de configuración para un determinado archivo de configuración de base de datos, es necesaria una conexión con la base de datos. Para obtener el valor activo actual de un parámetro de configuración para el gestor de bases de datos, es necesaria una conexión de instancia. En otro caso, no es necesaria una conexión con una base de datos o con una instancia.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2CfgGet (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2Cfg
{
   db2Uint32 numItems;
   struct db2CfgParam *paramArray;
   db2Uint32 flags;
   char *dbname;
} db2Cfg;

typedef SQL_STRUCTURE db2CfgParam
{
   db2Uint32 token;
   char *ptrvalue;
   db2Uint32 flags;
} db2CfgParam;
```

```
SQL_API_RC SQL_API_FN
  db2gCfgGet (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gCfg
{
   db2Uint32 numItems;
   struct db2gCfgParam *paramArray;
   db2Uint32 flags;
   db2Uint32 dbname_len;
   char *dbname;
} db2gCfg;

typedef SQL_STRUCTURE db2gCfgParam
{
   db2Uint32 token;
   db2Uint32 ptrvalue_len;
   char *ptrvalue;
   db2Uint32 flags;
} db2gCfgParam;
```

Parámetros de la API db2CfgGet

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2Cfg.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2Cfg

numItems
Entrada. Número de parámetros de configuración de la matriz paramArray. Establezca este valor en db2CfgMaxParam para especificar el número máximo de elementos de paramArray.

paramArray
Entrada. Puntero a la estructura db2CfgParam.

flags Entrada. Especifica el tipo de acción que se debe realizar. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

db2CfgDatabase
Especifica devolver los valores contenidos en el archivo de configuración de la base de datos.

db2CfgDatabaseManager
Especifica devolver los valores contenidos en el archivo de configuración del gestor de bases de datos.

db2CfgImmediate
Devuelve los valores actuales de los parámetros de configuración almacenados en la memoria.

db2CfgDelayed
Obtiene los valores de los parámetros de configuración contenidos

en disco. Estos valores no pasan a ser los valores actuales contenidos en la memoria hasta la siguiente conexión con la base de datos o instancia.

db2CfgGetDefaults
Devuelve los valores por omisión del parámetro de configuración.

db2CfgReset
Restaura los valores por omisión.

dbname
Entrada. Nombre de la base de datos.

Parámetros de la estructura de datos db2CfgParam

token Entrada. Identificador del parámetro de configuración.

Las entradas y tipos de datos válidos para el parámetro token de db2CfgParam están listados en el ″Resumen de parámetros de configuración″.

ptrvalue
Salida. Valor del parámetro de configuración.

flags Salida. Proporciona información específica para cada parámetro de una petición. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

db2CfgParamAutomatic
Indica si el parámetro recuperado tiene el valor automatic. Para determinar si un parámetro de configuración concreto se ha establecido en el valor automatic, ejecute una operación booleana AND sobre el valor devuelto por el distintivo y la palabra clave db2CfgParamAutomatic definida en db2ApiDf.h.

db2CfgParamComputed
Indica si el parámetro recuperado tiene el valor computed. Para determinar si un parámetro de configuración concreto se ha establecido en el valor computed, ejecute una operación booleana AND sobre el valor devuelto por el distintivo y la palabra clave db2CfgParamComputed definida en db2ApiDf.h.

Si la operación booleana AND tiene el valor false para las dos palabras clave anteriores, esto significa que el valor del parámetro recuperado se ha establecido manualmente.

Parámetros específicos de la estructura de datos db2gCfg

dbname_len
Entrada. Longitud, en bytes, del parámetro dbname.

Parámetros específicos de la estructura de datos db2gCfgParam

ptrvalue_len
Entrada. Longitud, en bytes, del parámetro ptrvalue.

Notas de uso

Los parámetros de configuración `maxagents` y `maxcagents` han quedado obsoletos. En un futuro release, es posible que estos parámetros de configuración se eliminen por completo.

La API db2CfgGet dará soporte a solicitudes de SQLF_KTN_MAXAGENTS y SQLF_KTN_MAXCAGENTS, pero se devolverá 0 si el servidor es DB2 v9.5.

db2CfgSet - Definir los parámetros de configuración del gestor de bases de datos o de la base de datos

Modifica entradas individuales de un determinado archivo de configuración de base de datos o archivo de configuración de gestor de bases de datos. Existe un archivo de configuración de base de datos en cada nodo donde se ha creado la base de datos.

Ámbito

Las modificaciones hechas en el archivo de configuración de la base de datos afectan por omisión a todas las particiones de base de datos.

Autorización

Para realizar modificaciones en el archivo de configuración de la base de datos es necesaria una de estas autorizaciones:

- sysadm
- sysctrl
- sysmaint

Para realizar modificaciones en el archivo de configuración del gestor de bases de datos:

- sysadm

Conexión necesaria

Para hacer una modificación en línea de un parámetro de configuración para una base de datos específica, es necesario establecer una conexión con la base de datos. Para hacer una modificación en línea de un parámetro de configuración para el gestor de bases de datos, es necesario establecer una conexión con la base de datos.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
   db2CfgSet (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2Cfg
{
   db2Uint32 numItems;
   struct db2CfgParam *paramArray;
   db2Uint32 flags;
   char *dbname;
   SQL_PDB_NODE_TYPE dbpartitionnum;
} db2Cfg;

typedef SQL_STRUCTURE db2CfgParam
{
```

```
   db2Uint32 token;
   char *ptrvalue;
   db2Uint32 flags;
} db2CfgParam;

SQL_API_RC SQL_API_FN
   db2gCfgSet (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gCfg
{
   db2Uint32 numItems;
   struct db2gCfgParam *paramArray;
   db2Uint32 flags;
   db2Uint32 dbname_len;
   char *dbname;
} db2gCfg;

typedef SQL_STRUCTURE db2gCfgParam
{
   db2Uint32 token;
   db2Uint32 ptrvalue_len;
   char *ptrvalue;
   db2Uint32 flags;
} db2gCfgParam;
```

Parámetros de la API db2CfgSet

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2Cfg.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2Cfg

numItems
Entrada. Número de parámetros de configuración de la matriz paramArray. Establezca este valor en db2CfgMaxParam para especificar el número máximo de elementos de paramArray.

paramArray
Entrada. Puntero a la estructura db2CfgParam.

flags Entrada. Especifica el tipo de acción que se debe realizar. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

db2CfgDatabase
Especifica devolver los valores contenidos en el archivo de configuración de la base de datos.

db2CfgDatabaseManager
Especifica devolver los valores contenidos en el archivo de configuración del gestor de bases de datos.

db2CfgImmediate
Devuelve los valores actuales de los parámetros de configuración almacenados en la memoria.

db2CfgDelayed
Obtiene los valores de los parámetros de configuración contenidos en disco. Estos valores no pasan a ser los valores actuales contenidos en la memoria hasta la siguiente conexión con la base de datos o instancia.

db2CfgGetDefaults
Devuelve los valores por omisión del parámetro de configuración.

db2CfgReset
Restaura los valores por omisión.

db2CfgSingleDbpartition
Para actualizar o restaurar la configuración de la base de datos en una partición de base de datos específica, establezca este distintivo y proporcione un valor para dbpartitionnum.

dbname
Entrada. Nombre de la base de datos.

dbpartitionnum
Entrada. Especifica en qué partición de la base de datos establecerá el valor de configuración esta API.

Parámetros de la estructura de datos db2CfgParam

token Entrada. Identificador del parámetro de configuración. Las entradas y tipos de datos válidos para el elemento de señal db2CfgParam están listados en ″Resumen de parámetros de configuración″.

ptrvalue
Salida. Valor del parámetro de configuración.

flags Entrada. Proporciona información específica para cada parámetro de una petición. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

db2CfgParamAutomatic
Indica si el parámetro recuperado tiene el valor automatic. Para determinar si el valor de un determinado parámetro de configuración se ha establecido en automatic, ejecute una operación boolena AND sobre el valor devuelto por el distintivo y la palabra clave db2CfgParamAutomatic definida en db2ApiDf.h.

db2CfgParamComputed
Indica si el parámetro recuperado tiene el valor computed. Para determinar si un parámetro de configuración concreto se ha establecido en el valor computed, ejecute una operación booleana AND sobre el valor devuelto por el distintivo y la palabra clave db2CfgParamComputed definida en db2ApiDf.h.

db2CfgParamManual
Se utiliza para suprimir la definición del valor automatic o computed de un parámetro y definir el parámetro en el valor actual. El campo **ptrvalue** se omite y puede definirse en NULL.

Parámetros específicos de la estructura de datos db2gCfg

dbname_len
Entrada. Longitud, en bytes, del parámetro dbname.

Parámetros específicos de la estructura de datos db2gCfgParam

ptrvalue_len
Entrada. Longitud, en bytes, del parámetro ptrvalue.

Notas de uso

Los parámetros de configuración maxagents y maxcagents han quedado obsoletos. En un futuro release, es posible que estos parámetros de configuración se eliminen por completo.

La API db2CfgSet dará soporte a actualizaciones de los parámetros de configuración maxagents y maxcagents, pero DB2 pasará por alto estas actualizaciones.

Ejemplos de uso

CASO 1: El parámetro MAXAPPLS se establecerá en 50 en dbpartitionnum 30.

CASO 2: El parámetro MAXAPPLS se establecerá en 50 en todas las dbpartitionnum.

```
int updateDbConfig()
{
    struct sqlca sqlca = {0};
    db2Cfg cfgStruct = {0};
    db2CfgParam cfgParameters[2];
    char *dbAlias = "SAMPLE";

    /* inicializar cfgParameters */
    cfgParameters[0].flags = 0;
    cfgParameters[0].token = SQLF_DBTN_TSM_OWNER;
    cfgParameters[0].ptrvalue = (char *)malloc(sizeof(char) * 65);
    cfgParameters[1].flags = 0;
    cfgParameters[1].token = SQLF_DBTN_MAXAPPLS;
    cfgParameters[1].ptrvalue = (char *)malloc(sizeof(sqluint16));

    /* establecer dos campos DB Config. */
    strcpy(cfgParameters[0].ptrvalue, "tsm_owner");
    *(sqluint16 *)(cfgParameters[1].ptrvalue) = 50;

    /* inicializar cfgStruct para actualizar la base de datos de configuración */
    /* en una sola partición */
    cfgStruct.numItems = 2;
    cfgStruct.paramArray = cfgParameters;
    cfgStruct.flags = db2CfgDatabase | db2CfgImmediate;
    cfgStruct.flags |= db2CfgSingleDbpartition;
    cfgStruct.dbname = dbAlias;
    cfgStruct.dbpartitionnum = 30;

    /* CASO 1: actualizar configuración de base de datos */
    db2CfgSet(db2Version950, (void *)&cfgStruct, &sqlca);

    /* inicializar cfgStruct para actualizar la base de datos de configuración */
    /* en todas las particiones de la base de datos */
    cfgStruct.flags &= ~db2CfgSingleDbpartition;

    /* CASO 2: actualizar configuración de base de datos */
    db2CfgSet(db2Version950, (void *)&cfgStruct, &sqlca);
    ..............
}
```

db2ConvMonStream - Convertir la corriente de supervisor en el formato anterior a la versión 6

Convierte el formato nuevo, autodescrito de un único elemento de datos lógicos (por ejemplo, SQLM_ELM_DB2) con la estructura de supervisor externa anterior a la versión 6 (por ejemplo, sqlm_db2). Cuando se actualizan las llamadas a API para que utilicen la corriente posterior a la versión 5, deben recorrerse los datos del supervisor utilizando el formato de corriente nuevo (por ejemplo, el usuario debe buscar el elemento SQLM_ELM_DB2). Seguidamente, esta parte de la corriente puede pasarse a la API de conversión para obtener los datos asociados anteriores a la versión 6.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2ConvMonStream (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ConvMonStreamData
{
   void *poTarget;
   struct sqlm_header_info *piSource;
   db2Uint32 iTargetType;
   db2Uint32 iTargetSize;
   db2Uint32 iSourceType;
} db2ConvMonStreamData;

SQL_API_RC SQL_API_FN
  db2gConvMonStream (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);
```

Parámetros de la API db2ConvMonStream

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2ConvMonStreamData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2ConvMonStreamData

poTarget
Salida. Puntero a la estructura de salida del supervisor de destino (por ejemplo, sqlm_db2). A continuación se ofrece una lista de tipos de salida y los tipos de entrada respectivos.

piSource
Entrada. Puntero al elemento de datos lógicos que se va a convertir (por ejemplo, SQLM_ELM_DB2). A continuación se ofrece una lista de tipos de salida y los tipos de entrada respectivos.

iTargetType
Entrada. Tipo de conversión que se va a realizar. Especifica el valor del tipo v5 en sqlmon.h para la instancia SQLM_DB2_SS.

iTargetSize
Entrada. A este parámetro normalmente puede asignársele el valor del tamaño de la estructura a la que apunta poTarget; no obstante, para elementos a los que usualmente se hace referencia mediante un valor de desplazamiento con respecto al fin de la estructura (por ejemplo, el texto de sentencia de sqlm_stmt), especifique un almacenamiento intermedio que sea lo suficientemente grande para contener los elementos de tamaño estático de sqlm_stmt, así como una sentencia del tamaño mayor posible que se vaya a extraer; es decir, SQL_MAX_STMT_SIZ más sizeof(sqlm_stmt).

iSourceType
Entrada. Tipo de corriente fuente. Los valores válidos son SQLM_STREAM_SNAPSHOT (corriente de instantánea) o SQLM_STREAM_EVMON (corriente de supervisor de sucesos).

Notas de uso

A continuación se proporciona una lista de elementos de datos convertibles soportados:

Tabla 6. Elementos de datos convertibles soportados: variables de instantáneas

Tipo de corriente de datos variable de instantánea	Estructura
SQLM_ELM_APPL	sqlm_appl
SQLM_ELM_APPL_INFO	sqlm_applinfo
SQLM_ELM_DB2	sqlm_db2
SQLM_ELM_FCM	sqlm_fcm
SQLM_ELM_FCM_NODE	sqlm_fcm_node
SQLM_ELM_DBASE	sqlm_dbase
SQLM_ELM_TABLE_LIST	sqlm_table_header
SQLM_ELM_TABLE	sqlm_table
SQLM_ELM_DB_LOCK_LIST	sqlm_dbase_lock
SQLM_ELM_APPL_LOCK_LIST	sqlm_appl_lock
SQLM_ELM_LOCK	sqlm_lock
SQLM_ELM_STMT	sqlm_stmt
SQLM_ELM_SUBSECTION	sqlm_subsection
SQLM_ELM_TABLESPACE_LIST	sqlm_tablespace_header

Tabla 6. Elementos de datos convertibles soportados: variables de instantáneas (continuación)

Tipo de corriente de datos variable de instantánea	Estructura
SQLM_ELM_TABLESPACE	sqlm_tablespace
SQLM_ELM_ROLLFORWARD	sqlm_rollfwd_info
SQLM_ELM_BUFFERPOOL	sqlm_bufferpool
SQLM_ELM_LOCK_WAIT	sqlm_lockwait
SQLM_ELM_DCS_APPL	sqlm_dcs_appl, sqlm_dcs_applid_info, sqlm_dcs_appl_snap_stats, sqlm_xid, sqlm_tpmon
SQLM_ELM_DCS_DBASE	sqlm_dcs_dbase
SQLM_ELM_DCS_APPL_INFO	sqlm_dcs_applid_info
SQLM_ELM_DCS_STMT	sqlm_dcs_stmt
SQLM_ELM_COLLECTED	sqlm_collected

Tabla 7. Elementos de datos convertibles soportados: variables de supervisor de sucesos

Tipo de corriente de datos variables de supervisor de sucesos	Estructura
SQLM_ELM_EVENT_DB	sqlm_db_event
SQLM_ELM_EVENT_CONN	sqlm_conn_event
SQLM_ELM_EVENT_TABLE	sqlm_table_event
SQLM_ELM_EVENT_STMT	sqlm_stmt_event
SQLM_ELM_EVENT_XACT	sqlm_xaction_event
SQLM_ELM_EVENT_DEADLOCK	sqlm_deadlock_event
SQLM_ELM_EVENT_DLCONN	sqlm_dlconn_event
SQLM_ELM_EVENT_TABLESPACE	sqlm_tablespace_event
SQLM_ELM_EVENT_DBHEADER	sqlm_dbheader_event
SQLM_ELM_EVENT_START	sqlm_evmon_start_event
SQLM_ELM_EVENT_CONNHEADER	sqlm_connheader_event
SQLM_ELM_EVENT_OVERFLOW	sqlm_overflow_event
SQLM_ELM_EVENT_BUFFERPOOL	sqlm_bufferpool_event
SQLM_ELM_EVENT_SUBSECTION	sqlm_subsection_event
SQLM_ELM_EVENT_LOG_HEADER	sqlm_event_log_header

La estructura sqlm_rollfwd_ts_info no se convierte; sólo contiene un nombre de espacio de tablas al que se puede acceder directamente desde la corriente. La estructura sqlm_agent tampoco se convierte; sólo contiene el pid del agente, al que también se puede acceder directamente desde la corriente.

db2DatabasePing - Sondear la base de datos para probar el tiempo de respuesta de la red

Prueba el tiempo de respuesta de red de la conectividad subyacente entre un cliente y un servidor de bases de datos. Una aplicación puede utilizar esta API cuando se accede a un servidor de bases de datos del sistema principal utilizando DB2 Connect, ya sea directamente o mediante una pasarela.

Autorización

Ninguna

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2DatabasePing (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2DatabasePingStruct
{
   char iDbAlias[SQL_ALIAS_SZ + 1];
   db2int32 RequestPacketSz;
   db2int32 ResponsePacketSz;
   db2Uint16 iNumIterations;
   db2Uint32 *poElapsedTime;
} db2DatabasePingStruct;

SQL_API_RC SQL_API_FN
  db2gDatabasePing (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gDatabasePingStruct
{
   db2Uint16 iDbAliasLength;
   char iDbAlias[SQL_ALIAS_SZ + 1];
   db2int32 RequestPacketSz;
   db2int32 ResponsePacketSz;
   db2Uint16 iNumIterations;
   db2Uint32 *poElapsedTime;
} db2gDatabasePingStruct;
```

Parámetros de la API db2DatabasePing

versionNumber
Entrada. Especifica la versión y el release de la base de datos DB2 o del producto DB2 Connect que la aplicación esté utilizando.

pParmStruct
Entrada. Puntero a la estructura db2DatabasePingStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2DatabasePingStruct

iDbAlias
Entrada. Nombre de alias de base de datos. Reservado para una utilización futura. Si se proporciona un valor, se omite.

RequestPacketSz
Entrada. Tamaño, en bytes, del paquete que se debe enviar al servidor. El tamaño debe estar entre 0 y 32767 inclusive. Este parámetro sólo es válido en servidores que ejecutan DB2 Universal Database (UDB) para Linux, UNIX y Windows Versión 8 o superior, o DB2 UDB para z/OS Versión 8 o superior.

ResponsePacketSz
Entrada. Tamaño, en bytes, del paquete que se debe devolver al cliente. El tamaño debe estar entre 0 y 32767 inclusive. Este parámetro sólo es válido en servidores que ejecutan DB2 UDB para Linux, UNIX y Windows Versión 8 o superior, o DB2 UDB para z/OS Versión 8 o superior.

iNumIterations
Entrada. Número de iteraciones de petición de prueba. El valor debe estar entre 1 y 32767 inclusive.

poElapsedTime
Salida. Puntero a una matriz de números enteros de 32 bits en la que el número de elementos es igual a iNumIterations. Cada elemento de la matriz contiene el tiempo transcurrido, en microsegundos, correspondiente a una iteración de petición de prueba.

Nota: La aplicación debe asignar memoria para esta matriz antes de invocar esta API.

Parámetros específicos de la estructura de datos db2gDatabasePingStruct

iDbAliasLength
Entrada. Longitud del nombre alias de la base de datos. Reservado para una utilización futura.

Notas de uso

Esta API no funcionará cuando se utilice desde un cliente DB2 UDB Versión 7 mediante DB2 Connect Versión 8 con un servidor de bases de datos de sistema principal DB2.

db2DatabaseQuiesce - Inmovilizar la base de datos

Desconecta de la base de datos a todos los usuarios, retrotrae inmediatamente todas las transacciones activas o espera a que finalicen sus unidades de trabajo actuales dentro del número especificado de minutos (si no finalizan dentro de ese tiempo, la operación falla), y coloca la base de datos en la modalidad de inmovilización. Esta API proporciona acceso exclusivo a la base de datos. Durante este período de inmovilización, los usuarios con la autorización adecuada pueden realizar la administración del sistema en la base de datos. Una vez se ha completado la administración, puede desinmovilizar la base de datos utilizando la

API db2DatabaseUnquiesce. La API db2DatabaseUnquiesce permite a otros usuarios conectarse a la base de datos, sin tener que concluir y realizar otro inicio de la base de datos. En esta modalidad, solamente los grupos o usuarios con autorización QUIESCE CONNECT y autorización sysadm, sysmaint o sysctrl tendrán acceso a la base de datos y a sus objetos.

Autorización

Una de las siguientes:

- sysadm
- dbadm

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2DatabaseQuiesce (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2DbQuiesceStruct
{
  char *piDatabaseName;
  db2Uint32 iImmediate;
  db2Uint32 iForce;
  db2Uint32 iTimeout;
} db2DbQuiesceStruct;

SQL_API_RC SQL_API_FN
  db2gDatabaseQuiesce (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gDbQuiesceStruct
{
  db2Uint32 iDatabaseNameLen;
  char *piDatabaseName;
  db2Uint32 iImmediate;
  db2Uint32 iForce;
  db2Uint32 iTimeout;
} db2gDbQuiesceStruct;
```

Parámetros de la API db2DatabaseQuiesce

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2DbQuiesceStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2DbQuiesceStruct

piDatabaseName
Entrada. Nombre de la base de datos.

iImmediate
Entrada. Los valores válidos son:

TRUE=1
Desconectar las aplicaciones inmediatamente.

FALSE=0
Desconexión diferida. Las aplicaciones esperarán el número de minutos especificado en el parámetro iTimeout para permitir que finalicen sus unidades de trabajo actuales, y luego concluirán. Si esta desconexión forzada no se puede realizar dentro del número de minutos especificado por el parámetro iTimeout, la operación de inmovilización fallará.

iForce Entrada. Reservado para una utilización futura.

iTimeout
Entrada. Especifica el tiempo, en minutos, que se debe esperar para que las aplicaciones confirmen la unidad actual de trabajo. Si no se especifica iTimeout, en un entorno de bases de datos de una sola partición, el valor por omisión es 10 minutos. En un entorno de bases de datos particionadas, se utilizará el valor especificado por el parámetro de configuración de gestor de bases de datos start_stop_time.

Parámetros específicos de la estructura de datos db2gDbQuiesceStruct

iDatabaseNameLen
Entrada. Especifica la longitud, en bytes, de piDatabaseName.

db2DatabaseRestart - Reiniciar base de datos

Reinicia una base de datos que ha terminado anormalmente y que se ha dejado en un estado incoherente. Cuando esta API se ha completado satisfactoriamente, la aplicación permanece conectada a la base de datos si el usuario tiene el privilegio CONNECT.

Ámbito

Esta API sólo afecta al servidor de particiones de base de datos en el que se ejecuta la API.

Autorización

Ninguna

Conexión necesaria

Esta API establece una conexión de base de datos.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2DatabaseRestart (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef struct db2RestartDbStruct
{
   char *piDatabaseName;
   char *piUserId;
   char *piPassword;
   char *piTablespaceNames;
   db2int32 iOption;
} db2RestartDbStruct;

SQL_API_RC SQL_API_FN
  db2gDatabaseRestart (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef struct db2gRestartDbStruct
{
   db2Uint32 iDatabaseNameLen;
   db2Uint32 iUserIdLen;
   db2Uint32 iPasswordLen;
   db2Uint32 iTablespaceNamesLen;
   char *piDatabaseName;
   char *piUserId;
   char *piPassword;
   char *piTablespaceNames;
} db2gRestartDbStruct;
```

Parámetros de la API db2DatabaseRestart

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParamStruct.

pParamStruct
Entrada. Puntero a la estructura db2RestartDbStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2RestartDbStruct

piDatabaseName
Entrada. Puntero a una serie que contiene el alias de la base de datos que se debe reiniciar.

piUserId
Entrada. Puntero a una serie que contiene el nombre de usuario de la aplicación. Puede ser NULL.

piPassword
Entrada. Puntero a una serie que contiene una contraseña para el nombre de usuario especificado (si lo hay). Puede ser nula. Puede ser NULL.

piTablespaceNames
Entrada. Puntero a una serie que contiene una lista de nombres de espacios de tablas que se deben descartar durante la operación de reinicio. Puede ser NULL.

iOption
Entrada. Los valores válidos son:

DB2_DB_SUSPEND_NONE
Efectúa una recuperación normal para errores.

DB2_DB_RESUME_WRITE
Valor necesario para realizar una recuperación para errores en una base de datos que tiene escrituras de E/S suspendidas.

Parámetros específicos de la estructura de datos db2gRestartDbStruct

iDatabaseNameLen
Entrada. Longitud, en bytes, del parámetro piDatabaseName.

iUserIdLen
Entrada. Longitud, en bytes, del parámetro piUserId.

iPasswordLen
Entrada. Longitud, en bytes, del parámetro piPassword.

iTablespaceNamesLen
Entrada. Longitud, en bytes, del parámetro piTablespaceNames.

Notas de uso

Llame a esta API si un intento de conexión a una base de datos devuelve un mensaje de error, que indica que se debe reiniciar la base de datos. Esta acción sólo se produce si la sesión anterior con esta base de datos ha terminado anormalmente (por ejemplo debido a una anomalía de alimentación).

Cuando esta API se ha completado, se mantiene una conexión compartida a la base de datos si el usuario tiene privilegio CONNECT y se emite un aviso de SQL si existe alguna transacción dudosa. En este caso, la base de datos sigue siendo utilizable pero si no se resuelven las transacciones dudosas antes de que se descarte la última conexión a la base de datos, se deberá realizar otra llamada a la API para poder utilizar de nuevo la base de datos.

En el caso del registro cronológico circular, una operación de reinicio de base de datos fallará si existe cualquier problema con los espacios de tablas, tal como un error de E/S, un sistema de archivos sin montar, etc. Si la pérdida de esos espacios de tablas no es una cuestión importante, sus nombres se pueden especificar explícitamente; esto los colocará en un estado de descarte pendiente y la operación de reinicio se puede ejecutar satisfactoriamente.

Sintaxis de la API de REXX

```
RESTART DATABASE database_alias [USER username USING password]
```

Parámetros de la API de REXX

database_alias
Alias de la base de datos que se debe reiniciar.

username
Nombre de usuario utilizado para reiniciar la base de datos.

password
Contraseña utilizada para autentificar el nombre de usuario.

db2DatabaseUnquiesce - Movilizar base de datos

Restaura el acceso de usuario a las bases de datos que se han inmovilizado para el mantenimiento u otras razones. El acceso del usuario se restaura sin necesita de cerrar y reiniciar la base de datos.

Autorización

Una de las siguientes:

- sysadm
- dbadm

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2DatabaseUnquiesce (
              db2Uint32 versionNumber,
              void * pParmStruct,
              struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2DbUnquiesceStruct
{
              char *piDatabaseName;
} db2DbUnquiesceStruct;

SQL_API_RC SQL_API_FN
  db2gDatabaseUnquiesce (
              db2Uint32 versionNumber,
              void * pParmStruct,
              struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gDbUnquiesceStruct
{
              db2Uint32 iDatabaseNameLen;
              char *piDatabaseName;
} db2gDbUnquiesceStruct;
```

Parámetros de la API db2DatabaseUnquiesce

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2DbUnquiesceStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2DbUnquiesceStruct

piDatabaseName
Entrada. Nombre de la base de datos.

Parámetros específicos de la estructura de datos db2gDbUnquiesceStruct

iDatabaseNameLen
Entrada. Especifica la longitud, en bytes, de piDatabaseName.

db2DbDirCloseScan - Finalizar una exploración del directorio de bases de datos locales o del sistema

Libera los recursos asignados por db2DbDirOpenScan.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2DbDirCloseScan (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2DbDirCloseScanStruct
{
   db2Uint16 iHandle;
} db2DbDirCloseScanStruct;

SQL_API_RC SQL_API_FN
  db2gDbDirCloseScan (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gDbDirCloseScanStruct
{
   db2Uint16 iHandle;
} db2gDbDirCloseScanStruct;
```

Parámetros de la API db2DbDirCloseScan

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2DbDirCloseScanStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2DbDirCloseScanStruct

iHandle
Entrada. Identificador devuelto desde la API db2DbDirOpenScan asociada.

db2DbDirGetNextEntry - Obtener la entrada siguiente del directorio de bases de datos locales o del sistema

Devuelve la próxima entrada de la copia del directorio de bases de datos del sistema o del directorio de bases de datos locales devuelta por db2DbDirOpenScan. Las llamadas subsiguientes a esta API devuelven entradas adicionales.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

db2ApiDf.h

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2DbDirGetNextEntry (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE     db2DbDirNextEntryStructV9
{
   db2Uint16 iHandle;
   struct db2DbDirInfoV9  *poDbDirEntry;
} db2DbDirNextEntryStructV9;

SQL_STRUCTURE      db2DbDirInfoV9
{
   _SQLOLDCHAR alias[SQL_ALIAS_SZ];
   _SQLOLDCHAR dbname[SQL_DBNAME_SZ];
   _SQLOLDCHAR     drive[SQL_DB_PATH_SZ];
   _SQLOLDCHAR intname[SQL_INAME_SZ];
   _SQLOLDCHAR nodename[SQL_NNAME_SZ];
   _SQLOLDCHAR dbtype[SQL_DBTYP_SZ];
   _SQLOLDCHAR comment[SQL_CMT_SZ];
   short com_codepage;
   _SQLOLDCHAR type;
   unsigned short            authentication;
   char glbdbname[SQL_DIR_NAME_SZ];
   _SQLOLDCHAR dceprincipal[SQL_DCEPRIN_SZ];
   short cat_nodenum;
   short nodenum;
   _SQLOLDCHAR althostname[SQL_HOSTNAME_SZ];
   _SQLOLDCHAR altportnumber[SQL_SERVICE_NAME_SZ];
};

SQL_API_RC SQL_API_FN
  db2gDbDirGetNextEntry (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);
```

```
typedef SQL_STRUCTURE     db2gDbDirNextEntryStrV9
{
   db2Uint16 iHandle;
   struct db2DbDirInfoV9  *poDbDirEntry;
} db2gDbDirNextEntryStrV9;
```

Parámetros de la API db2DbDirGetNextEntry

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura que se ha pasado como segundo parámetro, **pParmStruct**.

pParmStruct
Entrada. Un puntero para la estructura db2DbDirGetNextEntryStructV9.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2DbDirNextEntryStructV9

iHandle
Entrada. Identificador devuelto desde la API db2DbDirOpenScan asociada.

poDbDirEntry
Salida. Puntero a la estructura db2DbDirInfoV9. La API asigna el espacio de los datos de directorio y se devuelve al llamador un puntero a dicho espacio.

Parámetros de la estructura de datos db2DbDirInfoV9

alias Nombre alternativo de la base de datos.

dbname
Nombre de la base de datos.

drive El nombre de vía de acceso del directorio de bases de datos local donde reside la base de datos. Este campo sólo se devuelve si el directorio de bases de datos del sistema está abierto para exploración.

Nota: En Windows, este parámetro es CHAR(12).

intname
Símbolo que identifica el subdirectorio de base de datos. Este campo sólo se devuelve si el directorio de bases de datos locales está abierto para exploración.

nodename
Nombre del nodo donde reside la base de datos. Este campo sólo se devuelve si la base de datos catalogada es una base de datos remota.

dbtype
Información de release del gestor de bases de datos.

comment
El comentario asociado con la base de datos.

com_codepage
Página de códigos del comentario. No se utiliza.

type Tipo de entrada. Los valores válidos son:

SQL_INDIRECT
Base de datos creada por la instancia actual (tal como está definida por el valor de la variable de entorno **DB2INSTANCE**).

SQL_REMOTE
La base de datos reside en una instancia diferente.

SQL_HOME
La base de datos reside en este volumen (siempre HOME en el directorio de bases de datos locales).

SQL_DCE
La base de datos reside en directorios DCE.

authentication
Tipo de autentificación. Los valores válidos son:

SQL_AUTHENTICATION_SERVER
La autentificación del nombre de usuario y la contraseña tiene lugar en el servidor.

SQL_AUTHENTICATION_CLIENT
La autentificación del nombre de usuario y la contraseña tiene lugar en el cliente.

SQL_AUTHENTICATION_DCE
La autentificación se realiza mediante los Servicios de seguridad de DCE.

SQL_AUTHENTICATION_KERBEROS
La autentificación se realiza mediante el Mecanismo de seguridad de Kerberos.

SQL_AUTHENTICATION_NOT_SPECIFIED
DB2 ya no requiere autentificación para conservarse en el directorio de bases de datos. Especifique este valor cuando se conecte a un elemento que no sea un servidor anterior (DB2 V2 o inferior).

SQL_AUTHENTICATION_SVR_ENCRYPT
Especifica que la autentificación se realiza en el nodo donde reside la base de datos de destino, y que la contraseña de autentificación se debe cifrar.

SQL_AUTHENTICATION_DATAENC
Especifica que la autentificación se realiza en el nodo donde reside la base de datos de destino y que las conexiones deben utilizar el cifrado de datos.

SQL_AUTHENTICATION_GSSPLUGIN
Especifica que la autentificación se realiza utilizando un mecanismo de seguridad externo basado en un plugin de la API de GSS.

glbdbname
Nombre global de la base de datos destino en el directorio global (DCE), si la entrada es de tipo SQL_DCE.

dceprincipal
El nombre de principal si la autentificación es de tipo DCE o KERBEROS.

cat_nodenum
Número de nodo de catálogo.

nodenum
Número de nodo.

althostname
El nombre de sistema principal o dirección IP del servidor alternativo donde se reconecta la base de datos durante una conmutación por anomalía.

altportnumber
El número de puerto del servidor alternativo donde se reconecta la base de datos durante una conmutación por anomalía.

Notas de uso

Todos los campos del almacenamiento intermedio de información de entradas del directorio están rellenados con blancos por la derecha.

Una db2DbDirGetNextEntry subsiguiente obtiene la entrada situada a continuación de la entrada actual.

Si se llama a db2DbDirGetNextEntry cuando no hay más entradas que explorar, se establece SQL1014N en SQLCA.

El valor de número devuelto por la API db2DbDirOpenScan puede utilizarse para explorar todo el directorio emitiendo llamadas db2DbDirGetNextEntry, de una en una, hasta que el número de exploraciones sea igual al número de entradas.

db2DbDirOpenScan - Iniciar una exploración del directorio de bases de datos locales o del sistema

Almacena en la memoria una copia del directorio de bases de datos locales o del sistema y devuelve el número de entradas del directorio. Esta copia representa una instantánea del estado que tenía directorio cuando se abrió. Esta copia no se actualiza, aunque el propio directorio se modifique más tarde.

Utilice la API db2DbDirGetNextEntry para examinar las entradas del directorio de bases de datos. Cierre la conexión mediante la API db2DbDirCloseScan. Esta acción elimina la copia del directorio de la memoria.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2DbDirOpenScan (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);
```

```
typedef SQL_STRUCTURE db2DbDirOpenScanStruct
{
   char *piPath;
   db2Uint16 oHandle;
   db2Uint16 oNumEntries;
} db2DbDirOpenScanStruct;

SQL_API_RC SQL_API_FN
  db2gDbDirOpenScan (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gDbDirOpenScanStruct
{
   db2Uint32 iPath_len;
   char *piPath;
   db2Uint16 oHandle;
   db2Uint16 oNumEntries;
} db2gDbDirOpenScanStruct;
```

Parámetros de la API db2DbDirOpenScan

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2DbDirOpenScanStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2DbDirOpenScanStruct

piPath Entrada. Nombre de la vía de acceso en la que reside el directorio de bases de datos locales. Si la vía de acceso especificada es un puntero NULL, se utiliza el directorio de bases de datos del sistema.

oHandle
Salida. Área de 2 bytes para el identificador devuelto. Este identificador se debe pasar a la API db2DbDirGetNextEntry para explorar las entradas de base de datos y a la API db2DbDirCloseScan para liberar los recursos.

oNumEntries
Salida. Área de 2 bytes donde se devuelve el número de entradas de directorio.

Parámetros específicos de la estructura de datos db2gDbDirOpenScanStruct

iPath_len
Entrada. Longitud en bytes del parámetro piPath.

Notas de uso

El espacio de almacenamiento asignado por esta API es liberado por la API db2DbDirCloseScan.

La API db2DbDirOpenScan se puede ejecutar varias veces para un mismo directorio, pero los resultados pueden no ser los mismos. El directorio puede cambiar entre una apertura y otra.

Puede haber un máximo de ocho exploraciones del directorio de bases de datos abiertas por proceso.

db2DropContact - Eliminar un contacto de la lista de contactos a los que se pueden enviar mensajes de notificación

Elimina un contacto de la lista de contactos. Los contactos son usuarios a los que se pueden enviar mensajes de notificación.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2DropContact (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2DropContactData
{
   char *piUserid;
   char *piPassword;
   char *piName;
} db2DropContactData;
```

Parámetros de la API db2DropContact

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2DropContactData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2DropContactData

piUserid
Entrada. Nombre del usuario.

piPassword
Entrada. Contraseña de piUserid.

piName
Entrada. Nombre del contacto que se debe descartar.

Notas de uso

Esta API no está soportada en UNIX y Linux. No obstante, podrá acceder a la misma función utilizando la interfaz de SQL.

db2DropContactGroup - Eliminar un grupo de contactos de la lista de contactos a los que se pueden enviar mensajes de notificación

Elimina un grupo de contactos de la lista de contactos. Un grupo de contactos contiene una lista de usuarios a los que se pueden enviar mensajes de notificación.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2DropContactGroup (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2DropContactData
{
   char *piUserid;
   char *piPassword;
   char *piName;
} db2DropContactData;
```

Parámetros de la API db2DropContactGroup

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2DropContactData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2DropContactData

piUserid
Entrada. Nombre del usuario.

piPassword
Entrada. Contraseña de piUserid.

piName
Entrada. Nombre del contacto que se debe descartar.

Notas de uso

Esta API no está soportada en UNIX y Linux. No obstante, podrá acceder a la misma función utilizando la interfaz de SQL.

db2Export - Exportar datos de una base de datos

Exporta datos de una base de datos a uno de varios formatos de archivo externos. El usuario especifica los datos que se deben exportar proporcionando una sentencia SELECT de SQL o proporcionando información jerárquica para tablas de tipo.

Autorización

Una de las siguientes:

- sysadm
- dbadm

o privilegio CONTROL o SELECT en cada tabla o vista participante. Esta función impone el control de acceso basado en etiquetas (LBAC). Los datos exportados pueden limitarse por las credenciales LBAC del llamador si los datos están protegidos por LBAC.

Conexión necesaria

Base de datos. Si se ha habilitado la conexión implícita, se establece una conexión con la base de datos por omisión.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2Export (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ExportStruct
{
   char *piDataFileName;
   struct sqlu_media_list *piLobPathList;
   struct sqlu_media_list *piLobFileList;
   struct sqldcol *piDataDescriptor;
   struct sqllob *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piMsgFileName;
   db2int16 iCallerAction;
   struct db2ExportOut *poExportInfoOut;
   struct db2ExportIn *piExportInfoIn;
   struct sqlu_media_list *piXmlPathList;
   struct sqlu_media_list *piXmlFileList;
} db2ExportStruct;

typedef SQL_STRUCTURE db2ExportIn
{
   db2Uint16 *piXmlSaveSchema;
} db2ExportIn;
```

```
typedef SQL_STRUCTURE db2ExportOut
{
   db2Uint64 oRowsExported;
} db2ExportOut;

SQL_API_RC SQL_API_FN
  db2gExport (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gExportStruct
{
   char *piDataFileName;
   struct sqlu_media_list *piLobPathList;
   struct sqlu_media_list *piLobFileList;
   struct sqldcol *piDataDescriptor;
   struct sqllob *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piMsgFileName;
   db2int16 iCallerAction;
   struct db2ExportOut *poExportInfoOut;
   db2Uint16 iDataFileNameLen;
   db2Uint16 iFileTypeLen;
   db2Uint16 iMsgFileNameLen;
   struct db2ExportIn *piExportInfoIn;
   struct sqlu_media_list *piXmlPathList;
   struct sqlu_media_list *piXmlFileList;
} db2gExportStruct;
```

Parámetros de la API db2Export

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2ExportStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2ExportStruct

piDataFileName
Entrada. Serie que contiene la vía de acceso y el nombre del archivo externo al que deben exportarse los datos.

piLobPathList
Entrada. Puntero a una estructura sqlu_media_list cuyo campo media_type se ha establecido en SQLU_LOCAL_MEDIA y cuya estructura sqlu_media_entry lista vías de acceso del cliente en que se almacenan los archivos LOB. Los datos LOB exportados se distribuirán de manera uniforme entre todas las vías de acceso listadas en la estructura sqlu_media_entry.

piLobFileList
Entrada. Puntero a una estructura sqlu_media_list cuyo campo media_type se ha establecido en SQLU_CLIENT_LOCATION y cuya estructura sqlu_location_entry contiene nombres de archivo base.

Cuando se ha agotado el espacio de nombres utilizando el primer nombre de la lista, la API utilizará el segundo nombre, y así sucesivamente.

Cuando se crean archivos LOB durante una operación de exportación, los nombres de archivo se construyen añadiendo el nombre base actual de esta lista a la vía de acceso actual (de piLobPathList) y, a continuación, añadiendo un número de secuencia de 3 dígitos y la extensión .lob. Por ejemplo, si la vía de acceso de LOB actual es el directorio /u/foo/lob/vía-acceso, el nombre de archivo LOB actual es bar y si se ha establecido el modificador de tipo de archivo LOBSINSEPFILES, los archivos LOB creados serán /u/foo/LOB/path/bar.001.lob, /u/foo/LOB/path/bar.002.lob, etcétera. Si el modificador de tipo de archivo LOBSINSEPFILES no se ha establecido, todos los documentos LOB se concatenarán y se incluirán en el archivo /u/foo/lob/path/bar.001.lob

piDataDescriptor
Entrada. Puntero a una estructura sqldcol que especifica los nombres de columnas del archivo de salida. El valor del campo dcolmeth determina cómo interpretará el programa de utilidad de exportación el resto de la información proporcionada en este parámetro. Los valores válidos para este parámetro (definidos en el archivo de cabecera sqlutil del directorio de inclusión) son:

SQL_METH_N
Nombres. Especifica los nombres de columna que se deben utilizar en el archivo de salida.

SQL_METH_D
Valor por omisión. Los nombres de columna existentes de la tabla que deben utilizarse en el archivo de salida. En este caso, se hará caso omiso tanto del número de columnas como de la matriz de especificación de columnas. Los nombres de columnas se obtienen de la salida de la sentencia SELECT especificada en piActionString.

piActionString
Entrada. Puntero a una estructura sqllob que contiene una sentencia SELECT de SQL dinámica válida. La estructura contiene un campo de cuatro bytes de longitud, seguido por los caracteres que forman la sentencia SELECT. La sentencia SELECT especifica los datos que deben extraerse de la base de datos y grabarse en el archivo externo.

Las columnas del archivo externo (de piDataDescriptor) y las columnas de base de datos de la sentencia SELECT, se comparan de acuerdo con su posición en la lista o en la estructura, respectivamente. La primera columna de datos seleccionada de la base de datos se coloca en la primera columna del archivo externo, y su nombre de columna se obtiene del primer elemento de la matriz de columnas externas.

piFileType
Entrada. Serie que indica el formato de los datos del archivo externo. Los formatos de archivos externos soportados (definidos en el archivo de cabecera sqlutil) son:

SQL_DEL
ASCII delimitado, para intercambio con dBase, BASIC y los programas IBM Personal Decision Series, y muchos otros gestores de bases de datos y gestores de archivos.

SQL_WSF
Formatos de hojas de trabajo para realizar intercambios con los programas Lotus Symphony y 1-2-3.

SQL_IXF
Versión de PC del formato de intercambio integrado, el método favorito de exportar datos de una tabla. Los datos exportados a este formato de archivo pueden importarse o cargarse más tarde en la misma tabla o en la tabla de otro gestor de bases de datos.

piFileTypeMod
Entrada. Puntero a una estructura que contiene un campo de dos bytes de longitud seguido por una matriz de caracteres que especifica una o más opciones de proceso. Si este puntero es NULL, o si la estructura a la que apunta tiene cero caracteres, esta acción se interpreta como selección de una especificación por omisión.

No todas las opciones se pueden utilizar con todos los tipos de archivos soportados. Consulte a continuación el enlace relacionado ″Modificadores de tipo de archivo para el programa de utilidad de exportación″.

piMsgFileName
Entrada. Serie que contiene el destino de mensajes de error, mensajes de aviso y mensajes informativos devueltos por el programa de utilidad. Puede ser la vía de acceso y el nombre de un archivo del sistema operativo o un dispositivo estándar. Si el archivo ya existe, la información se añadirá. Si no existe, el archivo se creará.

iCallerAction
Entrada. Acción solicitada por el llamador. Los valores válidos (definidos en el archivo de cabecera sqlutil, ubicado en el directorio de inclusión) son:

SQLU_INITIAL
Llamada inicial. Este valor debe utilizarse en la primera llamada a la API. Si la llamada inicial o las llamadas posteriores devuelven el control y exigen a la aplicación que realiza la llamada que lleve a cabo alguna acción antes de completar la operación de exportación solicitada, la acción del llamador debe establecerse en una de las siguientes:

SQLU_CONTINUE
Continúa el proceso. Este valor sólo puede utilizarse en llamadas posteriores a la API, después de que la llamada inicial haya devuelto el control al programa de utilidad que solicitaba una entrada del usuario (por ejemplo, para responder a una condición de fin de cinta). Especifica que la acción del usuario solicitada por el programa de utilidad se ha completado y el programa de utilidad puede continuar procesando la petición inicial.

SQLU_TERMINATE
Termina el proceso. Este valor sólo puede utilizarse en llamadas posteriores a la API, después de que la llamada inicial haya devuelto el control al programa de utilidad que solicitaba una entrada del usuario (por ejemplo, para responder a una condición de fin de cinta). Especifica que la acción del usuario que había solicitado el programa de utilidad no se ha realizado y que éste ha de finalizar el proceso de la petición inicial.

poExportInfoOut
Puntero a la estructura db2ExportOut.

piExportInfoIn
Entrada. Puntero a la estructura db2ExportIn.

piXmlPathList
Entrada. Puntero a una estructura sqlu_media_list cuyo campo media_type se ha establecido en SQLU_LOCAL_MEDIA y cuya estructura sqlu_media_entry lista vías de acceso del cliente en que se almacenan los archivos XML. Los datos XML exportados se distribuirán de manera uniforme entre todas las vías de acceso listadas en la estructura sqlu_media_entry.

piXmlFileList
Entrada. Puntero a una estructura sqlu_media_list cuyo campo media_type se ha establecido en SQLU_CLIENT_LOCATION y cuya estructura sqlu_location_entry contiene nombres de archivo base.

Cuando se ha agotado el espacio de nombres utilizando el primer nombre de la lista, la API utilizará el segundo nombre, y así sucesivamente. Cuando se crean archivos XML durante una operación de exportación, los nombres de archivo se construyen añadiendo el nombre base actual de esta lista a la vía de acceso actual (de piXmlFileList) y, a continuación, añadiendo un número de secuencia de 3 dígitos y la extensión .xml. Por ejemplo, si la vía de acceso de XML actual es el directorio /u/foo/xml/path, el nombre de archivo XML actual es bar y si se ha establecido el modificador de tipo de archivo XMLINSEPFILES, los archivos XML creados serán /u/foo/xml/path/bar.001.xml, /u/foo/xml/path/bar.002.xml, etcétera. Si el modificador de tipo de archivo XMLINSEPFILES no se ha establecido, todos los documentos XML se concatenarán y se incluirán en el archivo /u/foo/xml/path/bar.001.xml

Parámetros de la estructura de datos db2ExportIn

piXmlSaveSchema
Entrada. Indica que el identificador de SQL del esquema XML utilizado para validar todos los documentos XML exportados, debe guardarse en el archivo de datos exportados. Los valores posibles son TRUE y FALSE.

Parámetros de la estructura de datos db2ExportOut

oRowsExported
Salida. Devuelve el número de registros exportados al archivo de destino.

Parámetros específicos de la estructura de datos db2gExportStruct

iDataFileNameLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud del nombre del archivo de datos, expresada en bytes.

iFileTypeLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud del tipo de archivo, expresada en bytes.

iMsgFileNameLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud del nombre del archivo de mensajes, expresada en bytes.

Notas de uso

Antes de iniciar una operación de exportación, asegúrese de completar todas las operaciones de tabla y de liberar todos los bloqueos de una de estas dos formas:

- Cierre todos los cursores abiertos que se han definido con la cláusula WITH HOLD y confirme los cambios en los datos ejecutando la sentencia COMMIT.
- Retrotraiga los cambios en los datos ejecutando la sentencia ROLLBACK.

Se pueden utilizar alias de tabla en la sentencia SELECT.

Los mensajes colocados en el archivo de mensajes incluyen la información devuelta del servicio de recuperación de mensajes. Cada mensaje empieza en una línea nueva.

Si el programa de utilidad de exportación genera algún aviso, el mensaje se grabará en un archivo de mensajes o en la salida estándar, si no se ha especificado ninguno.

Si el número de columnas (campo dcolnum de la estructura sqldcol) de la matriz de nombres de columnas externas, piDataDescriptor, no es igual al número de columnas generadas por la sentencia SELECT, se emitirá un mensaje de aviso. En este caso, el número de columnas grabadas en el archivo externo es el menor de ambos números. Las columnas de base de datos o los nombres de columnas externas de más no se utilizarán para generar el archivo de salida.

Si el módulo db2uexpm.bnd o cualquier otro archivo .bnd enviado están vinculados manualmente, no debe utilizarse la opción de formato del vinculador.

DB2 Connect se puede utilizar para exportar tablas de servidores DRDA, como DB2 para z/OS y OS/390, DB2 para VM y VSE, y DB2 para System i. Sólo se da soporte a la exportación de PC/IXF.

Se deberá utilizar la importación PC/IXF para mover datos entre bases de datos. Si los datos de tipo carácter que contienen separadores de filas se exportan a un archivo ASCII delimitado (DEL) y se procesan con un programa de transferencia de texto, los campos que contengan separadores de filas se acortarán o se ampliarán.

El programa de utilidad de exportación no creará archivos PC/IXF de múltiples partes cuando se invoque desde un sistema AIX.

En el archivo PC/IXF se incluyen definiciones de índice para una tabla si el contenido de una sola tabla de base de datos se exporta a un archivo PC/IXF con un parámetro piActionString que empieza por SELECT * FROM nombretabla y en el parámetro piDataDescriptor se especifican nombres por omisión. Los índices no se guardan en el caso de las vistas ni si la cláusula SELECT de piActionString incluye una condición de unión. Las cláusulas WHERE, GROUP BY o HAVING del parámetro piActionString no impedirán que se guarden los índices. En todos estos casos, si se exporta a partir de tablas con tipo, se deberá exportar la jerarquía entera.

El programa de utilidad de exportación almacenará el atributo NOT NULL WITH DEFAULT de la tabla en un archivo IXF si la sentencia SELECT proporcionada está en el formato SELECT * FROM nombretabla.

Cuando se exportan tablas con tipo, sólo se pueden expresar sentencias de subselección especificando el nombre de tabla de destino y la cláusula WHERE. La selección completa y la sentencia-select no se pueden especificar cuando se exporta una jerarquía.

Para formatos de archivos distintos de IXF, se recomienda especificar la lista de orden transversal porque indica a DB2 cómo atravesar la jerarquía y qué subtablas se deben exportar. Si no se especifica dicha lista, se exportan todas las tablas de la jerarquía y el orden por omisión es el orden de OUTER. La alternativa consiste en utilizar el orden por omisión, que es el orden proporcionado por la función OUTER.

Nota: Utilice el mismo orden transversal durante una operación de importación. El programa de utilidad de carga no soporta la carga de jerarquías o de subjerarquías.

Sintaxis de la API para REXX

```
EXPORT :stmt TO datafile OF filetype
[MODIFIED BY :filetmod] [USING :dcoldata]
MESSAGES msgfile [ROWS EXPORTED :number]


CONTINUE EXPORT


STOP EXPORT
```

Parámetros de la API para REXX

stmt Variable del lenguaje principal de REXX que contiene una sentencia SELECT de SQL dinámica válida. La sentencia especifica los datos que deben extraerse de la base de datos.

datafile
Nombre del archivo en el que se exportarán los datos.

filetype
Formato de los datos en el archivo de exportación. Los formatos de archivo soportados son:

- **DEL** ASCII delimitado
- **WSF** Formato de hoja de trabajo
- **IXF** Versión de PC del formato de intercambio integrado.

filetmod
Variable del lenguaje principal que contiene opciones de proceso adicionales.

dcoldata
Variable compuesta del lenguaje principal de REXX que contiene los nombres de columnas que se utilizarán en el archivo de exportación. En la información que sigue a continuación, XXX representa el nombre de la variable del lenguaje principal:

- **XXX.0** Número de columnas (número de elementos en el resto de la variable).
- **XXX.1** Nombre de la primera columna.
- **XXX.2** Nombre de la segunda columna.
- **XXX.3** y así sucesivamente.

Si este parámetro es NULL, o no se ha especifica un valor para dcoldata, el programa de utilidad utilizará los nombres de columna de la tabla de la base de datos.

msgfile
Archivo, vía de acceso o nombre de dispositivo al que se enviarán los mensajes de error y de aviso.

number
Variable del lenguaje principal que contendrá el número de filas exportadas.

db2GetAlertCfg - Obtener los valores de configuración de alertas para los indicadores de salud

Devuelve los valores de configuración de alertas para los indicadores de salud.

Autorización

Ninguna

Conexión necesaria

Instancia. Si no existe ninguna conexión de instancia, se creará una conexión de instancia por omisión.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2GetAlertCfg (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2GetAlertCfgData
{
   db2Uint32 iObjType;
   char *piObjName;
   db2Uint32 iDefault;
   char *piDbName;
   db2Uint32 ioNumIndicators;
   struct db2GetAlertCfgInd *pioIndicators;
} db2GetAlertCfgData;

typedef SQL_STRUCTURE db2GetAlertCfgInd
{
   db2Uint32 ioIndicatorID;
   sqlint64 oAlarm;
   sqlint64 oWarning;
   db2Uint32 oSensitivity;
   char *poFormula;
   db2Uint32 oActionEnabled;
   db2Uint32 oCheckThresholds;
   db2Uint32 oNumTaskActions;
   struct db2AlertTaskAction *poTaskActions;
   db2Uint32 oNumScriptActions;
   struct db2AlertScriptAction *poScriptActions;
   db2Uint32 oDefault;
} db2GetAlertCfgInd;

typedef SQL_STRUCTURE db2AlertTaskAction
{
   char *pTaskName;
```

```
   db2Uint32 condition;
   char *pUserID;
   char *pPassword;
   char *pHostName;
} db2AlertTaskAction;

typedef SQL_STRUCTURE db2AlertScriptAction
{
   db2Uint32 scriptType;
   db2Uint32 condition;
   char *pPathName;
   char *pWorkingDir;
   char *pCmdLineParms;
   char stmtTermChar;
   char *pUserID;
   char *pPassword;
   char *pHostName;
} db2AlertScriptAction;
```

Parámetros de la API db2GetAlertCfg

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2GetAlertCfgData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2GetAlertCfgData

iObjType
Entrada. Especifica el tipo de objeto para el que se solicita la configuración. Los valores válidos son:

- DB2ALERTCFG_OBJTYPE_DBM
- DB2ALERTCFG_OBJTYPE_DATABASES
- DB2ALERTCFG_OBJTYPE_TABLESPACES
- DB2ALERTCFG_OBJTYPE_TS_CONTAINERS
- DB2ALERTCFG_OBJTYPE_DATABASE
- DB2ALERTCFG_OBJTYPE_TABLESPACE
- DB2ALERTCFG_OBJTYPE_TS_CONTAINER

piObjName
Entrada. Nombre del espacio de tablas o contenedor de espacio de tablas cuando el tipo de objeto, iObjType, es DB2ALERTCFG_OBJTYPE_TABLESPACE o DB2ALERTCFG_OBJTYPE_TS_CONTAINER.

iDefault
Entrada. Indica que se deben obtener los valores de configuración de la instalación por omisión.

piDbname
Entrada. Nombre de alias de la base de datos para la que se solicita la configuración cuando el tipo de objeto, iObjType, es DB2ALERTCFG_OBJTYPE_TS_CONTAINER, DB2ALERTCFG_OBJTYPE_TABLESPACE o DB2ALERTCFG_OBJTYPE_DATABASE.

ioNumIndicators
Este parámetro se puede utilizar como parámetro de entrada o salida.

Entrada. Indica el número de pioIndicators enviados cuando se solicitan valores para un subconjunto de indicadores de salud.

Salida. Indica el número total de indicadores de salud devueltos por la API.

pioIndicators
Puntero a la estructura db2GetAlertCfgInd. Si su valor es NULL, se devuelven todos los indicadores de salud del objeto.

Parámetros de la estructura de datos db2GetAlertCfgInd

ioIndicatorID
El indicador de salud (definido en sqlmon.h).

oAlarm
Salida. Valor de umbral de alarma del indicador de salud. Este valor sólo es válido para indicadores de salud basados en un valor de umbral.

oWarning
Salida. Valor de umbral de aviso del indicador de salud. Este valor sólo es válido para indicadores de salud basados en un valor de umbral.

oSensitivity
Salida. Periodo de tiempo en el que debe permanecer un valor de indicador de salud dentro de una zona umbral para que se registre la correspondiente condición de alarma o de aviso.

poFormula
Salida. Serie que representa la fórmula utilizada para calcular el valor del indicador de salud.

oActionEnabled
Salida. Si el valor es TRUE, cuando se alcanza un valor umbral se desencadenan las acciones de alerta definidas en poTaskActions o poScriptActions. Si el valor es FALSE, no se invocará ninguna de las acciones definidas.

oCheckThresholds
Salida. Si el valor es TRUE, se evaluarán las violaciones de umbral o los cambios de estado. Si no se evalúan las violaciones o estados de umbral, no se emitirán alertas y no se invocarán acciones de alerta aunque el valor de oActionEnabled sea TRUE.

oNumTaskActions
Salida. Número de acciones de alerta de tarea de la matriz pTaskAction.

poTaskActions
Puntero a la estructura db2AlertTaskAction.

oNumScriptActions
Salida. Número de acciones de script de la matriz poScriptActions.

poScriptActions
Puntero a la estructura db2AlertScriptAction.

oDefault
Salida. Indica si los valores actuales se heredan del valor por omisión. Se

establece en el valor TRUE para indicar que los valores actuales se heredan de la configuración por omisión; de lo contrario, se establece en el valor FALSE.

Parámetros de la estructura de datos db2AlertTaskAction

pTaskname
Nombre de la tarea.

condition
Condición para la que se ejecuta la acción.

pUserID
Cuenta de usuario utilizada para ejecutar el script.

pPassword
Contraseña de la cuenta de usuario pUserId.

pHostName
Nombre del sistema principal en el que ejecutar el script. Esto es aplicable a la tarea y al script.

Script Nombre del sistema principal donde reside el script y desde donde se ejecutará.

Task Nombre del sistema principal donde reside el planificador de tareas.

Parámetros de la estructura de datos db2AlertScriptAction

scriptType
Especifica el tipo de script. Los valores válidos son:

- DB2ALERTCFG_SCRIPTTYPE_DB2CMD
- DB2ALERTCFG_SCRIPTTYPE_OS

condition
Condición en la que se ejecuta la acción. Los valores válidos para los indicadores de salud basados en valores umbrales son:

- DB2ALERTCFG_CONDITION_ALL
- DB2ALERTCFG_CONDITION_WARNING
- DB2ALERTCFG_CONDITION_ALARM

Para los indicadores de salud basados en estados, utilice el valor numérico definido en sqlmon.

pPathname
Vía de acceso absoluta del script.

pWorkingDir
Vía de acceso absoluta del directorio donde se debe ejecutar el script.

pCmdLineParms
Parámetros de línea de mandatos que se deben pasar al script cuando éste se invoque. Aplicable opcionalmente para DB2ALERTCFG_SCRIPTTYPE_OS solamente.

stmtTermChar
Carácter utilizado en el script para finalizar sentencias. Aplicable opcionalmente para DB2ALERTCFG_SCRIPTTYPE_DB2CMD solamente.

pUserID
Cuenta de usuario utilizada para ejecutar el script.

pPassword
Contraseña de la cuenta de usuario pUserId.

pHostName
Nombre del sistema principal en el que ejecutar el script. Esto es aplicable a la tarea y al script.

Script Nombre del sistema principal donde reside el script y desde donde se ejecutará.

Task Nombre del sistema principal donde reside el planificador de tareas.

Notas de uso

Si se deja pioIndicators en NULL, se devuelven todos los indicadores de salud para ese objeto. Este parámetro se puede establecer en una matriz de estructuras db2GetAlertCfgInd con ioIndicatorID establecido en el indicador de salud para el que se desee la configuración. Cuando se utilice de esta manera, asegúrese de establecer ioNumIndicators en la longitud de matriz de entrada y de establecer el resto de los campos en db2GetAlertCfgInd en 0 o en NULL.

Cuando la API db2GetAlertCfg finaliza sin ningún error, debe utilizar la API db2GetAlertCfgFree para liberar toda la memoria de este puntero que es asignada por el motor. Vea db2ApiDf.h, en el directorio de inclusión, para obtener información sobre la API db2GetAlertCfgFree.

db2GetAlertCfgFree - Liberar la memoria asignada por la API db2GetAlertCfg

Libera la memoria asignada por la API db2GetAlertCfg.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2GetAlertCfgFree (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);
```

Parámetros de la API db2GetAlertCfgFree

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2GetAlertCfgData.

pSqlca
Salida. Puntero a la estructura sqlca.

db2GetContactGroup - Obtener la lista de contactos de un solo grupo de contactos al que se puedan enviar mensajes de notificación

Devuelve los contactos incluidos en un solo grupo de contactos. Los contactos son usuarios a los que se pueden enviar mensajes de notificación. Los contactos se pueden definir localmente en el sistema o en una lista global. El valor del parámetro de configuración contact_host del Servidor de administración de DB2 (DAS) determina si la lista es local o global.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2GetContactGroup (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ContactGroupData
{
   char *pGroupName;
   char *pDescription;
   db2Uint32 numContacts;
   struct db2ContactTypeData *pContacts;
} db2ContactGroupData;

typedef SQL_STRUCTURE db2ContactTypeData
{
   db2Uint32 contactType;
   char *pName;
} db2ContactTypeData;
```

Parámetros de la API db2GetContactGroup

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2ContactGroupData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2ContactGroupData

pGroupName
Entrada. Nombre del grupo que se debe recuperar.

pDescription
Descripción del grupo.

numContacts
Número de contactos.

pContacts
Puntero a la estructura db2ContactTypeData. El usuario debe preasignar los campos pGroupName, pDescription, pContacts y pContacts.pName con sus respectivos tamaños máximos. Invoque db2GetContactGroup con numContacts=0 y pContacts=NULL para que la API devuelva en numContacts la longitud necesaria para pContacts.

Parámetros de la estructura de datos db2ContactTypeData

contactType
Especifica el tipo de contacto. Los valores válidos son:

- DB2CONTACT_SINGLE
- DB2CONTACT_GROUP

pName
Nombre del grupo de contactos, o nombre del contacto si contactType es DB2CONTACT_SINGLE.

Notas de uso

Esta API no está soportada en UNIX y Linux. No obstante, podrá acceder a la misma función utilizando la interfaz de SQL.

db2GetContactGroups - Obtener la lista de grupos de contactos a los que se puedan enviar mensajes de notificación

Devuelve la lista de grupos de contactos. Los contactos son usuarios a los que se pueden enviar mensajes de notificación. Los grupos de contactos se pueden definir localmente en el sistema o en una lista global. El valor del parámetro de configuración contact_host del Servidor de administración de DB2 (DAS) determina si la lista es local o global.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2GetContactGroups (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2GetContactGroupsData
```

```
{
   db2Uint32 ioNumGroups;
   struct db2ContactGroupDesc *poGroups;
} db2GetContactGroupsData;

typedef SQL_STRUCTURE db2ContactGroupDesc
{
   char *poName;
   char *poDescription;
} db2ContactGroupDesc;
```

Parámetros de la API db2GetContactGroups

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2GetContactGroupsData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2GetContactGroupsData

ioNumGroups
Número de grupos. Si oNumGroups = 0 y poGroups = NULL, este parámetro contiene el número estructuras db2ContactGroupDesc que es necesario que existan en poGroups.

poGroups
Salida. Puntero a la estructura db2ContactGroupDesc.

Parámetros de la estructura de datos db2ContactGroupDesc

poName
Salida. Nombre del grupo. El llamador debe preasignar este parámetro con el tamaño máximo respectivo.

poDescription
Salida. Descripción del grupo. El llamador debe preasignar este parámetro con el tamaño máximo respectivo.

Notas de uso

Esta API no está soportada en UNIX y Linux. No obstante, podrá acceder a la misma función utilizando la interfaz de SQL.

db2GetContacts - Obtener la lista de contactos a los que se pueden enviar mensajes de notificación

Devuelve la lista de contactos. Los contactos son usuarios a los que se pueden enviar mensajes de notificación. Los contactos se pueden definir localmente en el sistema o en una lista global. El valor del parámetro de configuración contact_host del Servidor de administración de DB2 (DAS) determina si la lista es local o global.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

db2ApiDf.h

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2GetContacts (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2GetContactsData
{
   db2Uint32 ioNumContacts;
   struct db2ContactData *poContacts;
} db2GetContactsData;

typedef SQL_STRUCTURE db2ContactData
{
   char *pName;
   db2Uint32 type;
   char *pAddress;
   db2Uint32 maxPageLength;
   char *pDescription;
} db2ContactData;
```

Parámetros de la API db2GetContacts

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2GetContactsData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2GetContactsData

ioNumContacts
Número de contactos.

poContacts
Salida. Puntero a la estructura db2ContactData. El usuario debe preasignar los campos poContacts, pocontacts.pAddress, pocontacts.pDescription y pocontacts.pName con sus respectivos tamaños máximos. Invoque db2GetContacts con numContacts=0 y poContacts=NULL para que la API devuelva en numContacts la longitud necesaria para poContacts.

Parámetros de la estructura de datos db2ContactData

pName
Nombre del contacto.

type Especifica el tipo de contacto. Los valores válidos son:

- DB2CONTACT_EMAIL
- DB2CONTACT_PAGE

pAddress
Dirección del parámetro type.

maxPageLength
Longitud máxima del mensaje para cuando el valor de type es DB2CONTACT_PAGE.

pDescription
Descripción del contacto proporcionada por el usuario.

Notas de uso

Esta API no está soportada en UNIX y Linux. No obstante, podrá acceder a la misma función utilizando la interfaz de SQL.

db2GetHealthNotificationList - Obtener la lista de contactos a los que se puedan enviar notificaciones de alerta de salud

Devuelve la lista de contactos y/o de grupos de contactos a los que se notifica acerca del estado de una instancia. Una lista de contactos se compone de direcciones de correo electrónico y de direcciones de Internet de buscapersonas de individuos a los que hay que notificar cuando se cumplan determinadas condiciones de salud que no sean normales para una instancia o para cualquiera de sus objetos de base de datos.

Autorización

Ninguna

Conexión necesaria

Instancia. Si no existe ninguna conexión de instancia, se creará una conexión de instancia por omisión.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2GetHealthNotificationList (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2GetHealthNotificationListData
{
   db2Uint32 ioNumContacts;
   struct db2ContactTypeData *poContacts;
} db2GetHealthNotificationListData;

typedef SQL_STRUCTURE db2ContactTypeData
{
   db2Uint32 contactType;
   char *pName;
} db2ContactTypeData;
```

Parámetros de la API db2GetHealthNotificationList

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2GetHealthNotificationListData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2GetHealthNotificationListData

ioNumContacts
Número de contactos. Si la API se invocó con poContact = NULL, ioNumContacts será igual al número de contactos que el usuario debe asignar para que la llamada a la API sea satisfactoria.

poContacts
Salida. Puntero a la estructura db2ContactTypeData.

Parámetros de la estructura de datos db2ContactTypeData

contactType
Especifica el tipo de contacto. Los valores válidos son:

- DB2CONTACT_SINGLE
- DB2CONTACT_GROUP

pName
Nombre del grupo de contactos, o nombre del contacto si contactType es DB2CONTACT_SINGLE.

db2GetRecommendations - Obtener recomendaciones para resolver un indicador de salud en estado de alerta

Obtiene un conjunto de recomendaciones para resolver un indicador de estado en estado de alerta para un objeto determinado. - Las recomendaciones se proporcionan en forma de documento XML.

Autorización

Ninguna

Conexión necesaria

Instancia. Si no existe ninguna conexión de instancia, se creará una conexión de instancia por omisión.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2GetRecommendations (
   db2Uint32 versionNumber,
   void * pParmStruct,
```

```
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2GetRecommendationsData
{
   db2Uint32 iSchemaVersion;
   db2Uint32 iNodeNumber;
   db2Uint32 iIndicatorID;
   db2Uint32 iObjType;
   char *piObjName;
   char *piDbName;
   char *poRecommendation;
} db2GetRecommendationsData;
```

Parámetros de la API db2GetRecommendations

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2GetRecommendationsData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2GetRecommendationsData

iSchemaVersion
Entrada. ID de versión del esquema utilizado para representar el documento XML. El documento de recomendaciones solamente contiene elementos o atributos que se definieron para esa versión de esquema determinada. Establezca el valor de este parámetro en:
`DB2HEALTH_RECSCHEMA_VERSION8_2`

iNodeNumber
Entrada. Especifica el número de partición de base de datos en el que el indicador de salud (HI) ha entrado en un estado de alerta. Utilice la constante SQLM_ALL_NODES para recuperar recomendaciones para un objeto en un HI determinado a través de todas las particiones de base de datos. Si el HI contiene las mismas recomendaciones en diferentes particiones de base de datos, dichas recomendaciones se agruparán en un solo conjunto de recomendaciones, donde el problema es el grupo de HI en diferentes particiones de base de datos y las recomendaciones se aplican a todos estos HI. Para recuperar recomendaciones en la partición actual de base de datos, utilice el valor constante SQLM_CURRENT_NODE. Para instancias autónomas, se debe utilizar SQLM_CURRENT_NODE.

iIndicatorID
Entrada. Indicador de salud que ha entrado en un estado de alerta para el cual se solicita una recomendación. Los valores se externalizan en el archivo de cabecera sqlmon.h del directorio de inclusión.

iObjType
Entrada. Especifica el tipo de objeto en el que el indicador de salud (identificado por iIndicatorID) ha entrado en un estado de alerta. Los valores válidos son:

- DB2HEALTH_OBJTYPE_DBM
- DB2HEALTH_OBJTYPE_DATABASE
- DB2HEALTH_OBJTYPE_TABLESPACE

- DB2HEALTH_OBJTYPE_TS_CONTAINER

piObjName
Entrada. El nombre del espacio de tablas o contenedor de espacio de tablas cuando parámetro de tipo de objeto, iObjType, se establece en DB2HEALTH_OBJTYPE_TABLESPACE o en DB2HEALTH_OBJTYPE_TS_CONTAINER. Especifique NULL, en caso de que no sea necesario. En el caso de un contenedor de espacio de tablas, el nombre de objeto se especifica como <nombre de espacio de tablas>.<nombre de contenedor>.

piDbname
Entrada. Nombre de alias para la base de datos en la que el HI ha entrado en un estado de alerta cuando el parámetro de tipo de objeto iObjType es DB2HEALTH_OBJTYPE_TS_CONTAINER, DB2HEALTH_OBJTYPE_TABLESPACE o DB2HEALTH_OBJTYPE_DATABASE. En caso contrario, especifique NULL.

poRecommendation
Salida. Puntero de caracteres que se establecerá con la dirección de un almacenamiento intermedio en la memoria que contiene el texto de recomendación, con formato de documento XML según el esquema proporcionado en sqllib/misc/DB2RecommendationSchema.xsd. El documento XML estará codificado en UTF-8 y el texto del documento estará en el entorno local del llamador.

El atributo xml:lang del nodo DB2_HEALTH se establecerá con el idioma de cliente adecuado. Deberá considerarse la API como una fuente de confianza y el documento XML no deberá validarse. XML se utiliza como medio de estructurar los datos de salida. Toda la memoria bajo este puntero es asignada por el motor y se debe liberar con una llamada a db2GetRecommendationsFree cada vez que db2GetRecommendations finaliza sin ningún error.

Notas de uso

- Invoque esta API para recuperar un conjunto de recomendaciones para resolver una alerta de salud en un objeto DB2 determinado. Si el indicador de salud de entrada no está en un estado de alerta para el objeto identificado, se devolverá un error.
- - Las recomendaciones se proporcionan en forma de documento XML, y pueden contener información sobre acciones y scripts que se pueden ejecutar para resolver la alerta. Los scripts devueltos por la API se deben ejecutar en la instancia en la que el indicador de salud ha entrado en el estado de alerta. Para obtener información sobre la estructura y el contenido del documento XML de recomendaciones devuelto, consulte el esquema en sqllib/misc/DB2RecommendationSchema.xsd.
- Toda la memoria asignada por el motor y devuelta por esta función (el documento de recomendaciones) se debe liberar con una llamada a db2GetRecommendationsFree cada vez que db2GetRecommendations finaliza sin ningún error.

db2GetRecommendationsFree - Liberar la memoria asignada por la API db2GetRecommendations

Libera la memoria asignada por la API db2GetRecommendations.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2GetRecommendationsFree (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);
```

Parámetros de la API db2GetRecommendationsFree

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2GetRecommendationsData.

pSqlca
Salida. Puntero a la estructura sqlca.

db2GetSnapshot - Obtener una instantánea del estado operacional del gestor de bases de datos

Recopila información de supervisión del gestor de bases de datos y la devuelve a un almacenamiento intermedio de datos asignado por el usuario. La información devuelta representa una instantánea del estado operacional que tenía el gestor de bases de datos cuando se invocó la API.

Ámbito

Esta API puede devolver información para el servidor de particiones de base de datos de la instancia o para todas las particiones de base de datos de la instancia.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint
- sysmon

Conexión necesaria

Instancia. Si no existe ninguna conexión de instancia, se creará una conexión de instancia por omisión.

Para obtener una instantánea de una instancia remota (o de una instancia local diferente), es necesario conectarse primero a dicha instancia.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2GetSnapshot (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2GetSnapshotData
{
   void *piSqlmaData;
   struct sqlm_collected *poCollectedData;
   void *poBuffer;
   db2Uint32 iVersion;
   db2Uint32 iBufferSize;
   db2Uint32 iStoreResult;
   db2int32 iNodeNumber;
   db2Uint32 *poOutputFormat;
   db2Uint32 iSnapshotClass;
} db2GetSnapshotData;

SQL_API_RC SQL_API_FN
  db2gGetSnapshot (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gGetSnapshotData
{
   void *piSqlmaData;
   struct sqlm_collected *poCollectedData;
   void *poBuffer;
   db2Uint32 iVersion;
   db2Uint32 iBufferSize;
   db2Uint32 iStoreResult;
   db2int32 iNodeNumber;
   db2Uint32 *poOutputFormat;
   db2Uint32 iSnapshotClass;
} db2gGetSnapshotData;
```

Parámetros de la API

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct. Para utilizar la estructura tal como se describe más arriba, especifique db2Version810 o una versión más moderna. Si desea utilizar una versión diferente de esta estructura, vea la lista completa de versiones soportadas en el archivo de cabecera db2ApiDf.h del directorio de inclusión. Debe utilizar la versión de la estructura db2GetSnapshotData correspondiente al número de versión que especifique.

pParmStruct
Entrada/Salida. Puntero a la estructura db2GetSnapshotData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2GetSnapshotData

piSqlmaData
Entrada. Puntero a la estructura sqlma (monitor area) asignada por el usuario o a la estructura de datos de petición ″poRequestData″, creada y devuelta por la API db2AddSnapshotRequest. La estructura especifica el tipo o tipos de datos de instantánea que se deben recopilar. Si se utiliza un puntero a la estructura sqlma, el número de versión pasado en el parámetro versionNumber a la API db2GetSnapshot debe ser menor que db2Version900 (por ejemplo, db2Version810, db2Version822). Si se utiliza un puntero a la estructura de datos de petición devuelta por la API db2AddSnapshotRequest en el parámetro poRequestData, se debe pasar el valor db2Version900 en el parámetro versionNumber de la API db2GetSnapshot.

poCollectedData
Salida. Puntero a la estructura sqlm_collected en la que el supervisor de bases de datos entrega estadísticas de resumen y el número de cada tipo de estructura de datos devuelta en el área de almacenamiento intermedio.

Nota: Esta estructura sólo se utiliza para corrientes de datos anteriores a la Versión 6. Sin embargo, si se realiza una llamada de instantánea a un servidor remoto de una versión anterior, se debe pasar esta estructura para procesar los datos. Por tanto, es recomendable pasar siempre este parámetro.

poBuffer
Salida. Puntero al área de datos definida por el usuario en la que se devuelve información de instantánea.

iVersion
Entrada. ID de versión de los datos del supervisor de bases de datos que se deben recopilar. El supervisor de bases de datos solamente devuelve datos que estaban disponibles para la versión solicitada. Establezca este parámetro en una de las constantes siguientes:

- SQLM_DBMON_VERSION1
- SQLM_DBMON_VERSION2
- SQLM_DBMON_VERSION5
- SQLM_DBMON_VERSION5_2
- SQLM_DBMON_VERSION6
- SQLM_DBMON_VERSION7
- SQLM_DBMON_VERSION8
- SQLM_DBMON_VERSION9
- SQLM_DBMON_VERSION9_5

Nota: Las constantes SQLM_DBMON_VERSION5_2 y anteriores han quedado obsoletas y es posible que se eliminen en un futuro release de DB2.

iBufferSize
Entrada. Longitud de los datos del almacenamiento intermedio. Utilice la API db2GetSnapshotSize para calcular el tamaño aproximado de este almacenamiento intermedio. Si éste no es suficientemente grande, se emite un aviso, junto con la información que estará en el almacenamiento intermedio asignado. Tal vez sea necesario volver a calcular el tamaño del almacenamiento intermedio y llamar de nuevo a la API.

iStoreResult
Entrada. Indicador establecido en el valor constante TRUE o FALSE, dependiendo de si los resultados de la instantánea se van a almacenar en el servidor DB2 para su visualización mediante SQL. El valor de este parámetro debe ser TRUE solamente cuando la instantánea se toma a través de una conexión de base de datos y cuando uno de los tipos de instantánea existentes en sqlma es SQLMA_DYNAMIC_SQL.

iNodeNumber
Entrada. El nodo al que debe enviarse la petición. De acuerdo con este valor, la petición se procesará para el nodo actual, para todos los nodos o para un nodo especificado por el usuario. Los valores válidos son:

- SQLM_CURRENT_NODE
- SQLM_ALL_NODES. Este valor solamente está permitido si el valor del parámetro iVersion es SQLM_DBMON_VERSION7 o una versión más moderna.
- valor de nodo

Nota: Para instancias autónomas, se debe utilizar el valor SQLM_CURRENT_NODE.

poOutputFormat
Formato de la corriente de datos devuelta por el servidor. Puede tener uno de estos valores:

- SQLM_STREAM_STATIC_FORMAT
- SQLM_STREAM_DYNAMIC_FORMAT

iSnapshotClass
Entrada. Calificador de clase de la instantánea. Los valores válidos (definidos en el archivo de cabecera sqlmon, situado en el directorio de inclusión) son:

- SQLM_CLASS_DEFAULT para una instantánea estándar
- SQLM_CLASS_HEALTH para una instantánea de estado
- SQLM_CLASS_HEALTH_WITH_DETAIL para una instantánea de estado que incluye detalles adicionales

Notas de uso

Si se especifica un alias de una base de datos que reside en una instancia diferente, se devuelve un mensaje de error.

Para obtener una instantánea de estado que incluya información completa, utilice el campo AGENT_ID de la estructura de datos SQLMA.

db2GetSnapshotSize - Calcular el tamaño del almacenamiento intermedio de salida necesario para la API db2GetSnapshot

Calcula el tamaño del almacenamiento intermedio necesario para la API db2GetSnapshot.

Ámbito

Esta API puede afectar al servidor de particiones de base de datos de la instancia o a todas las particiones de base de datos de la instancia.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint
- sysmon

Conexión necesaria

Instancia. Si no existe ninguna conexión de instancia, se creará una conexión de instancia por omisión.

Para obtener información de una instancia remota (o de una instancia local diferente), es necesario conectar primero esa instancia. Si no existe una conexión, se establece una conexión de instancia implícita con el nodo especificado por la variable de entorno DB2INSTANCE.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2GetSnapshotSize (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2GetSnapshotSizeData
{
   void *piSqlmaData;
   sqluint32 *poBufferSize;
   db2Uint32 iVersion;
   db2int32 iNodeNumber;
   db2Uint32 iSnapshotClass;
} db2GetSnapshotSizeData;

SQL_API_RC SQL_API_FN
  db2gGetSnapshotSize (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gGetSnapshotSizeData
{
   void *piSqlmaData;
   sqluint32 *poBufferSize;
   db2Uint32 iVersion;
   db2int32 iNodeNumber;
   db2Uint32 iSnapshotClass;
} db2gGetSnapshotSizeData;
```

Parámetros de la API db2GetSnapshotSize

versionNumber

Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct. Para utilizar la estructura tal como se describe más arriba, especifique db2Version810 o una versión más moderna. Si desea utilizar una versión diferente de esta estructura, vea la lista completa de versiones soportadas en el archivo de

cabecera db2ApiDf.h del directorio de inclusión. Debe utilizar la versión de la estructura db2GetSnapshotSizeStruct correspondiente al número de versión que especifique.

pParmStruct
Entrada. Puntero a la estructura db2GetSnapshotSizeStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2GetSnapshotSizeData

piSqlmaData
Entrada. Puntero a la estructura sqlma (monitor area) asignada por el usuario o a la estructura de datos de petición ″poRequestData″, creada y devuelta por la API db2AddSnapshotRequest. La estructura especifica el tipo o tipos de datos de instantánea que se deben recopilar. Si se utiliza un puntero a la estructura sqlma, la versión transferida en el parámetro versionNumber a la API db2GetSnapshotSize debe ser menor que db2Version900 (por ejemplo, db2Version810, db2Version822). Si se utiliza un puntero a la estructura de datos de petición devuelta por la API db2AddSnapshotRequest en el parámetro poRequestData, se debe pasar el valor db2Version900 en el parámetro versionNumber de la API db2GetSnapshotSize.

poBufferSize
Salida. Puntero al tamaño de almacenamiento intermedio calculado necesario para la API GET SNAPSHOT.

iVersion
Entrada. ID de versión de los datos del supervisor de bases de datos que se deben recopilar. El supervisor de bases de datos solamente devuelve datos que estaban disponibles para la versión solicitada. Establezca este parámetro en una de las constantes siguientes:

- SQLM_DBMON_VERSION1
- SQLM_DBMON_VERSION2
- SQLM_DBMON_VERSION5
- SQLM_DBMON_VERSION5_2
- SQLM_DBMON_VERSION6
- SQLM_DBMON_VERSION7
- SQLM_DBMON_VERSION8
- SQLM_DBMON_VERSION9
- SQLM_DBMON_VERSION9_5

Nota: Las constantes SQLM_DBMON_VERSION5_2 y anteriores han quedado obsoletas y es posible que se eliminen en un futuro release de DB2.

iNodeNumber
Entrada. El servidor de particiones de base de datos que debe enviarse la petición. De acuerdo con este valor, la petición se procesará para el servidor de particiones de base de datos actual, para todos los servidores de particiones de base de datos o para un servidor de particiones de base de datos especificado por el usuario. Los valores válidos son:

- SQLM_CURRENT_NODE

- SQLM_ALL_NODES. Este valor solamente está permitido si el valor de iVersion es SQLM_DBMON_VERSION7 o una versión más moderna.
- valor de nodo

Para las instancias autónomas, se debe utilizar el valor SQLM_CURRENT_NODE.

iSnapshotClass
Entrada. Calificador de clase de la instantánea. Los valores válidos (definidos en el archivo de cabecera sqlmon, situado en el directorio de inclusión) son:

- SQLM_CLASS_DEFAULT para una instantánea estándar
- SQLM_CLASS_HEALTH para una instantánea de estado
- SQLM_CLASS_HEALTH_WITH_DETAIL para una instantánea de estado que incluye detalles adicionales

Notas de uso

Esta función genera una cantidad significativa de actividad general. La asignación y liberación dinámica de memoria para cada llamada a la API db2GetSnapshot exige también recursos. Por ejemplo, si se invoca db2GetSnapshot repetidamente, por ejemplo, al muestrear datos durante un periodo de tiempo, puede ser preferible asignar un almacenamiento intermedio de tamaño fijo, en lugar de invocar db2GetSnapshotSize.

Si el supervisor del sistema de base de datos no encuentra ninguna base de datos ni aplicación activa, puede devolver un tamaño de almacenamiento intermedio igual a cero (por ejemplo, si se solicita información sobre bloqueos referente a una base de datos que no está activa). Compruebe que el tamaño calculado de almacenamiento intermedio devuelto por esta API sea distinto de cero antes de invocar db2GetSnapshot. Si db2GetSnapshot devuelve un error debido un espacio insuficiente de almacenamiento intermedio para contener los datos de salida, invoque de nuevo esta API para determinar las nuevas necesidades de tamaño.

db2GetSyncSession - Obtener un identificador de sesión de sincronización de satélites

Obtiene el identificador de sesión de sincronización actual del satélite.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2GetSyncSession (
   db2Uint32 versionNumber,
   void * pParmStruct,
```

```
   struct sqlca * pSqlca);

typedef struct db2GetSyncSessionStruct
{
   char *poSyncSessionID;
} db2GetSyncSessionStruct;
```

Parámetros de la API db2GetSyncSession

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2GetSyncSessionStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2GetSyncSessionStruct

poSyncSessionID
Salida. Especifica un identificador para la sesión de sincronización utilizada actualmente por un satélite.

db2HADRStart - Iniciar operaciones de HADR (high availability disaster recovery)

Inicia operaciones de HADR para una base de datos.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint

Conexión necesaria

Instancia. La API establece una conexión de base de datos si todavía no existe una y cierra la conexión de base de datos cuando la API finaliza.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2HADRStart (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HADRStartStruct
{
   char *piDbAlias;
   char *piUserName;
   char *piPassword;
   db2Uint32 iDbRole;
   db2Uint16 iByForce;
```

```
} db2HADRStartStruct;

SQL_API_RC SQL_API_FN
  db2gHADRStart (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gHADRStartStruct
{
   char *piDbAlias;
   db2Uint32 iAliasLen;
   char *piUserName;
   db2Uint32 iUserNameLen;
   char *piPassword;
   db2Uint32 iPasswordLen;
   db2Uint32 iDbRole;
   db2Uint16 iByForce;
} db2gHADRStartStruct;
```

Parámetros de la API db2HADRStart

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2HADRStartStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2HADRStartStruct

piDbAlias
Entrada. Puntero al alias de la base de datos.

piUserName
Entrada. Puntero al nombre de usuario con el que se ejecutará el mandato.

piPassword
Entrada. Puntero a una serie que contiene la contraseña.

iDbRole
Entrada. Especifica qué rol de base de datos HADR se debe iniciar en la base de datos especificada. Los valores válidos son:

DB2HADR_DB_ROLE_PRIMARY
Iniciar operaciones HADR para la base de datos en el rol primario.

DB2HADR_DB_ROLE_STANDBY
Iniciar operaciones HADR para la base de datos en el rol de espera.

iByForce
Entrada. Este argumento no se tiene en cuenta si el valor del parámetro iDbRole es DB2HADR_DB_ROLE_STANDBY. Los valores válidos son:

DB2HADR_NO_FORCE
Especifica que HADR se inicia en la base de datos primaria solamente si una base de datos de espera se conecta a ella dentro del límite de tiempo prescrito.

DB2HADR_FORCE
Especifica que se debe forzar el inicio de HADR, sin esperar a que la base de datos de espera se conecte a la base de datos primaria.

Parámetros específicos de la estructura de datos db2gHADRStartStruct

iAliasLen
Entrada. Especifica la longitud del alias de base de datos, en bytes.

iUserNameLen
Entrada. Especifica la longitud del nombre de usuario, en bytes.

iPasswordLen
Entrada. Especifica la longitud, en bytes, de la contraseña.

db2HADRStop - Detener operaciones de HADR (high availability disaster recovery)

Detiene operaciones de HADR para una base de datos.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint

Conexión necesaria

Instancia. La API establece una conexión de base de datos si todavía no existe una y cierra la conexión de base de datos cuando la API finaliza.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2HADRStop (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HADRStopStruct
{
   char *piDbAlias;
   char *piUserName;
   char *piPassword;
} db2HADRStopStruct;

SQL_API_RC SQL_API_FN
  db2gHADRStop (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gHADRStopStruct
{
   char *piDbAlias;
```

```
   db2Uint32 iAliasLen;
   char *piUserName;
   db2Uint32 iUserNameLen;
   char *piPassword;
   db2Uint32 iPasswordLen;
} db2gHADRStopStruct;
```

Parámetros de la API db2HADRStop

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2HADRStopStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2HADRStopStruct

piDbAlias
Entrada. Puntero al alias de la base de datos.

piUserName
Entrada. Puntero al nombre de usuario con el que se ejecutará el mandato.

piPassword
Entrada. Puntero a una serie que contiene la contraseña.

Parámetros específicos de la estructura de datos db2gHADRStopStruct

iAliasLen
Entrada. Especifica la longitud del alias de base de datos, en bytes.

iUserNameLen
Entrada. Especifica la longitud del nombre de usuario, en bytes.

iPasswordLen
Entrada. Especifica la longitud, en bytes, de la contraseña.

db2HADRTakeover - Dar instrucciones a una base de datos para que se convierta en la base de datos primaria de HADR (high availability disaster recovery)

Da instrucciones a una base de datos de espera para que se convierta en la base de datos primaria. Esta API solamente se puede invocar para una base de datos de espera.

Autorización

Una de las siguientes:

- *sysadm*
- *sysctrl*
- *sysmaint*

Conexión necesaria

Instancia. La API establece una conexión de base de datos si todavía no existe una y cierra la conexión de base de datos cuando la API finaliza.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2HADRTakeover (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HADRTakeoverStruct
{
   char *piDbAlias;
   char *piUserName;
   char *piPassword;
   db2Uint16 iByForce;
} db2HADRTakeoverStruct;

SQL_API_RC SQL_API_FN
  db2gHADRTakeover (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gHADRTakeoverStruct
{
   char *piDbAlias;
   db2Uint32 iAliasLen;
   char *piUserName;
   db2Uint32 iUserNameLen;
   char *piPassword;
   db2Uint32 iPasswordLen;
   db2Uint16 iByForce;
} db2gHADRTakeoverStruct;
```

Parámetros de la API db2HADRTakeover

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2HADRTakeoverStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2HADRTakeoverStruct

piDbAlias
Entrada. Puntero al alias de la base de datos.

piUserName
Entrada. Puntero al nombre de usuario con el que se ejecutará el mandato.

piPassword
Entrada. Puntero a una serie que contiene la contraseña.

iByForce
Entrada. Los valores válidos son:

DB2HADR_NO_FORCE
Especifica que tiene lugar un intercambio de roles solamente si existe una conexión establecida entre los dos sistemas y éstos están en un estado similar; esto da como resultado una inversión de roles entre las bases de datos primaria y de espera de HADR.

DB2HADR_FORCE
Especifica que la base de datos en espera toma el control como base de datos primaria sin esperar la confirmación de que se ha concluido la base de datos primaria original. La toma de control forzada se debe emitir cuando la base de datos está en los estados actualización remota pendiente o similar.

DB2HADR_FORCE_PEERWINDOW
Cuando se especifica esta opción, no se producirá una pérdida de transacciones confirmadas si el mandato es satisfactorio y se desactiva la base de datos primaria antes del final del periodo de ventana similar (establecido en el parámetro de configuración de base de datos **HADR_PEER_WINDOW** en un valor no cero). Si no se desactiva la base de datos primaria antes de que caduque la ventana similar, dará lugar a la *división de conocimientos*. Si se ejecutan cuando el par HADR no está en estado similar o similar desconectado (la ventana similar ha caducado), se devuelve un error.

Nota: La operación de toma de control con el parámetro DB2HADR_FORCE_PEERWINDOW puede comportarse incorrectamente si el reloj de la base de datos primaria y el reloj de la base de datos en espera no están sincronizados en 5 segundos entre sí. Es decir, la operación puede ser correcta cuando debería fallar, o fallar cuando debería ser correcta. Debe utilizar un servicio de sincronización de tiempo (por ejemplo, NTP) para que los relojes estén sincronizados con la misma fuente.

```
/* Valores para iByForce                                    */
#define DB2HADR_NO_FORCE          0    /* No realiza las operaciones*/
                                       /* START ni TAKEOVER HADR    */
                                       /* forzadas                  */
#define DB2HADR_FORCE             1    /* Realiza las operaciones   */
                                       /* START ni TAKEOVER HADR    */
                                       /* forzadas                  */
#define DB2HADR_FORCE_PEERWINDOW  2    /* Realiza la operación      */
                                       /* TAKEOVER HADR forzada sólo*/
                                       /* en la ventana similar     */
```

Parámetros específicos de la estructura de datos db2gHADRTakeoverStruct

iAliasLen
Entrada. Especifica la longitud del alias de base de datos, en bytes.

iUserNameLen
Entrada. Especifica la longitud del nombre de usuario, en bytes.

iPasswordLen
Entrada. Especifica la longitud, en bytes, de la contraseña.

db2HistoryCloseScan - Finalizar la exploración del archivo histórico

Finaliza una exploración del archivo histórico y libera recursos de DB2 necesarios para la exploración. Antes de ejecutar esta API es necesario invocar satisfactoriamente la API db2HistoryOpenScan.

Autorización

Ninguna

Conexión necesaria

Instancia. No es necesario invocar la API sqleatin antes de invocar esta API.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2HistoryCloseScan (
       db2Uint32 versionNumber,
       void * piHandle,
       struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  db2gHistoryCloseScan (
       db2Uint32 versionNumber,
       void * piHandle,
       struct sqlca * pSqlca);
```

Parámetros de la API db2HistoryCloseScan

versionNumber
Entrada. Especifica la versión y nivel de release del segundo parámetro, piHandle.

piHandle
Entrada. Especifica un puntero al descriptor de contexto de exploración que fue devuelto por la API db2HistoryOpenScan.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Para obtener una descripción detallada del uso de las API para archivos históricos, consulte la API db2HistoryOpenScan.

Sintaxis de la API de REXX

```
CLOSE RECOVERY HISTORY FILE :scanid
```

Parámetros de la API de REXX

scanid Variable de lenguaje principal que contiene el identificador de exploración devuelto por OPEN RECOVERY HISTORY FILE SCAN.

db2HistoryGetEntry - Obtener la entrada siguiente del archivo histórico

Obtiene la entrada siguiente del archivo histórico. Antes de ejecutar esta API es necesario invocar satisfactoriamente la API db2HistoryOpenScan.

Autorización

Ninguna

Conexión necesaria

Instancia. No es necesario invocar sqleatin antes de invocar esta API.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2HistoryGetEntry (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HistoryGetEntryStruct
{
   struct db2HistoryData *pioHistData;
   db2Uint16 iHandle;
   db2Uint16 iCallerAction;
} db2HistoryGetEntryStruct;

SQL_API_RC SQL_API_FN
  db2gHistoryGetEntry (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);
```

Parámetros de la API db2HistoryGetEntry

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2HistoryGetEntryStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2HistoryGetEntryStruct

pioHistData
Entrada. Puntero a la estructura db2HistData.

iHandle
Entrada. Contiene el descriptor de contexto de acceso de exploración que fue devuelto por la API db2HistoryOpenScan.

iCallerAction
Entrada. Especifica el tipo de acción que se debe realizar. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2HISTORY_GET_ENTRY
Obtiene la entrada siguiente, pero sin ningún dato de mandato.

DB2HISTORY_GET_DDL
Obtiene solamente los datos del mandato de la recuperación de datos anterior.

DB2HISTORY_GET_ALL
Obtiene la entrada siguiente, incluidos todos los datos.

Notas de uso

Los registros obtenidos se han seleccionado utilizando los valores especificados en la llamada a la API db2HistoryOpenScan.

Para obtener una descripción detallada del uso de las API para archivos históricos, consulte la API db2HistoryOpenScan.

Sintaxis de la API de REXX

```
GET RECOVERY HISTORY FILE ENTRY :scanid [USING :value]
```

Parámetros de la API de REXX

scanid Variable de lenguaje principal que contiene el identificador de exploración devuelto por OPEN RECOVERY HISTORY FILE SCAN.

value Variable compuesta de lenguaje principal de REXX en la que se devuelve la información sobre entradas del archivo histórico. En el texto que sigue a continuación, XXX representa el nombre de la variable de lenguaje principal:

XXX.0 Número de elementos de primer nivel de la variable (siempre 15)

XXX.1 Número de elementos de espacio de tablas

XXX.2 Número de elementos de espacio de tablas utilizados

XXX.3 OPERATION (tipo de operación ejecutada)

XXX.4 OBJECT (granularidad de la operación)

XXX.5 OBJECT_PART (indicación de fecha y hora y número de secuencia)

XXX.6 OPTYPE (calificador de la operación)

XXX.7 DEVICE_TYPE (tipo de dispositivo utilizado)

XXX.8 FIRST_LOG (ID del archivo de anotaciones más antiguo)

XXX.9 LAST_LOG (ID del archivo de anotaciones actual)

XXX.10
BACKUP_ID (identificador de la copia de seguridad)

XXX.11
SCHEMA (calificador del nombre de tabla)

XXX.12
TABLE_NAME (nombre de la tabla cargada)

XXX.13.0
NUM_OF_TABLESPACES (número de espacios de tablas que intervienen en la copia de seguridad o restauración)

XXX.13.1
Nombre del primer espacio de tablas copiado o restaurado

XXX.13.2
Nombre del segundo espacio de tablas copiado o restaurado

XXX.13.3
y así sucesivamente

XXX.14
LOCATION (ubicación donde se almacena la copia de seguridad)

XXX.15
COMMENT (texto para describir la entrada).

---

db2HistoryOpenScan - Iniciar una exploración del archivo histórico

Esta API inicia una exploración del archivo histórico.

Autorización

Ninguna

Conexión necesaria

Instancia. Si la base de datos está catalogada como remota, invoque la API sqleatin antes de invocar esta API.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2HistoryOpenScan (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HistoryOpenStruct
{
   char *piDatabaseAlias;
   char *piTimestamp;
   char *piObjectName;
   db2Uint32 oNumRows;
   db2Uint32 oMaxTbspaces;
   db2Uint16 iCallerAction;
   db2Uint16 oHandle;
} db2HistoryOpenStruct;

SQL_API_RC SQL_API_FN
  db2gHistoryOpenScan (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gHistoryOpenStruct
{
   char *piDatabaseAlias;
```

```
   char *piTimestamp;
   char *piObjectName;
   db2Uint32 iAliasLen;
   db2Uint32 iTimestampLen;
   db2Uint32 iObjectNameLen;
   db2Uint32 oNumRows;
   db2Uint32 oMaxTbspaces;
   db2Uint16 iCallerAction;
   db2Uint16 oHandle;
} db2gHistoryOpenStruct;
```

Parámetros de la API db2HistoryOpenScan

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada o salida. Puntero a la estructura de datos db2HistoryOpenStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2HistoryOpenStruct

piDatabaseAlias
Entrada. Puntero a una serie que contiene el alias de base de datos.

piTimestamp
Entrada. Puntero a una serie que especifica la indicación de fecha y hora que se debe utilizar para seleccionar registros. Se seleccionan los registros cuya indicación de fecha y hora es igual o posterior a este valor. Si el valor de este parámetro es NULL o apunta a cero, se impide el filtrado de entradas realizado mediante una indicación de fecha y hora.

piObjectName
Entrada. Puntero a una serie que especifica el nombre de objeto que se debe utilizar para seleccionar registros. El objeto puede ser una tabla o un espacio de tablas. Si se trata de una tabla, debe facilitarse el nombre de tabla totalmente calificado. Si el valor de este parámetro es NULL o apunta a cero, se impide el filtrado de entradas realizado mediante el nombre de objeto.

oNumRows
Salida. Después de finalizar la llamada a la API, este parámetro contiene el número de entradas coincidentes del archivo histórico.

oMaxTbspaces
Salida. Número máximo de nombres de espacios de tablas que se almacenan con cualquier entrada del archivo histórico.

iCallerAction
Entrada. Especifica el tipo de acción que se debe realizar. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2HISTORY_LIST_HISTORY
Lista todos los sucesos que están actualmente anotados cronológicamente en el archivo histórico.

DB2HISTORY_LIST_BACKUP
Lista las operaciones de copia de seguridad y restauración.

DB2HISTORY_LIST_ROLLFORWARD
Lista las operaciones de avance.

DB2HISTORY_LIST_DROPPED_TABLE
Lista los registros descartados. El campo DDL asociado con una entrada no se devuelve. Para recuperar la información de DDL para una entrada, se debe invocar db2HistoryGetEntry con la acción DB2HISTORY_GET_DDL inmediatamente después de obtener la entrada.

DB2HISTORY_LIST_LOAD
Lista las operaciones de carga.

DB2HISTORY_LIST_CRT_TABLESPACE
Lista las operaciones de crear y descartar espacios de tablas.

DB2HISTORY_LIST_REN_TABLESPACE
Lista las operaciones de redenominación de espacios de tablas.

DB2HISTORY_LIST_ALT_TABLESPACE
Lista las operaciones de modificar espacio de tablas. El campo DDL asociado con una entrada no se devuelve. Para recuperar la información de DDL para una entrada, se debe invocar db2HistoryGetEntry con la acción DB2HISTORY_GET_DDL inmediatamente después de obtener la entrada.

DB2HISTORY_LIST_REORG
Lista las operaciones de reorganización de tabla. Este valor no se puede utilizar actualmente.

oHandle
Salida. Después de finalizar la llamada a la API, este parámetro contiene el descriptor de contexto para el acceso de exploración. Posteriormente, se utiliza en las API db2HistoryGetEntry y db2HistoryCloseScan.

Parámetros específicos de la estructura de datos db2gHistoryOpenStruct

iAliasLen
Entrada. Especifica la longitud, en bytes, del alias de base de datos.

iTimestampLen
Entrada. Especifica la longitud de la serie de indicación de fecha y hora, en bytes.

iObjectNameLen
Entrada. Especifica la longitud de la serie del nombre de objeto, en bytes.

Notas de uso

Se pueden utilizar conjuntamente la indicación de fecha y hora, el nombre de objeto y la acción del llamador para filtrar registros. Solamente se devuelven los registros que pasan todos los filtros especificados.

El efecto de filtrado del nombre de objeto depende del valor especificado:

- Si se especifica una tabla, se devuelven registros correspondientes a operaciones de carga, pues esta es la única información para tablas contenida en el archivo histórico.
- Si se especifica un espacio de tablas, se devuelven registros correspondientes a operaciones de copia de seguridad, restauración y carga del espacio de tablas.

Nota: Para devolver registros para tablas, las tablas se deben especificar en la forma esquema.nombretabla. La especificación del nombre de tabla solamente devuelve registros para espacios de tablas.

Está permitido un máximo de ocho exploraciones de archivo histórico por cada proceso.

Para listar todas las entradas del archivo histórico, una aplicación típica realizará estos pasos:

1. Llamar a la API db2HistoryOpenScan, que devuelve el valor de parámetro oNumRows.
2. Asignar una estructura db2HistData con espacio para n campos oTablespace, donde n es un número arbitrario.
3. Establecer el campo iNumTablespaces de la estructura db2HistoryData en n.
4. Efectuar lo siguiente en un bucle:
   - Llamar a la API db2HistoryGetEntry para obtener entradas del archivo histórico.
   - Si la API db2HistoryGetEntry devuelve un valor de SQLCODE SQL_RC_OK, utilice el campo oNumTablespaces de la estructura db2HistoryData para determinar el número de entradas de espacio de tablas devueltas.
   - Si la API db2HistoryGetEntry devuelve un valor SQLUH_SQLUHINFO_VARS_WARNING de SQLCODE, significa que no se ha asignado espacio suficiente para todos los espacios de tablas que DB2 está intentando devolver; libere y reasigne la estructura db2HistoryData con espacio suficiente para entradas de espacio de tablas oDB2UsedTablespace y establezca iDB2NumTablespace en oDB2UsedTablespace.
   - Si la API db2HistoryGetEntry devuelve SQLE_RC_NOMORE como valor de SQLCODE, significa que se han recuperado todas las entradas del archivo histórico.
   - Cualquier otro valor de SQLCODE indica la existencia de un problema.
5. Cuando se haya obtenido toda la información, llame a la API db2HistoryCloseScan para liberar los recursos asignados por la llamada a db2HistoryOpenScan.

Se proporciona la macro SQLUHINFOSIZE(n) (definida en el archivo de cabecera sqlutil) para ayudar a determinar cuánta memoria es necesaria para una estructura db2HistoryData con espacio para n entradas oTablespace.

Sintaxis de la API de REXX

```
OPEN [BACKUP] RECOVERY HISTORY FILE FOR database_alias
[OBJECT objname] [TIMESTAMP :timestamp]
USING :value
```

Parámetros de la API de REXX

database_alias
Alias de la base de datos para la que se debe listar su archivo histórico.

objname
Especifica el nombre del objeto que se va a utilizar para seleccionar registros. El objeto puede ser una tabla o un espacio de tablas. Si se trata de una tabla, debe facilitarse el nombre de tabla totalmente calificado. Establecer este parámetro en NULL impide el filtrado de entradas utilizando el nombre de objeto.

timestamp
Especifica la indicación de fecha y hora que se va a utilizar para seleccionar registros. Se seleccionan los registros cuya indicación de fecha y hora es igual o posterior a este valor. Establecer este parámetro en NULL impide el filtrado de entradas utilizando la indicación de fecha y hora.

value Variable compuesta de lenguaje principal de REXX en la que se devuelve información sobre el archivo histórico. En las expresiones que siguen, XXX representa el nombre de la variable de lenguaje principal.

XXX.0 Número de elementos de la variable (siempre 2)

XXX.1 Identificador (descriptor de contexto) para futuros accesos de exploración

XXX.2 Número de entradas coincidentes del archivo histórico.

db2HistoryUpdate - Actualizar una entrada de archivo histórico

Actualiza la ubicación, el tipo de dispositivo o el comentario en una entrada de archivo histórico.

Autorización

Una de las siguientes:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Conexión necesaria

Base de datos. Para actualizar entradas del archivo histórico para una base de datos que no sea la base de datos por omisión, se debe establecer una conexión con la base de datos antes de invocar esta API.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2HistoryUpdate (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HistoryUpdateStruct
{
   char *piNewLocation;
   char *piNewDeviceType;
   char *piNewComment;
   char *piNewStatus;
   db2HistoryEID iEID;
} db2HistoryUpdateStruct;

typedef SQL_STRUCTURE db2HistoryEID
{
   SQL_PDB_NODE_TYPE ioNode;
   db2Uint32 ioHID;
```

```
} db2HistoryEID;

SQL_API_RC SQL_API_FN
  db2gHistoryUpdate (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gHistoryUpdateStruct
{
   char *piNewLocation;
   char *piNewDeviceType;
   char *piNewComment;
   char *piNewStatus;
   db2Uint32 iNewLocationLen;
   db2Uint32 iNewDeviceLen;
   db2Uint32 iNewCommentLen;
   db2Uint32 iNewStatusLen;
   db2HistoryEID iEID;
} db2gHistoryUpdateStruct;
```

Parámetros de la API db2HistoryUpdate

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2HistoryUpdateStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2HistoryUpdateStruct

piNewLocation
Entrada. Puntero a una serie que especifica una nueva ubicación para la imagen de copia de seguridad, restauración o de copia de carga. Si el valor de este parámetro NULL, o apunta a cero, el valor permanece inalterado.

piNewDeviceType
Entrada. Puntero a una serie que especifica un nuevo tipo de dispositivo para almacenar la imagen de copia de seguridad, restauración o de copia de carga. Si el valor de este parámetro NULL, o apunta a cero, el valor permanece inalterado. Los tipos de dispositivo válidos son:

D Disco

K Disquete

T Cinta

F Copia de seguridad instantánea

A Tivoli Storage Manager

U Salida de usuario

P Área de interconexión de memoria

N Dispositivo nulo

X XBSA

Q Sentencia de SQL

O Otro

piNewComment
Entrada. Puntero a una serie que especifica un comentario nuevo para describir la entrada. Si el valor de este parámetro es NULL o apunta a cero, el comentario permanece inalterado.

piNewStatus
Entrada. Puntero a una serie que especifica un tipo de estado nuevo para la entrada. Si el valor de este parámetro es NULL o apunta a cero, el estado permanece inalterado. Los valores válidos son:

A Activa. La imagen de copia de seguridad está en la cadena de anotación cronológica activa. La mayoría de las entradas están activas.

I Inactiva. Las imágenes de copia de seguridad que ya no se corresponden con la secuencia de anotaciones cronológicas actuales, que también se denomina cadena de anotaciones cronológicas actuales, están marcadas como inactivas.

E Caducada. Las imágenes de copia de seguridad que ya no se necesitan porque hay más de NUM_DB_BACKUPS imágenes activas se marcan como caducadas.

D Suprimida. Las imágenes de copia de seguridad que ya no están disponibles para su recuperación deberían marcarse como ya suprimidas.

X Do_not_delete. Las entradas del histórico de recuperación que se marcan para no suprimir, no se recortarán ni suprimirán por las llamadas al mandato PRUNE HISTORY, ejecutando el procedimiento ADMIN_CMD con PRUNE HISTORY, las llamadas a la API db2Prune ni el recorte del archivo histórico de recuperación automatizada. Puede utilizar el estado do_not_delete para proteger las entradas del archivo de recuperación de claves contra el recorte y los objetos de recuperación asociados a ellas contra la supresión.

iEID Entrada. Identificador exclusivo que se puede utilizar para actualizar una entrada específica del archivo histórico.

Parámetros de la estructura de datos db2HistoryEID

ioNode
Este parámetro se puede utilizar como parámetro de entrada o salida.

Indica el número de nodo.

ioHID Este parámetro se puede utilizar como parámetro de entrada o salida.

Indica el ID de entrada del archivo histórico local.

Parámetros específicos de la estructura de datos db2gHistoryUpdateStruct

iNewLocationLen
Entrada. Especifica la longitud, en bytes, del parámetro piNewLocation.

iNewDeviceLen
Entrada. Especifica la longitud, en bytes, del parámetro piNewDeviceType.

iNewCommentLen
Entrada. Especifica la longitud, en bytes, del parámetro piNewComment.

iNewStatusLen
Entrada. Especifica la longitud, en bytes, del parámetro piNewStatus.

Notas de uso

Esta es una función de actualización, y toda la información anterior al cambio es sustituida y no se puede volver a crear. Estos cambios no se registran en el archivo de anotaciones.

El propósito principal del archivo histórico de la base de datos es registrar información, pero los datos contenidos en el histórico son utilizados directamente por operaciones de restauración automática. Durante cualquier restauración donde se especifique la opción AUTOMATIC, el programa de restauración utilizará y hará referencia al histórico de imágenes de copia de seguridad y a sus ubicaciones para realizar la petición de restauración automática. Si se va a utilizar la función de restauración automática y se ha cambiado la ubicación de las imágenes de copia de seguridad desde que se crearon, se recomienda que el registro del histórico de la base de datos para estas imágenes se actualice para reflejar la ubicación actual. Si la ubicación de las imágenes de copia de seguridad en el histórico de la base de datos no está actualizada, la restauración automática no podrá ubicar las imágenes de copia de seguridad, pero todavía se pueden utilizar satisfactoriamente los mandatos de restauración manual.

Sintaxis de la API de REXX

```
UPDATE RECOVERY HISTORY USING :value
```

Parámetros de la API de REXX

value Variable compuesta de lenguaje principal de REXX que contiene información referente a la nueva ubicación de una entrada del archivo histórico. En el texto que sigue a continuación, XXX representa el nombre de la variable de lenguaje principal:

XXX.0 Número de elementos de la variable (debe estar entre 1 y 4)

XXX.1 OBJECT_PART (indicación de fecha y hora con un número de secuencia entre 001 y 999)

XXX.2 Nueva ubicación de la imagen de copia de seguridad o de copia (este parámetro es opcional)

XXX.3 Nuevo dispositivo utilizado para almacenar la imagen de copia de seguridad o de copia (este parámetro es opcional)

XXX.4 Nuevo comentario (este parámetro es opcional).

db2Import - Importar datos a una tabla, jerarquía, apodo o vista

Inserta datos de un archivo externo con un formato de archivo soportado en una tabla, jerarquía, apodo o vista. El programa de utilidad de carga es más rápido que esta función. No obstante, el programa de utilidad de carga no soporta la carga de datos a nivel de jerarquía ni la carga en un apodo.

Autorización

- IMPORT utilizando la opción **INSERT** requiere una de las autorizaciones siguientes:
  - sysadm

  - dbadm
  - Privilegio CONTROL en cada tabla, vista o apodo participante
  - Privilegio INSERT y SELECT en cada tabla o vista participante
- IMPORT en una tabla existente utilizando la opción **INSERT_UPDATE**, necesita una de las autorizaciones siguientes:
  - sysadm
  - dbadm
  - Privilegio CONTROL en la tabla, vista o apodo
  - Privilegio INSERT, SELECT, UPDATE y DELETE en cada tabla o vista participante
- IMPORT en una tabla existente utilizando la opción **REPLACE** o **REPLACE_CREATE**, requiere una de las autorizaciones siguientes:
  - sysadm
  - dbadm
  - Privilegio CONTROL sobre la tabla o la vista
  - Privilegio INSERT, SELECT y DELETE sobre la tabla o vista
- IMPORT en una tabla nueva utilizando la opción **CREATE** o **REPLACE_CREATE**, requiere una de las autorizaciones siguientes:
  - sysadm
  - dbadm
  - Autorización CREATETAB para la base de datos y privilegio USE para el espacio de tablas, y también uno de los elementos siguientes:
    - Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito de la tabla no existe
    - Privilegio CREATEIN sobre el esquema, si el nombre esquema de la tabla hace referencia a un esquema existente.
- IMPORT a una tabla o jerarquía que no existe utilizando **CREATE**, o la opción **REPLACE_CREATE**, necesita una de las autorizaciones siguientes:
  - sysadm
  - dbadm
  - Autorización CREATETAB para la base de datos y uno de los elementos siguientes:
    - Autorización IMPLICIT_SCHEMA sobre la base de datos, si no existe el nombre de esquema de la tabla
    - Privilegio CREATEIN sobre el esquema, si existe el esquema de la tabla
    - Privilegio CONTROL en cada subtabla de la jerarquía, si se utiliza la opción **REPLACE_CREATE** en la jerarquía entera
- IMPORT en una jerarquía existente utilizando la opción **REPLACE** requiere una de las autorizaciones siguientes:
  - sysadm
  - dbadm
  - Privilegio CONTROL en cada subtabla de la jerarquía

Conexión necesaria

Base de datos. Si se ha habilitado la conexión implícita, se establece una conexión con la base de datos por omisión.

Archivo de inclusión de la API

db2ApiDf.h

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2Import (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ImportStruct
{
   char *piDataFileName;
   struct sqlu_media_list *piLobPathList;
   struct sqldcol *piDataDescriptor;
   struct sqlchar *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piMsgFileName;
   db2int16 iCallerAction;
   struct db2ImportIn *piImportInfoIn;
   struct db2ImportOut *poImportInfoOut;
   db2int32 *piNullIndicators;
   struct sqllob *piLongActionString;
} db2ImportStruct;

typedef SQL_STRUCTURE db2ImportIn
{
   db2Uint64 iRowcount;
   db2Uint64 iRestartcount;
   db2Uint64 iSkipcount;
   db2int32 *piCommitcount;
   db2Uint32 iWarningcount;
   db2Uint16 iNoTimeout;
   db2Uint16 iAccessLevel;
   db2Uint16 *piXmlParse;
   struct db2DMUXmlValidate *piXmlValidate;
} db2ImportIn;

typedef SQL_STRUCTURE db2ImportOut
{
   db2Uint64 oRowsRead;
   db2Uint64 oRowsSkipped;
   db2Uint64 oRowsInserted;
   db2Uint64 oRowsUpdated;
   db2Uint64 oRowsRejected;
   db2Uint64 oRowsCommitted;
} db2ImportOut;

typedef SQL_STRUCTURE db2DMUXmlMapSchema
{
   struct db2Char                      iMapFromSchema;
   struct db2Char                      iMapToSchema;
} db2DMUXmlMapSchema;

typedef SQL_STRUCTURE db2DMUXmlValidateXds
{
   struct db2Char *piDefaultSchema;
   db2Uint32 iNumIgnoreSchemas;
   struct db2Char *piIgnoreSchemas;
   db2Uint32 iNumMapSchemas;
   struct db2DMUXmlMapSchema *piMapSchemas;
} db2DMUXmlValidateXds;

typedef SQL_STRUCTURE db2DMUXmlValidateSchema
{
```

```
   struct db2Char *piSchema;
} db2DMUXmlValidateSchema;

typedef SQL_STRUCTURE db2DMUXmlValidate
{
   db2Uint16 iUsing;
   struct db2DMUXmlValidateXds *piXdsArgs;
   struct db2DMUXmlValidateSchema *piSchemaArgs;
} db2DMUXmlValidate;

SQL_API_RC SQL_API_FN
  db2gImport (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gImportStruct
{
   char *piDataFileName;
   struct sqlu_media_list *piLobPathList;
   struct sqldcol *piDataDescriptor;
   struct sqlchar *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piMsgFileName;
   db2int16 iCallerAction;
   struct db2gImportIn *piImportInfoIn;
   struct dbg2ImportOut *poImportInfoOut;
   db2int32 *piNullIndicators;
   db2Uint16 iDataFileNameLen;
   db2Uint16 iFileTypeLen;
   db2Uint16 iMsgFileNameLen;
   struct sqllob *piLongActionString;
} db2gImportStruct;

typedef SQL_STRUCTURE db2gImportIn
{
   db2Uint64 iRowcount;
   db2Uint64 iRestartcount;
   db2Uint64 iSkipcount;
   db2int32 *piCommitcount;
   db2Uint32 iWarningcount;
   db2Uint16 iNoTimeout;
   db2Uint16 iAccessLevel;
   db2Uint16 *piXmlParse;
   struct db2DMUXmlValidate *piXmlValidate;
} db2gImportIn;

typedef SQL_STRUCTURE db2gImportOut
{
   db2Uint64 oRowsRead;
   db2Uint64 oRowsSkipped;
   db2Uint64 oRowsInserted;
   db2Uint64 oRowsUpdated;
   db2Uint64 oRowsRejected;
   db2Uint64 oRowsCommitted;
} db2gImportOut;
```

Parámetros de la API db2Import

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura que se ha pasado como segundo parámetro **pParmStruct**.

pParmStruct
Entrada/Salida. Un puntero para la estructura db2ImportStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2ImportStruct

piDataFileName
Entrada. Serie que contiene la vía de acceso y el nombre del archivo de entrada externo a partir del que deben importarse los datos.

piLobPathList
Entrada. Puntero para sqlu_media_list con su campo media_type establecido en SQLU_LOCAL_MEDIA, y su estructura sqlu_media_entry que lista las vías de acceso del cliente donde se pueden encontrar los archivos LOB. Este parámetro no es válido cuando se importa a un apodo.

piDataDescriptor
Entrada. Puntero para una estructura sqldcol que contiene información acerca de las columnas que se seleccionan para importación desde el archivo externo. El valor del campo dcolmeth determina cómo interpretará el programa de utilidad de importación el resto de la información proporcionada en este parámetro. Los valores válidos para este parámetro son:

SQL_METH_N
Nombres. La selección de columnas del archivo de entrada externo se hace por nombre de columna.

SQL_METH_P
Posiciones. La selección de columnas del archivo de entrada externo se hace por posición de columna.

SQL_METH_L
Ubicaciones. La selección de columnas del archivo de entrada externo se hace por ubicación de la columna. El gestor de bases de datos rechaza una llamada de importación con un par de valores de ubicación que no son válidos porque cualquiera de las condiciones siguientes:

- O bien la ubicación inicial o bien la ubicación final no están comprendidas entre 1 y el entero con signo de dos bytes más grande.
- La ubicación final es más pequeña que la ubicación inicial.
- El ancho de la columna de entrada definido por el par de valores de ubicación no es compatible con el tipo y la longitud de la columna de destino.

Un par de valores de ubicación con ambas ubicaciones igual a cero indica que una columna que puede tener valores nulos se va a llenar con valores NULL.

SQL_METH_D
Valor por omisión. Si **piDataDescriptor** es NULL, o se establece en SQL_METH_D, se realiza la selección por omisión de columnas desde el archivo de entrada externo. En este caso, se hará caso omiso tanto del número de columnas como de la matriz de especificación de columnas. Para los archivos DEL, IXF o WSF, las primeras n columnas de datos del archivo de entrada externo se toman en su orden natural, donde n es el número de columnas de bases de datos en las que se van a importar los datos.

piActionString
En desuso. Sustituido por **piLongActionString**.

piLongActionString
Entrada. Puntero a una estructura sqllob que contiene un campo de 4 bytes de longitud, seguido por una matriz de caracteres que especifica una acción que afecta a la tabla.

La matriz de caracteres tiene este formato:

```
{INSERT | INSERT_UPDATE | REPLACE | CREATE | REPLACE_CREATE}
INTO {tname[(tcolumn-list)] |
[{ALL TABLES | (tname[(tcolumn-list)][, tname[(tcolumn-list)]])}]
[IN] HIERARCHY {STARTING tname | (tname[, tname])}
[UNDER sub-table-name | AS ROOT TABLE]}
```

INSERT
Añade los datos importados a la tabla sin cambiar los datos de tabla existentes.

INSERT_UPDATE
Añade las filas importadas si los valores de clave primaria no están en la tabla y las utiliza para realizar una actualización si se encuentran los valores de clave primaria. Esta opción sólo es válida si la tabla de destino tiene una clave primaria y la lista especificada (o implícita) de columnas de destino que se van a importar incluye todas las columnas de la clave primaria. Esta opción no es aplicable a las vistas.

REPLACE
Suprime todos los datos existentes de la tabla truncando el objeto de tabla e inserta los datos importados. La definición de tabla y las definiciones de índice no se modifican. (Los índices se suprimen y se sustituyen si indexixf se encuentra en FileTypeMod y FileType es SQL_IXF.) Si la tabla aún no se ha definido, se devolverá un error.

Nota: Si se produce un error después de suprimir los datos existentes, esos datos se perderán.
Este parámetro no es válido cuando se importa a un apodo.

CREATE

Nota: El parámetro CREATE ha quedado obsoleto y se puede eliminar en un futuro release. Para obtener información detallada, consulte “Las opciones del mandato IMPORT, CREATE y REPLACE_CREATE, están en desuso”.

Crea la definición de la tabla y el contenido de las filas utilizando la información especificada en el archivo PC/IXF especificado, si la tabla especificada no está definida. Si DB2 exportó anteriormente el archivo, también se crearán los índices. Si la tabla especificada ya se ha definido, se devolverá un error. Esta opción sólo es válida para el formato de archivos PC/IXF. Este parámetro no es válido cuando se importa a un apodo.

REPLACE_CREATE

Nota: El parámetro REPLACE_CREATE se ha quedado obsoleto y se puede eliminar en un futuro release. Para obtener información

detallada, consulte “Las opciones del mandato IMPORT, CREATE y REPLACE_CREATE, están en desuso”.

Sustituye el contenido de la tabla utilizando la información de filas PC/IXF del archivo PC/IXF, si se ha definido la tabla especificada. Si aún no se ha definido la tabla, la definición de la tabla y el contenido de las filas se crearán utilizando la información del archivo PC/IXF especificado. Si DB2 exportó anteriormente el archivo PC/IXF, también se crearán los índices. Esta opción sólo es válida para el formato de archivos PC/IXF.

Nota: Si se produce un error después de suprimir los datos existentes, esos datos se perderán.
Este parámetro no es válido cuando se importa a un apodo.

tname Nombre de la tabla, tabla escrita, vista o vista del objeto en que se van a insertar los datos. Se puede especificar un alias para REPLACE, INSERT_UPDATE o INSERT, excepto en el caso de un servidor anterior, en que se debe especificar un nombre calificado o no calificado. Si es una vista, no puede ser una vista de sólo lectura.

tcolumn-list
Lista de nombres de columnas de tablas o vistas en que deben insertarse los datos. Los nombres de columnas deben estar separados por comas. Si no se especifican nombres de columnas, se utilizarán los nombres de columnas definidos en las sentencias CREATE TABLE o ALTER TABLE. Si no se ha especificado una lista de columnas para las tablas de tipo, los datos se insertarán en todas las columnas de cada subtabla.

sub-table-name
Especifica una tabla padre cuando se crean una o más subtablas en la opción CREATE.

ALL TABLES
Palabra clave implícita sólo para jerarquía. Cuando se importa una jerarquía, el valor por omisión es importar todas las tablas especificadas en la lista de orden transversal.

HIERARCHY
Especifica que deben importarse datos jerárquicos.

STARTING
Palabra clave sólo para jerarquía. Especifica que debe utilizarse el orden por omisión, empezando por el nombre de una determinada subtabla.

UNDER
Palabra clave sólo para jerarquía y CREATE. Especifica que la nueva jerarquía, subjerarquía o subtabla se creará bajo una determinada subtabla.

AS ROOT TABLE
Palabra clave sólo para jerarquía y CREATE. Especifica que la nueva jerarquía, subjerarquía o subtabla se creará como una jerarquía autónoma.

Los parámetros nombret y lista-columnast se corresponden con las listas nombres de tablas y de nombres de columnas de las sentencias INSERT de SQL y tienen las mismas restricciones.

Las columnas de lista-columnast y las columnas externas (tanto especificadas como implícitas) coinciden según con su posición en la lista o en la estructura (los datos de la primera columna especificada en la estructura sqldcol se insertan en el campo de tabla o de vista correspondiente al primer elemento de la lista-columnast).

Si se especifican números distintos de columnas, el número de columnas realmente procesadas es el menor de los dos números. Eso puede producir un error (puesto que no hay valores que poner en algunos campos de la tabla que no pueden contener valores nulos) o un mensaje informativo (porque algunas columnas de archivo externas se pasan por alto).

Este parámetro no es válido cuando se importa a un apodo.

piFileType
Entrada. Serie que indica el formato de los datos del archivo externo. Los formatos de archivos externos soportados son:

SQL_ASC
ASCII no delimitado.

SQL_DEL
ASCII delimitado, para intercambio con dBase, BASIC y los programas IBM Personal Decision Series, y muchos otros gestores de bases de datos y gestores de archivos.

SQL_IXF
Versión de PC del formato de intercambio integrado, el método favorito de exportar datos de una tabla para poder importarlos más tarde a la misma tabla o a la tabla de otro gestor de bases de datos.

SQL_WSF
Formatos de hojas de trabajo para realizar intercambios con los programas Lotus Symphony y 1-2-3. El tipo de archivo WSF no está soportado cuando se importa a un apodo.

piFileTypeMod
Entrada. Puntero a una estructura que contiene un campo de dos bytes de longitud seguido por una matriz de caracteres que especifica una o más opciones de proceso. Si este puntero es NULL, o la estructura a la que apunta tiene cero caracteres, esta acción se interpreta como selección de una especificación por omisión.

No todas las opciones pueden utilizarse con todos los tipos de archivos soportados. Consulte el enlace relacionado ″Modificadores de tipo de archivo para el programa de utilidad de importación″.

piMsgFileName
Entrada. Serie que contiene el destino de mensajes de error, mensajes de aviso y mensajes informativos devueltos por el programa de utilidad. Puede ser la vía de acceso y el nombre de un archivo del sistema operativo o un dispositivo estándar. Si el archivo ya existe, la información se añadirá. Si no existe, el archivo se creará.

iCallerAction
Entrada. Acción solicitada por el llamador. Los valores válidos son:

SQLU_INITIAL
Llamada inicial. Este valor debe utilizarse en la primera llamada a la API. Si la llamada inicial o las llamadas posteriores devuelven el control y exigen a la aplicación que realiza la llamada que lleve a

cabo alguna acción antes de completar la operación de importación solicitada, la acción del peticionario debe establecerse en una de las siguientes:

SQLU_CONTINUE
Continúa el proceso. Este valor sólo puede utilizarse en llamadas posteriores a la API, después de que la llamada inicial haya devuelto el control al programa de utilidad que solicitaba una entrada del usuario (por ejemplo, para responder a una condición de fin de cinta). Especifica que la acción del usuario solicitada por el programa de utilidad se ha completado y el programa de utilidad puede continuar procesando la petición inicial.

SQLU_TERMINATE
Termina el proceso. Este valor sólo puede utilizarse en llamadas posteriores a la API, después de que la llamada inicial haya devuelto el control al programa de utilidad que solicitaba una entrada del usuario (por ejemplo, para responder a una condición de fin de cinta). Especifica que la acción del usuario que había solicitado el programa de utilidad no se ha realizado y que éste ha de finalizar el proceso de la petición inicial.

piImportInfoIn
Entrada. Puntero para la estructura db2ImportIn.

poImportInfoOut
Salida. Puntero para la estructura db2ImportOut.

piNullIndicators
Entrada. Sólo para archivos ASC. Matriz de enteros que indica si los datos de la columna pueden contener valores nulos o no. El número de elementos de esta matriz debe coincidir con el número de columnas del archivo de entrada; hay una correspondencia unívoca entre los elementos de esta matriz y las columnas que se van a importar del archivo de datos. Por lo tanto, el número de elementos debe ser igual al campo dcolnum del parámetro **piDataDescriptor**. Cada elemento de la matriz contiene un número que identifica una columna del archivo de datos que se utilizará como campo de indicador nulo, o un cero indicando que la columna de tabla no puede contener valores nulos. Si el elemento no es cero, la columna identificada en el archivo de datos debe contener una Y o una N. Una Y indica que los datos de columna de tabla son NULL y N indica que los datos de columna de tabla no son NULL.

piXmlPathList
Entrada. Puntero para sqlu_media_list con su campo media_type establecido en SQLU_LOCAL_MEDIA, y su estructura sqlu_media_entry que lista las vías de acceso del cliente donde se pueden encontrar los archivos XML.

Parámetros de la estructura de datos db2ImportIn

iRowcount
Entrada. Número de registros físicos a cargar. Permite a un usuario cargar solamente las primeras **iRowcount** filas de un archivo. Si **iRowcount** es 0, la importación intentará procesar todas las filas del archivo.

iRestartcount
Entrada. Número de registros que hay que saltarse antes de insertar o

actualizar registros. Es funcionalmente equivalente al parámetro **iSkipcount**. Los parámetros **iRestartcount** e **iSkipcount** se excluyen mutuamente.

iSkipcount
Entrada. Número de registros que hay que saltarse antes de insertar o actualizar registros. Es funcionalmente equivalente a **iRestartcount**.

piCommitcount
Entrada. Número de registros que hay que importar antes de confirmarlos con la base de datos. Se realiza una confirmación siempre que se importan **piCommitcount** registros. El valor NULL especifica el valor del número de confirmación por omisión, que es cero para la importación fuera de línea y AUTOMATIC para la importación en línea. Commitcount AUTOMATIC se especifica pasando el valor DB2IMPORT_COMMIT_AUTO.

iWarningcount
Entrada. Detiene la operación de importación después de **iWarningcount** avisos. Establezca este parámetro si no se esperan avisos, pero se desea que se verifique si se están utilizando el archivo y la tabla correctos. Si el archivo de importación o la tabla de destino se especifican de modo incorrecto, el programa de utilidad de importación generará un aviso por cada fila que intente importar, lo que hará que la importación falle.

Si **iWarningcount** es 0, o no se especifica esta opción, la operación de importación continuará sin tener en cuenta el número de avisos emitidos.

iNoTimeout
Entrada. Especifica que el programa de utilidad de importación no superará el tiempo de espera mientras espere bloqueos. Esta opción sustituye al parámetro de configuración de base de datos **locktimeout**. No se verá afectada ninguna otra aplicación. Los valores válidos son:

DB2IMPORT_LOCKTIMEOUT
Indica que se debe respetar el valor del parámetro de configuración **locktimeout**.

DB2IMPORT_NO_LOCKTIMEOUT
Indica que no hay tiempo de espera excedido.

iAccessLevel
Entrada. Especifica el nivel de acceso. Los valores válidos son:

- SQLU_ALLOW_NO_ACCESS
Especifica que el programa de utilidad de importación bloquea la tabla de forma exclusiva.

- SQLU_ALLOW_WRITE_ACCESS
Especifica que los datos de la tabla deben ser accesibles a los lectores y grabadores mientras la operación de importación está en curso.

Cuando se inserta la primera fila, se adquiere un bloqueo de intento exclusivo (IX) sobre la tabla de destino. Esto permite que haya varios lectores y grabadores que accedan simultáneamente a los datos de la tabla. La modalidad en línea no es compatible con las opciones de importación **REPLACE**, **CREATE** o **REPLACE_CREATE**. La modalidad en línea no se puede utilizar conjuntamente con las inserciones en almacenamiento intermedio. La operación de importación confirmará periódicamente datos insertados para evitar la escala de bloqueo en un bloqueo de tabla y para evitar el agotamiento del espacio de anotación cronológica activa. Estas

confirmaciones se realizarán aunque no se haya utilizado el parámetro **piCommitCount**. Durante cada confirmación, la importación perderá su bloqueo de tabla IX e intentará readquirirlo tras la confirmación. Este parámetro es necesario cuando se importa a un apodo y se debe especificar el parámetro **piCommitCount** con un número válido (AUTOMATIC no se considera una opción válida).

piXmlParse
Entrada. Tipo de análisis que debe realizarse para los documentos XML. Los valores válidos que se encuentran en el archivo de cabecera db2ApiDf del directorio de inclusión, son:

DB2DMU_XMLPARSE_PRESERVE_WS
Los espacios en blanco deben conservarse.

DB2DMU_XMLPARSE_STRIP_WS
Los caracteres en blanco deben eliminarse.

piXmlValidate
Entrada. Puntero para la estructura db2DMUXmlValidate. Indica que debe llevarse a cabo la validación de esquemas XML para los documentos XML.

Parámetros de la estructura de datos db2ImportOut

oRowsRead
Salida. Número de registros leídos en el archivo durante la importación.

oRowsSkipped
Salida. Número de registros que se han saltado antes de que se empiece a insertar o actualizar.

oRowsInserted
Salida. Número de filas insertadas en la tabla de destino.

oRowsUpdated
Salida. Número de filas de la tabla de destino actualizadas con información de los registros importados (registros cuyo valor de clave primaria ya existe en la tabla).

oRowsRejected
Salida. Número de registros que no se han podido importar.

oRowsCommitted
Salida. Número de registros importados satisfactoriamente y confirmados en la base de datos.

Parámetros de la estructura de datos db2DMUXmlMapSchema

iMapFromSchema
Entrada. Identificador de SQL del esquema XML desde el que realizar la correlación.

iMapToSchema
Entrada. Identificador de SQL del esquema XML al que realizar la correlación.

Parámetros de la estructura de datos db2DMUXmlValidateXds

piDefaultSchema
Entrada. Identificador SQL del esquema XML que debe utilizarse para su validación cuando un XDS no contiene un atributo SCH.

iNumIgnoreSchemas
Entrada. Número de esquemas XML que no se tendrán en cuenta durante la validación de esquemas XML si un atributo SCH de XDS les hace referencia.

piIgnoreSchemas
Entrada. Lista de esquemas XML que no se tendrán en cuenta durante la validación de esquemas XML si un atributo SCH de XDS les hace referencia.

iNumMapSchemas
Entrada. Número de esquemas XML que se correlacionarán durante la validación de esquemas XML. El primer esquema del par de correlaciones de esquemas representa un esquema al que hace referencia un atributo SCH de un XDS. El segundo esquema del par representa el esquema que debe utilizarse para llevar a cabo la validación de esquemas.

piMapSchemas
Entrada. Lista de pares de esquemas XML, donde cada par representa una correlación de un esquema con otro distinto. El primer esquema del par representa un esquema al que hace referencia un atributo SCH de un XDS. El segundo esquema del par representa el esquema que debe utilizarse para llevar a cabo la validación de esquemas.

Parámetros de la estructura de datos db2DMUXmlValidateSchema

piSchema
Entrada. Identificador de SQL del esquema XML que debe utilizarse.

Parámetros de la estructura de datos db2DMUXmlValidate

iUsing
Entrada. Especificación de qué utilizar para realizar la validación de esquemas XML. Los valores válidos que se encuentran en el archivo de cabecera db2ApiDf del directorio de inclusión, son:

- DB2DMU_XMLVAL_XDS
La validación debe tener lugar de acuerdo con el XDS. Esto se corresponde con la cláusula CLP "**XMLVALIDATE USING XDS**".

- DB2DMU_XMLVAL_SCHEMA
La validación debe tener lugar de acuerdo con un esquema especificado. Esto se corresponde con la cláusula CLP "**XMLVALIDATE USING SCHEMA**".

- DB2DMU_XMLVAL_SCHEMALOC_HINTS
La validación debe tener lugar de acuerdo con las sugerencias de schemaLocation que se encuentran en el documento XML. Esto se corresponde con la cláusula "**XMLVALIDATE USING SCHEMALOCATION HINTS**".

piXdsArgs
Entrada. Puntero para una estructura db2DMUXmlValidateXds, representando argumentos que corresponden a la cláusula CLP "**XMLVALIDATE USING XDS**".

Este parámetro sólo se aplica cuando el parámetro **iUsing** de la misma estructura se establece en DB2DMU_XMLVAL_XDS.

piSchemaArgs
Entrada. Puntero para una estructura db2DMUXmlValidateSchema, representando argumentos que corresponden a la cláusula CLP "**XMLVALIDATE USING SCHEMA**".

Este parámetro sólo se aplica cuando el parámetro **iUsing** de la misma estructura se establece en DB2DMU_XMLVAL_SCHEMA.

Parámetros específicos de la estructura de datos db2gImportStruct

iDataFileNameLen
Entrada. Especifica la longitud en bytes del parámetro **piDataFileName**.

iFileTypeLen
Entrada. Especifica la longitud en bytes del parámetro **piFileType**.

iMsgFileNameLen
Entrada. Especifica la longitud en bytes del parámetro **piMsgFileName**.

Notas de uso

Antes de iniciar una operación de importación, asegúrese de completar todas las operaciones de tabla y de liberar todos los bloqueos de una de estas dos formas:

- Cierre todos los cursores abiertos que se han definido con la cláusula WITH HOLD y confirme los cambios en los datos ejecutando la sentencia COMMIT.
- Retrotraiga los cambios en los datos ejecutando la sentencia ROLLBACK.

El programa de utilidad de importación añade filas a la tabla de destino utilizando la sentencia INSERT de SQL.

El programa de utilidad emite una sentencia INSERT para cada fila de datos del archivo de entrada. Si falla una sentencia INSERT, se produce una de dos acciones:

- Si es probable que las sentencias INSERT subsiguientes puedan ser satisfactorias, se graba un mensaje de aviso en el archivo de mensajes y el proceso continúa.
- Si es probable que las sentencias INSERT subsiguientes fallen y existen posibilidades de que se dañe la base de datos, se graba un mensaje de error en el archivo de mensajes y el proceso se detiene.

El programa de utilidad realiza una operación COMMIT automática después de que se supriman las filas antiguas durante una operación **REPLACE** o **REPLACE_CREATE**. Por consiguiente, si el sistema falla o si la aplicación interrumpe el gestor de bases de datos después de que se trunque el objeto de tabla, se perderán todos los datos anteriores. Antes de utilizar estas opciones, asegúrese de que los datos anteriores ya no son necesarios.

Si la anotación cronológica se llena durante una operación **CREATE**, **REPLACE**, o **REPLACE_CREATE**, el programa de utilidad realiza una operación COMMIT automática en los registros insertados. Si el sistema falla o la aplicación interrumpe el gestor de bases de datos después de una operación COMMIT automática, permanece en la base de datos una tabla con datos parciales. Utilice la opción **REPLACE** o **REPLACE_CREATE** para volver a ejecutar toda la operación de importación, o bien utilice **INSERT** con el parámetro **iRestartcount** establecido en el número de filas importadas satisfactoriamente.

Por omisión, las operaciones COMMIT automáticas no se realizan para la opción **INSERT** o **INSERT_UPDATE**. Sin embargo, se realizan si el parámetro ***piCommitcount** no es cero. Una anotación cronológica llena producirá una retrotracción (ROLLBACK).

Siempre que el programa de utilidad de importación realiza un COMMIT, se graban dos mensajes en el archivo de mensajes: uno indica el número de registros que se deben confirmar y el otro se graba después de una operación COMMIT satisfactoria. Cuando reinicie la operación de importación después de una anomalía, especifique el número de registros que se deben saltar, según se determine en el último COMMIT satisfactorio.

El programa de utilidad de importación acepta datos de entrada con problemas de incompatibilidad menores (por ejemplo, se pueden importar datos de tipo carácter utilizando relleno o truncamiento y se pueden importar datos numéricos con un tipo de datos numéricos diferente), pero no se aceptan datos con problemas de incompatibilidad más importantes.

No se puede utilizar **REPLACE** o **REPLACE_CREATE** en una tabla de objeto si tiene dependientes distintos de ella misma, o una vista de objeto si su tabla base tiene algún dependiente (incluida ella misma). Para sustituir una tabla o una vista de este tipo, realice lo siguiente:

1. Descarte todas las claves foráneas en las que la tabla es padre.
2. Ejecute el programa de utilidad de importación.
3. Modifique la tabla para volver a crear las claves foráneas.

Si se produce un error al volver a crear las claves foráneas, modifique los datos para mantener la integridad referencial.

Las restricciones de referencias y las definiciones de claves foráneas no se conservan al crear tablas a partir de los archivos PC/IXF. (Las definiciones de claves primarias se conservan si anteriormente se han exportado los datos utilizando SELECT *.)

La importación a una base de datos remota requiere suficiente espacio de disco en el servidor para una copia del archivo de datos de entrada, para el archivo de mensajes de salida y para el aumento potencial de tamaño de la base de datos.

Si se ejecuta una operación de importación en una base de datos remota y el tamaño del archivo de mensajes de salida es muy grande (más de 60 KB), puede que el archivo de mensajes devuelto al usuario en el cliente no incluya los mensajes que se hayan producido durante la operación de importación. Los primeros y los últimos 30 KB de información de mensajes siempre se retienen.

Los valores que no son por omisión para **piDataDescriptor**, o la especificación de una lista explícita de columnas de tabla en **piLongActionString**, hace que la importación a una base de datos remota sea más lenta.

La tabla de base de datos o la jerarquía deben existir antes de que se puedan importar datos en formatos de archivo ASC, DEL o WSF; sin embargo, si la tabla aún no existe, IMPORT **CREATE** o IMPORT **REPLACE_CREATE** crean la tabla cuando importan datos desde un archivo PC/IXF. Para las tablas de tipo, IMPORT **CREATE** puede crear también la jerarquía de tipos y la jerarquía de tablas.

La importación PC/IXF deberá utilizarse para mover datos (incluidos datos jerárquicos) entre bases de datos. Si los datos de tipo carácter que contienen separadores de filas se exportan a un archivo ASCII delimitado (DEL) y se procesan con un programa de transferencia de texto, los campos que contengan separadores de filas se acortarán o se ampliarán.

Se supone que los datos de los archivos ASC y DEL están en la página de códigos de la aplicación cliente que realiza la importación. Se recomiendan los archivos PC/IXF, que permiten diferentes páginas de códigos, al importar datos de páginas de códigos diferentes. Si el archivo PC/IXF y el programa de utilidad de importación están en la misma página de códigos, el proceso se produce como para una aplicación normal. Si las dos páginas son diferentes y se especifica la opción **FORCEIN**, el programa de utilidad de importación presupone que los datos del archivo PC/IXF tienen la misma página de códigos que la aplicación que realiza la importación. Esto se produce incluso si existe una tabla de conversión para las dos páginas de códigos. Si las dos son diferentes, no se especifica la opción **FORCEIN** y hay una tabla de conversión, todos los datos del archivo PC/IXF se convertirán de la página de códigos del archivo a la página de códigos de la aplicación. Si las dos son diferentes, no se especifica la opción **FORCEIN** y no hay ninguna tabla de conversión, la operación de importación fallará. Esto se aplica únicamente a archivos PC/IXF en clientes de DB2 para AIX.

Para objetos de tabla de una página de 8KB que estén cerca del límite de 1012 columnas, la importación de archivos de datos PC/IXF puede hacer que DB2 devuelva un error, porque se ha excedido el tamaño máximo de una sentencia de SQL. Esta situación sólo se puede producir si las columnas son de tipo CHAR, VARCHAR o CLOB. La restricción no se aplica a la importación de archivos DEL o ASC.

Se puede utilizar DB2 Connect para importar datos en servidores DRDA como DB2 para OS/390, DB2 para VM y VSE y DB2 para OS/400. Sólo está soportada la importación PC/IXF (opción **INSERT**). El parámetro **restartcnt** también está soportado, pero no así el parámetro **commitcnt**.

Cuando utilice la opción CREATE con tablas de tipo, cree cada subtabla definida en el archivo PC/IXF; las definiciones de subtabla no se pueden modificar. Cuando utilice opciones distintas de CREATE con tablas de tipo, la lista de orden transversal permite especificar el orden transversal; por consiguiente, la lista de orden transversal debe coincidir con la que se ha utilizado durante la operación de exportación. Para el formato de archivo PC/IXF, sólo es necesario especificar el nombre de subtabla de destino y utilizar el orden transversal almacenado en el archivo. El programa de utilidad de importación se puede utilizar para recuperar una tabla exportada anteriormente a un archivo PC/IXF. La tabla vuelve al estado en el que estaba al exportarse.

No se pueden importar datos a una tabla de sistema, a una tabla declarada temporal o a una tabla de resumen.

No se pueden crear vistas mediante el programa de utilidad de importación.

En el sistema operativo Windows:

- No se soporta la importación de archivos PC/IXF subdivididos de forma lógica.
- No se soporta la importación de archivos PC/IXF o WSF de formato incorrecto.

Consideraciones federadas

Cuando se utiliza la API db2Import y los parámetros **INSERT**, **UPDATE** o **INSERT_UPDATE**, debe asegurarse de tener el privilegio CONTROL en el apodo participante. Debe asegurarse de que exista el apodo que desea utilizar en una operación de importación.

db2Inspect - Inspeccionar la base de datos para comprobar la integridad de la arquitectura

Inspecciona la base de datos para comprobar la integridad de la arquitectura y la coherencia de las páginas.

Ámbito

En un entorno de bases de datos de una sola partición, el ámbito es esa partición de base de datos solamente. En un entorno de bases de datos particionadas, es la colección de todas las particiones de base de datos lógicas definidas en `db2nodes.cfg`. Para tablas particionadas, el ámbito para la inspección de nivel de espacio de tablas y base de datos incluye particiones de datos individuales e índices no particionados. La inspección a nivel de tabla para una tabla particionada comprueba todas las particiones de datos e índices de una tabla, en lugar de comprobar una sola partición de datos o índice.

Autorización

Una de las siguientes:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*
- Privilegio CONTROL sobre la tabla

Conexión necesaria

Base de datos

Archivo de inclusión de la API

`db2ApiDf.h`

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2Inspect (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2InspectStruct
{
   char *piTablespaceName;
   char *piTableName;
   char *piSchemaName;
   char *piResultsName;
   char *piDataFileName;
   SQL_PDB_NODE_TYPE *piNodeList;
   db2Uint32 iAction;
```

```
   db2int32 iTablespaceID;
   db2int32 iObjectID;
   db2Uint32 iFirstPage;
   db2Uint32 iNumberOfPages;
   db2Uint32 iFormatType;
   db2Uint32 iOptions;
   db2Uint32 iBeginCheckOption;
   db2int32 iLimitErrorReported;
   db2Uint16 iObjectErrorState;
   db2Uint16 iCatalogToTablespace;
   db2Uint16 iKeepResultfile;
   db2Uint16 iAllNodeFlag;
   db2Uint16 iNumNodes;
   db2Uint16 iLevelObjectData;
   db2Uint16 iLevelObjectIndex;
   db2Uint16 iLevelObjectLong;
   db2Uint16 iLevelObjectLOB;
   db2Uint16 iLevelObjectBlkMap;
   db2Uint16 iLevelExtentMap;
   db2Uint16 iLevelObjectXML;
   db2Uint32 iLevelCrossObject;
} db2InspectStruct;

SQL_API_RC SQL_API_FN
  db2gInspect (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gInspectStruct
{
   char *piTablespaceName;
   char *piTableName;
   char *piSchemaName;
   char *piResultsName;
   char *piDataFileName;
   SQL_PDB_NODE_TYPE *piNodeList;
   db2Uint32 iResultsNameLength;
   db2Uint32 iDataFileNameLength;
   db2Uint32 iTablespaceNameLength;
   db2Uint32 iTableNameLength;
   db2Uint32 iSchemaNameLength;
   db2Uint32 iAction;
   db2int32 iTablespaceID;
   db2int32 iObjectID;
   db2Uint32 iFirstPage;
   db2Uint32 iNumberOfPages;
   db2Uint32 iFormatType;
   db2Uint32 iOptions;
   db2Uint32 iBeginCheckOption;
   db2int32 iLimitErrorReported;
   db2Uint16 iObjectErrorState;
   db2Uint16 iCatalogToTablespace;
   db2Uint16 iKeepResultfile;
   db2Uint16 iAllNodeFlag;
   db2Uint16 iNumNodes;
   db2Uint16 iLevelObjectData;
   db2Uint16 iLevelObjectIndex;
   db2Uint16 iLevelObjectLong;
   db2Uint16 iLevelObjectLOB;
   db2Uint16 iLevelObjectBlkMap;
   db2Uint16 iLevelExtentMap;
   db2Uint16 iLevelObjectXML;
   db2Uint32 iLevelCrossObject;
} db2gInspectStruct;
```

Parámetros de la API db2Inspect

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2InspectStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2InspectStruct

piTablespaceName
Entrada. Serie que contiene el nombre del espacio de tablas. Debe identificarse el espacio de tablas para operaciones en un espacio de tablas. Si el puntero es NULL, se utiliza como entrada el valor de ID de espacio de tablas.

piTableName
Entrada. Serie que contiene el nombre de tabla. Debe identificarse la tabla para operaciones en una tabla o en un objeto de tabla. Si el puntero es NULL, se utilizan como entrada los valores de ID de espacio de tablas e ID de objeto de tabla.

piSchemaName
Entrada. Serie que contiene el nombre de esquema.

piResultsName
Entrada. Serie que contiene el nombre del archivo de salida de resultados. Este necesario proporcionar esta entrada. El archivo se grabará en la vía de acceso del directorio de datos de diagnóstico.

piDataFileName
Entrada. Reservado para una utilización futura. Su valor se debe establecer en NULL.

piNodeList
Entrada. Puntero a una matriz de números de partición de base de datos en los que se realiza la operación.

iAction
Entrada. Especifica la acción de inspección. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2INSPECT_ACT_CHECK_DB
Inspecciona la base de datos completa.

DB2INSPECT_ACT_CHECK_TABSPACE
Inspecciona un espacio de tablas.

DB2INSPECT_ACT_CHECK_TABLE
Inspecciona una tabla.

DB2INSPECT_ACT_FORMAT_XML
Formatea una página de objeto XML.

DB2INSPECT_ACT_ROWCMPEST_TBL
Calcula la efectividad de la compresión de filas en una tabla.

iTablespaceID
Entrada. Especifica el ID de espacios de tablas. Si se debe identificar el

espacio de tablas, el valor del ID del espacio de tablas se utiliza como entrada si el puntero al nombre del espacio de tablas es NULL.

iObjectID
Entrada. Especifica el ID de objeto. Si se debe identificar la tabla, el valor del ID de objeto se utiliza como entrada si el puntero al nombre de tabla es NULL.

iBeginCheckOption
Entrada. Opción para indicar qué operación de comprobación de la base de datos o del espacio de tablas se debe iniciar. Su valor se debe establecer en cero para comenzar desde el inicio normal. Los valores son:

DB2INSPECT_BEGIN_TSPID
Utilice este valor para que el examen de la base de datos comience con el espacio de tablas especificado por el campo del ID de espacios de tablas. Para utilizar esta opción, el ID del espacio de tablas debe estar definido.

DB2INSPECT_BEGIN_TSPID_OBJID
Utilice este valor para que el examen de la base de datos comience con la tabla especificada por el campo del ID de espacio de tablas y del ID de objeto. Para utilizar esta opción, el ID de espacio de tablas y el ID de objeto deben estar definidos.

DB2INSPECT_BEGIN_OBJID
Utilice este valor para que el examen del espacio de tablas comience con la tabla especificada por el campo del ID de objeto. Para utilizar esta opción, el ID de objeto debe estar definido.

iLimitErrorReported
Entrada. Especifica el límite de páginas erróneas que se deben notificar para un objeto. Especifique el número que desee utilizar como valor límite o especifique uno de los valores siguientes:

DB2INSPECT_LIMIT_ERROR_DEFAULT
Utilice este valor para especificar que el número máximo de páginas erróneas que se deben notificar es el tamaño de extensión del objeto.

DB2INSPECT_LIMIT_ERROR_ALL
Utilice este valor para notificar todas las páginas erróneas.

Cuando se utiliza DB2INSPECT_LVL_XOBJ_INXDAT_RID en el campo iLevelCrossObject, el valor de límite especificado, o los valores DEFAULT o ALL anteriores, representan un límite en el número de errores, en lugar del número de páginas con error, que se deben informar en la comprobación en línea de la coherencia de los datos de índice.

iObjectErrorState
Entrada. Especifica si se deben explorar objetos en estado de error. Los valores válidos son:

DB2INSPECT_ERROR_STATE_NORMAL
Procesar objetos solamente en estado normal.

DB2INSPECT_ERROR_STATE_ALL
Procesar todos los objetos, incluidos los objetos en estado de error.

Cuando se utiliza DB2INSPECT_LVL_XOBJ_INXDAT_RID en el campo iLevelCrossObject, siempre que el objeto de índice o de datos esté en

estado erróneo, se pasará por alto DB2INSPECT_ERROR_STATE_ALL si se especifica en este campo, y no se realizará la comprobación en línea de la coherencia de los datos de índice.

iKeepResultfile
Entrada. Especifica la retención del archivo de resultados. Los valores válidos son:

DB2INSPECT_RESFILE_CLEANUP
Si se notifican errores, se retendrá el archivo de salida de resultados. En otro caso, el archivo de resultados se eliminará al final de la operación.

DB2INSPECT_RESFILE_KEEP_ALWAYS
Se retendrá el archivo de salida de resultados.

iAllNodeFlag
Entrada. Indica si la operación se debe aplicar a todos los nodos definidos en `db2nodes.cfg`. Los valores válidos son:

DB2_NODE_LIST
Aplicar a todos los nodos contenidos en la lista de nodos que se proporciona en pNodeList.

DB2_ALL_NODES
Aplicar a todos los nodos. pNodeList debe ser NULL. Es el valor por omisión.

DB2_ALL_EXCEPT
Aplicar a todos los nodos excepto los contenidos en la lista de nodos que se proporciona en pNodeList.

iNumNodes
Entrada. Especifica el número de nodos de la matriz pNodeList.

iLevelObjectData
Entrada. Especifica el nivel de proceso para el objeto de datos. Los valores válidos son:

DB2INSPECT_LEVEL_NORMAL
El nivel es normal.

DB2INSPECT_LEVEL_LOW
El nivel es bajo.

DB2INSPECT_LEVEL_NONE
El nivel es ninguno.

iLevelObjectIndex
Entrada. Especifica el nivel de proceso para el objeto de índice. Los valores válidos son:

DB2INSPECT_LEVEL_NORMAL
El nivel es normal.

DB2INSPECT_LEVEL_LOW
El nivel es bajo.

DB2INSPECT_LEVEL_NONE
El nivel es ninguno.

iLevelObjectLong
Entrada. Especifica el nivel de proceso para el objeto largo. Los valores válidos son:

DB2INSPECT_LEVEL_NORMAL
El nivel es normal.

DB2INSPECT_LEVEL_LOW
El nivel es bajo.

DB2INSPECT_LEVEL_NONE
El nivel es ninguno.

iLevelObjectLOB
Entrada. Especifica el nivel de proceso para el objeto LOB. Los valores válidos son:

DB2INSPECT_LEVEL_NORMAL
El nivel es normal.

DB2INSPECT_LEVEL_LOW
El nivel es bajo.

DB2INSPECT_LEVEL_NONE
El nivel es ninguno.

iLevelObjectBlkMap
Entrada. Especifica el nivel de proceso para el objeto de correlación de bloques. Los valores válidos son:

DB2INSPECT_LEVEL_NORMAL
El nivel es normal.

DB2INSPECT_LEVEL_LOW
El nivel es bajo.

DB2INSPECT_LEVEL_NONE
El nivel es ninguno.

iLevelExtentMap
Entrada. Especifica el nivel de proceso para la correlación de extensión. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2INSPECT_LEVEL_NORMAL
El nivel es normal.

DB2INSPECT_LEVEL_LOW
El nivel es bajo.

DB2INSPECT_LEVEL_NONE
El nivel es ninguno.

iLevelObjectXML
Entrada. Especifica el nivel de proceso de un objeto XML. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2INSPECT_LEVEL_NORMAL
El nivel es normal.

DB2INSPECT_LEVEL_LOW
El nivel es bajo.

DB2INSPECT_LEVEL_NONE
El nivel es ninguno.

iLevelCrossObject
Campo basado en bits que se utiliza para cualquier comprobación cruzada de coherencia de objetos. Los valores válidos son:

DB2INSPECT_LVL_XOBJ_NONE
No se realizará la comprobación en línea de la coherencia de los datos de índice (0x00000000).

DB2INSPECT_LVL_XOBJ_INXDAT_RID
La comprobación INDEXDATA está habilitada en el índice RID (0x00000001) y se realizará con el bloqueo de tabla IS para permitir la lectura y la grabación.

Parámetros específicos de la estructura de datos db2gInspectStruct

iResultsNameLength
Entrada. Longitud de serie del nombre del archivo de resultados.

iDataFileNameLength
Entrada. Longitud de serie del nombre del archivo de salida de datos.

iTablespaceNameLength
Entrada. Longitud de serie del nombre del espacio de tablas.

iTableNameLength
Entrada. Longitud de serie del nombre de tabla.

iSchemaNameLength
Entrada. Longitud de serie del nombre de esquema.

Notas de uso

El proceso de inspección en línea accederá a objetos de base de datos utilizando la lectura no confirmada a nivel de aislamiento. El proceso de confirmación se realizará durante el proceso de inspección. Es aconsejable finalizar la unidad de trabajo mediante la confirmación o retrotracción de los cambios (ejecutando una sentencia COMMIT o ROLLBACK respectivamente) antes de iniciar la operación de inspección.

El proceso de inspección escribe los resultados del proceso, sin formato, en el archivo de resultados. El archivo se grabará en la vía de acceso del directorio de datos de diagnóstico. Si el proceso de comprobación no encuentra errores, el archivo de salida de resultados se borrará al final de la operación de inspección. Si el proceso de comprobación encuentra errores, el archivo de salida de resultados no se borrará al final de la operación de inspección. Para ver los detalles de la inspección, formatee el archivo de salida de resultados de la inspección con el programa de utilidad db2inspf.

En un entorno de bases de datos particionadas, la extensión del archivo de salida de resultados se corresponde con el número de partición de la base de datos. El archivo reside en la vía de acceso del directorio de datos de diagnóstico del gestor de bases de datos.

Se debe especificar un nombre de archivo de salida de resultados exclusivo. Si el archivo de salida de resultados ya existe, la operación no se procesará.

Cuando se llama a la API db2Inspect, es necesario especificar `iLevelCrossObject` en db2InspectStruct con un valor correcto. Cuando se utiliza

DB2INSPECT_LVL_XOBJ_NONE, no se realizará la comprobación en línea de la coherencia de los datos de índice. Para habilitar la comprobación en línea de la coherencia de los datos de índice, es necesario especificar DB2INSPECT_LVL_XOBJ_INXDAT_RID en el campo `iLevelCrossObject`.

El proceso de espacios de tablas solamente procesa los objetos que residen en el espacio de tablas especificado. La excepción se produce durante una comprobación de la coherencia de los datos de índice, cuando los objetos de datos pueden residir en otros espacios de tablas y seguir beneficiándose de la comprobación, siempre que los objetos de índice estén en el espacio de tablas que se debe inspeccionar. Para una tabla particionada, cada índice puede residir en un espacio de tablas diferente. Sólo los índices que residen en el espacio de tablas especificado se beneficiarán de la comprobación de los datos de índice.

db2InstanceQuiesce - Inmovilizar instancia

Desconecta de la instancia a todos los usuarios, e inmediatamente retrotrae todas las transacciones activas y coloca la base de datos en la modalidad de inmovilización. Esta API proporciona acceso exclusivo a la instancia. Durante este período de inmovilización puede realizarse la administración del sistema en la instancia. Una vez se ha completado la administración, puede desinmovilizar la base de datos utilizando la API db2DatabaseUnquiesce. Esta API permite a otros usuarios conectarse a la base de datos, sin tener que concluir y realizar otro inicio de la base de datos.

En esta modalidad, solamente los grupos o usuarios con autorización QUIESCE CONNECT y autorización sysadm, sysmaint o sysctrl tendrán acceso a la base de datos y a sus objetos.

Autorización

Una de las siguientes:

- sysadm
- sysctrl

Conexión necesaria

Ninguna

Archivo de inclusión de la API

`db2ApiDf.h`

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2InstanceQuiesce (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2InsQuiesceStruct
{
             char *piInstanceName;
             char *piUserId;
             char *piGroupId;
             db2Uint32 iImmediate;
             db2Uint32 iForce;
             db2Uint32 iTimeout;
```

```
} db2InsQuiesceStruct;

SQL_API_RC SQL_API_FN
  db2gInstanceQuiesce (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gInsQuiesceStruct
{
             db2Uint32 iInstanceNameLen;
             char *piInstanceName;
             db2Uint32 iUserIdLen;
             char *piUserId;
             db2Uint32 iGroupIdLen;
             char *piGroupId;
             db2Uint32 iImmediate;
             db2Uint32 iForce;
             db2Uint32 iTimeout;
} db2gInsQuiesceStruct;
```

Parámetros de la API db2InstanceQuiesce

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2InsQuiesceStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2InsQuiesceStruct

piInstanceName
Entrada. Nombre de la instancia.

piUserId
Entrada. Nombre del usuario a quien se permitirá acceder a la instancia mientras está inmovilizada.

piGroupId
Entrada. Nombre de un grupo a quien se permitirá acceder a la instancia mientras está inmovilizada.

iImmediate
Entrada. Los valores válidos son:

TRUE=1
Desconectar las aplicaciones inmediatamente.

FALSE=0
Desconexión diferida. Las aplicaciones esperarán el número de minutos especificado en el parámetro iTimeout para permitir que finalicen sus unidades de trabajo actuales, y luego concluirán. Si esta desconexión forzada no se puede realizar dentro del número de minutos especificado por el parámetro iTimeout, la operación de inmovilización fallará.

iForce Entrada. Reservado para una utilización futura.

iTimeout
Entrada. Especifica el tiempo, en minutos, que se debe esperar para que las aplicaciones confirmen la unidad actual de trabajo. Si no se especifica

iTimeout, en un entorno de bases de datos de una sola partición, el valor por omisión es 10 minutos. En un entorno de bases de datos particionadas, se utilizará el valor especificado por el parámetro de configuración de gestor de bases de datos start_stop_time.

Parámetros específicos de la estructura de datos db2gInsQuiesceStruct

iInstanceNameLen
Entrada. Especifica la longitud, en bytes, de piInstanceName.

iUserIdLen
Entrada. Especifica la longitud, en bytes, de piUserID.

iGroupIdLen
Entrada. Especifica la longitud, en bytes, de piGroupId.

---

db2InstanceStart - Iniciar instancia

Inicia una instancia local o remota.

Ámbito

En un entorno de bases de datos de una sola partición, el ámbito es esa partición de base de datos solamente. En un entorno de bases de datos particionadas, es la colección de todos los servidores de particiones lógicas de base de datos definidos en el archivo de configuración de nodos, `db2nodes.cfg`.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint

Conexión necesaria

Ninguna

Archivo de inclusión de la API

`db2ApiDf.h`

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2InstanceStart (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2InstanceStartStruct
{
             db2int8 iIsRemote;
             char *piRemoteInstName;
             db2DasCommData * piCommData;
             db2StartOptionsStruct * piStartOpts;
} db2InstanceStartStruct;

typedef SQL_STRUCTURE db2DasCommData
```

```
{
             db2int8 iCommParam;
             char *piNodeOrHostName;
             char *piUserId;
             char *piUserPw;
} db2DasCommData;

typedef SQL_STRUCTURE db2StartOptionsStruct
{
             db2Uint32 iIsProfile;
             char *piProfile;
             db2Uint32 iIsNodeNum;
             db2NodeType iNodeNum;
             db2Uint32 iOption;
             db2Uint32 iIsHostName;
             char *piHostName;
             db2Uint32 iIsPort;
             db2PortType iPort;
             db2Uint32 iIsNetName;
             char *piNetName;
             db2Uint32 iTblspaceType;
             db2NodeType iTblspaceNode;
             db2Uint32 iIsComputer;
             char *piComputer;
             char *piUserName;
             char *piPassword;
             db2QuiesceStartStruct iQuiesceOpts;
} db2StartOptionsStruct;

typedef SQL_STRUCTURE db2QuiesceStartStruct
{
             db2int8 iIsQRequested;
             char *piQUsrName;
             char *piQGrpName;
             db2int8 iIsQUsrGrpDef;
} db2QuiesceStartStruct;

SQL_API_RC SQL_API_FN
  db2gInstanceStart (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gInstanceStStruct
{
             db2int8 iIsRemote;
             db2Uint32 iRemoteInstLen;
             char *piRemoteInstName;
             db2gDasCommData * piCommData;
             db2gStartOptionsStruct * piStartOpts;
} db2gInstanceStStruct;

typedef SQL_STRUCTURE db2gDasCommData
{
             db2int8 iCommParam;
             db2Uint32 iNodeOrHostNameLen;
             char *piNodeOrHostName;
             db2Uint32 iUserIdLen;
             char *piUserId;
             db2Uint32 iUserPwLen;
             char *piUserPw;
} db2gDasCommData;

typedef SQL_STRUCTURE db2gStartOptionsStruct
{
             db2Uint32 iIsProfile;
             char *piProfile;
```

```
             db2Uint32 iIsNodeNum;
             db2NodeType iNodeNum;
             db2Uint32 iOption;
             db2Uint32 iIsHostName;
             char *piHostName;
             db2Uint32 iIsPort;
             db2PortType iPort;
             db2Uint32 iIsNetName;
             char *piNetName;
             db2Uint32 iTblspaceType;
             db2NodeType iTblspaceNode;
             db2Uint32 iIsComputer;
             char *piComputer;
             char *piUserName;
             char *piPassword;
             db2gQuiesceStartStruct iQuiesceOpts;
} db2gStartOptionsStruct;

typedef SQL_STRUCTURE db2gQuiesceStartStruct
{
             db2int8 iIsQRequested;
             db2Uint32 iQUsrNameLen;
             char *piQUsrName;
             db2Uint32 iQGrpNameLen;
             char *piQGrpName;
             db2int8 iIsQUsrGrpDef;
} db2gQuiesceStartStruct;
```

Parámetros de la API db2InstanceStart

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2InstanceStartStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2InstanceStartStruct

iIsRemote
Entrada. Indicador establecido en un valor entero constante TRUE o FALSE. Este parámetro debe establecerse en TRUE en caso de que se trate de un inicio remoto.

piRemoteInstName
Entrada. Nombre de la instancia remota.

piCommData
Entrada. Puntero a la estructura db2DasCommData.

piStartOpts
Entrada. Puntero a la estructura db2StartOptionsStruct.

Parámetros de la estructura de datos db2DasCommData

iCommParam
Entrada. Indicador establecido en el valor TRUE o FALSE. Este parámetro debe establecerse en TRUE en caso de que se trate de un inicio remoto.

piNodeOrHostName
Entrada. Partición de base de datos o nombre de sistema principal.

piUserId
Entrada. Nombre del usuario.

piUserPw
Entrada. Contraseña del usuario.

Parámetros de la estructura de datos db2StartOptionsStruct

iIsProfile
Entrada. Indica si se especifica un perfil. Si este campo indica que no se ha especificado un perfil, se utilizará el archivo db2profile.

piProfile
Entrada. Nombre del archivo del perfil que se va a ejecutar en cada nodo para definir el entorno DB2 (MPP solamente). Este archivo se ejecuta antes de que se inicien los nodos. El valor por omisión es db2profile.

iIsNodeNum
Entrada. Indica si se especifica un número de nodo. Si se especifica, el mandato de inicio sólo afecta al nodo especificado.

iNodeNum
Entrada. Número de partición de base de datos.

iOption
Entrada. Especifica una acción. Los valores válidos para OPTION (definidos en el archivo de cabecera sqlenv, ubicado en el directorio de inclusión) son:

SQLE_NONE
Emite la operación db2start normal.

SQLE_ADDNODE
Emite el mandato ADD NODE.

SQLE_RESTART
Emite el mandato RESTART DATABASE.

SQLE_RESTART_PARALLEL
Emita el mandato RESTART DATABASE para la ejecución en paralelo.

SQLE_STANDALONE
Inicia el nodo en la modalidad STANDALONE.

iIsHostName
Entrada. Indica si se especifica un nombre de sistema principal.

piHostName
Entrada. Nombre del sistema.

iIsPort
Entrada. Indica si se especifica un número de puerto.

iPort Entrada. Número de puerto.

iIsNetName
Entrada. Indica si se especifica un nombre de red.

piNetName
Entrada. Nombre de la red.

iTblspaceType
Entrada. Especifica el tipo de definiciones de espacios de tablas temporales del sistema que se debe utilizar para el nodo que se está añadiendo. Los valores válidos son:

SQLE_TABLESPACES_NONE
No crear ningún espacio de tablas temporales del sistema.

SQLE_TABLESPACES_LIKE_NODE
Los contenedores para los espacios de tablas temporales del sistema deben ser los mismos que los utilizados para el nodo especificado.

SQLE_TABLESPACES_LIKE_CATALOG
Los contenedores para los espacios de tablas temporales del sistema deben ser los mismos que los utilizados para el nodo de catálogo de cada base de datos.

iTblspaceNode
Entrada. Especifica el número de nodo del que deben obtenerse definiciones de espacios de tablas temporales del sistema. El número de nodo debe existir en el archivo db2nodes.cfg y solamente se utiliza si el campo tblspace_type tiene el valor SQLE_TABLESPACES_LIKE_NODE.

iIsComputer
Entrada. Indica si se especifica un nombre de sistema. Sólo es válido en el sistema operativo Windows.

piComputer
Entrada. Nombre del sistema. Sólo es válido en el sistema operativo Windows.

piUserName
Entrada. Nombre de usuario de cuenta de conexión. Sólo es válido en el sistema operativo Windows.

piPassword
Entrada. Contraseña que corresponde al nombre de usuario de la cuenta de conexión.

iQuiesceOpts
Entrada. Puntero de la estructura db2QuiesceStartStruct.

Parámetros de la estructura de datos db2QuiesceStartStruct

iIsQRequested
Entrada. Indicador establecido en el valor TRUE o FALSE. Este parámetro debe establecerse en TRUE si se solicita la inmovilización.

piQUsrName
Entrada. Nombre de usuario inmovilizado.

piQGrpName
Entrada. Nombre del grupo inmovilizado.

iIsQUsrGrpDef
Entrada. Indicador establecido en el valor TRUE o FALSE. Este parámetro debe establecerse en TRUE si se ha definido el usuario inmovilizado o el grupo inmovilizado.

Parámetros específicos de la estructura de datos db2gInstanceStStruct

iRemoteInstLen
Entrada. Especifica la longitud, en bytes, de piRemoteInstName.

Parámetros específicos de la estructura de datos db2gDasCommData

iNodeOrHostNameLen
Entrada. Especifica la longitud, en bytes, de piNodeOrHostName.

iUserIdLen
Entrada. Especifica la longitud, en bytes, de piUserId.

iUserPwLen
Entrada. Especifica la longitud, en bytes, de piUserPw.

Parámetros específicos de la estructura de datos db2gQuiesceStartStruct

iQUsrNameLen
Entrada. Especifica la longitud, en bytes, de piQusrName.

iQGrpNameLen
Entrada. Especifica la longitud, en bytes, de piQGrpName.

db2InstanceStop - Detener instancia

Detiene la instancia local o remota de DB2.

Ámbito

En un entorno de bases de datos de una sola partición, el ámbito es esa partición de base de datos solamente. En un entorno de bases de datos particionadas, el ámbito es el conjunto de todos los servidores de particiones lógicas de base de datos definidos en el archivo de configuración de nodos, db2nodes.cfg.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2InstanceStop (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);
```

```
typedef SQL_STRUCTURE db2InstanceStopStruct
{
             db2int8 iIsRemote;
             char *piRemoteInstName;
             db2DasCommData * piCommData;
             db2StopOptionsStruct * piStopOpts;
} db2InstanceStopStruct;

typedef SQL_STRUCTURE db2DasCommData
{
             db2int8 iCommParam;
             char *piNodeOrHostName;
             char *piUserId;
             char *piUserPw;
} db2DasCommData;

typedef SQL_STRUCTURE db2StopOptionsStruct
{
             db2Uint32 iIsProfile;
             char *piProfile;
             db2Uint32 iIsNodeNum;
             db2NodeType iNodeNum;
             db2Uint32 iStopOption;
             db2Uint32 iCallerac;
} db2StopOptionsStruct;

SQL_API_RC SQL_API_FN
  db2gInstanceStop (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gInstanceStopStruct
{
             db2int8 iIsRemote;
             db2Uint32 iRemoteInstLen;
             char *piRemoteInstName;
             db2gDasCommData * piCommData;
             db2StopOptionsStruct * piStopOpts;
} db2gInstanceStopStruct;

typedef SQL_STRUCTURE db2gDasCommData
{
             db2int8 iCommParam;
             db2Uint32 iNodeOrHostNameLen;
             char *piNodeOrHostName;
             db2Uint32 iUserIdLen;
             char *piUserId;
             db2Uint32 iUserPwLen;
             char *piUserPw;
} db2gDasCommData;
```

Parámetros de la API db2InstanceStop

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2InstanceStopStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2InstanceStopStruct

iIsRemote
Entrada. Indicador establecido en el valor TRUE o FALSE. Este parámetro debe establecerse en TRUE en caso de que se trate de un inicio remoto.

piRemoteInstName
Entrada. Nombre de la instancia remota.

piCommData
Entrada. Puntero a la estructura db2DasCommData.

piStopOpts
Entrada. Puntero a la estructura db2StopOptionsStruct.

Parámetros de la estructura de datos db2DasCommData

iCommParam
Entrada. Indicador establecido en el valor TRUE o FALSE. Este parámetro debe establecerse en TRUE en caso de que se trate de un inicio remoto.

piNodeOrHostName
Entrada. Partición de base de datos o nombre de sistema principal.

piUserId
Entrada. Nombre del usuario.

piUserPw
Entrada. Contraseña del usuario.

Parámetros de la estructura de datos db2StopOptionsStruct

iIsProfile
Entrada. Indica si se especifica un perfil. Los valores posibles son TRUE y FALSE. Si este campo indica que no se ha especificado un perfil, se utilizará el archivo db2profile.

piProfile
Entrada. Nombre del archivo del perfil que se ha ejecutado en el arranque para definir el entorno DB2 para aquellos nodos que se han iniciado (MPP solamente). Si se ha especificado un perfil para la API db2InstanceStart, aquí debe especificarse el mismo perfil.

iIsNodeNum
Entrada. Indica si se especifica un número de nodo. Los valores posibles son TRUE y FALSE. Si se especifica, el mandato de detención solo afecta al nodo especificado.

iNodeNum
Entrada. Número de partición de base de datos.

iStopOption
Entrada. Opción. Los valores válidos son:

SQLE_NONE
Emitir la operación db2stop normal.

SQLE_FORCE
Emitir el mandato FORCE APPLICATION (ALL).

SQLE_DROP
Descartar el nodo del archivo db2nodes.cfg.

iCallerac
Entrada. Este campo solamente es válido para el valor SQLE_DROP del campo OPTION. Los valores válidos son:

SQLE_DROP
Llamada inicial. Es el valor por omisión.

SQLE_CONTINUE
Llamada subsiguiente. Continuar el proceso después de un mensaje de solicitud.

SQLE_TERMINATE
Llamada subsiguiente. Terminar el proceso después de un mensaje de solicitud.

Parámetros específicos de la estructura de datos db2gInstanceStopStruct

iRemoteInstLen
Entrada. Especifica la longitud, en bytes, de piRemoteInstName.

Parámetros específicos de la estructura de datos db2gDasCommData

iNodeOrHostNameLen
Entrada. Especifica la longitud, en bytes, de piNodeOrHostName.

iUserIdLen
Entrada. Especifica la longitud, en bytes, de piUserId.

iUserPwLen
Entrada. Especifica la longitud, en bytes, de piUserPw.

db2InstanceUnquiesce - Movilizar instancia

Moviliza todas las bases de datos de la instancia.

Autorización

Una de las siguientes:
- sysadm
- sysctrl

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2InstanceUnquiesce (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2InsUnquiesceStruct
{
```

```
            char *piInstanceName;
} db2InsUnquiesceStruct;

SQL_API_RC SQL_API_FN
  db2gInstanceUnquiesce (
            db2Uint32 versionNumber,
            void * pParmStruct,
            struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gInsUnquiesceStruct
{
            db2Uint32 iInstanceNameLen;
            char *piInstanceName;
} db2gInsUnquiesceStruct;
```

Parámetros de la API db2InstanceUnquiesce

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2InsUnquiesceStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2InsUnquiesceStruct

piInstanceName
Entrada. Nombre de la instancia.

Parámetros específicos de la estructura de datos db2gInsUnquiesceStruct

iInstanceNameLen
Entrada. Especifica la longitud, en bytes, de piInstanceName.

---

db2LdapCatalogDatabase - Registrar la base de datos en el servidor LDAP

Cataloga una entrada de base de datos en LDAP (Lightweight Directory Access Protocol).

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2LdapCatalogDatabase (
   db2Uint32 versionNumber,
   void * pParmStruct,
```

```
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LdapCatalogDatabaseStruct
{
   char *piAlias;
   char *piDatabaseName;
   char *piComment;
   char *piNodeName;
   char *piGWNodeName;
   char *piParameters;
   char *piARLibrary;
   unsigned short                       iAuthentication;
   char *piDCEPrincipalName;
   char *piBindDN;
   char *piPassword;
} db2LdapCatalogDatabaseStruct;
```

Parámetros de la API db2LdapCatalogDatabase

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParamStruct.

pParamStruct
Entrada. Puntero a la estructura db2LdapCatalogDatabaseStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2LdapCatalogDatabaseStruct

piAlias
Entrada. Especifica un alias que se utilizará como nombre alternativo para la base de datos que se está catalogando. Si no se especifica ningún alias, el gestor de bases de datos utiliza el nombre de base de datos como nombre de alias.

piDatabaseName
Entrada. Especifique el nombre de la base de datos a catalogar. Es parámetro es obligatorio.

piComment
Entrada. Describe el servidor DB2. Se puede entrar cualquier comentario que ayude a describir el servidor registrado en el directorio de red. La longitud máxima es de 30 caracteres. No se permite ningún retorno de carro o carácter de salto de línea.

piNodeName
Entrada. Especifique el nombre de nodo del servidor de bases de datos en el que reside la base de datos. Este parámetro es necesario si la base de datos reside en un servidor remoto de bases de datos.

piGWNodename
Entrada. Especifique el nombre de nodo del servidor de pasarela de DB2 Connect. Si el tipo de nodo del servidor de bases de datos es DCS (reservado para servidores de bases de datos de sistema principal), y el cliente no tiene instalado DB2 Connect, el cliente se conectará al servidor de pasarela de DB2 Connect.

piParameters
Entrada. Especifique un parámetro para pasarlo al peticionario de

aplicaciones. La autentificación de DCE no se puede utilizar. La autentificación de DCE no está soportada.

piARLibrary
Entrada. Especifique el nombre de la biblioteca del peticionario de aplicaciones.

iAuthentication
Entrada. La especificación de un tipo de autentificación puede producir un beneficio en el rendimiento.

piDCEPrincipalName
Entrada. Especifique el nombre de principal de DCE totalmente calificado para el servidor de destino.

piBindDN
Entrada. Especifique el nombre distinguido (DN) LDAP del usuario. El DN LDAP de usuario debe tener suficiente autorización para crear y actualizar el objeto del directorio LDAP. Si no se especifica el nombre DN LDAP del usuario, se utilizarán las credenciales del usuario de la conexión actual.

piPassword
Entrada. Contraseña de cuenta.

Notas de uso

Puede ser necesario registrar o catalogar manualmente una base de datos en LDAP si:

- El servidor de bases de datos no es compatible con LDAP. En este caso, el administrador necesita registrar manualmente cada base de datos en LDAP para que los clientes que utilizan LDAP puedan acceder a la base de datos sin tener que catalogar la base de datos localmente en cada máquina cliente.

- La aplicación desea utilizar un nombre diferente para conectar con la base de datos. En este caso, el administrador necesita catalogar la base de datos utilizando un alias diferente.

- Durante la operación CREATE DATABASE IN LDAP, el nombre de base de datos ya existe en LDAP. La base de datos se crea de todos modos en la máquina local (y pueden acceder a ella las aplicaciones locales), pero la entrada existente en LDAP no se modificará para reflejar la nueva base de datos. En este caso, el administrador puede: -- Suprimir la entrada de base de datos existente en LDAP y registrar manualmente la nueva base de datos en LDAP. -- Registrar la nueva base de datos en LDAP utilizando un alias diferente.

db2LdapCatalogNode - Proporcionar un alias para el nombre de nodo en el servidor LDAP

Especifica un nombre alternativo para la entrada de nodo en LDAP (Lightweight Directory Access Protocol), o un tipo de protocolo diferente para conectar con el servidor de bases de datos.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2LdapCatalogNode (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LdapCatalogNodeStruct
{
   char *piAlias;
   char *piNodeName;
   char *piBindDN;
   char *piPassword;
} db2LdapCatalogNodeStruct;
```

Parámetros de la API db2LdapCatalogNode

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParamStruct.

pParamStruct
Entrada. Puntero a la estructura db2LdapCatalogNodeStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2LdapCatalogNodeStruct

piAlias
Entrada. Especifique un nuevo alias para utilizarlo como nombre alternativo para la entrada de nodo.

piNodeName
Entrada. Especifique un nombre de nodo que represente al servidor DB2 en LDAP.

piBindDN
Entrada. Especifique el nombre distinguido (DN) LDAP del usuario. El DN LDAP de usuario debe tener suficiente autorización para crear y actualizar el objeto del directorio LDAP. Si no se especifica el nombre DN LDAP del usuario, se utilizarán las credenciales del usuario de la conexión actual.

piPassword
Entrada. Contraseña de cuenta.

db2LdapDeregister - Desregistrar el servidor DB2 y las bases de datos catalogadas del servidor LDAP

Desregistra el servidor DB2 de LDAP (Lightweight Directory Access Protocol).

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2LdapDeregister (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LdapDeregisterStruct
{
   char *piNodeName;
   char *piBindDN;
   char *piPassword;
} db2LdapDeregisterStruct;
```

Parámetros de la API db2LdapDeregister

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParamStruct.

pParamStruct
Entrada. Puntero a la estructura db2LdapDeregisterStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2LdapDeregisterStruct

piNodeName
Entrada. Especifique un nombre corto que represente al servidor DB2 en LDAP.

piBindDN
Entrada. Especifique el nombre distinguido (DN) LDAP del usuario. El DN LDAP de usuario debe tener suficiente autorización para suprimir el objeto del directorio LDAP. Si no se especifica el nombre DN LDAP del usuario, se utilizarán las credenciales del usuario de la conexión actual.

piPassword
Entrada. Contraseña de cuenta.

db2LdapRegister - Registrar el servidor DB2 en el servidor LDAP

Registra el servidor DB2 en LDAP (Lightweight Directory Access Protocol).

Nota: NetBIOS ya no está soportado. SNA, incluyendo sus API APPC, APPN y CPI-C, tampoco está soportado. Si utiliza estos protocolos, debe volver a catalogar los nodos y bases de datos utilizando un protocolo soportado como, por ejemplo, TCP/IP. Las referencias a estos protocolos se deben pasar por alto.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

db2ApiDf.h

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2LdapRegister (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LdapRegisterStruct
{
   char *piNodeName;
   char *piComputer;
   char *piInstance;
   unsigned short                       iNodeType;
   unsigned short                       iOsType;
   db2LdapProtocolInfo iProtocol;
   char *piComment;
   char *piBindDN;
   char *piPassword;
} db2LdapRegisterStruct;

typedef SQL_STRUCTURE db2LdapProtocolInfo
{
   char iType;
   char *piHostName;
   char *piServiceName;
   char *piNetbiosName;
   char *piNetworkId;
   char *piPartnerLU;
   char *piTPName;
   char *piMode;
   unsigned short                       iSecurityType;
   char *piLanAdapterAddress;
   char *piChangePasswordLU;
   char *piIpxAddress;
} db2LdapProtocolInfo;
```

Parámetros de la API db2LdapRegister

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParamStruct.

pParamStruct
Entrada. Puntero a la estructura db2LdapRegisterStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2LdapRegisterStruct

piNodeName
Entrada. Especifique un nombre corto (menos de 8 caracteres) que represente el servidor DB2 en LDAP.

piComputer
Entrada. Especifique el nombre del sistema en el que reside el servidor DB2. El valor del nombre de sistema debe ser el mismo que el especificado al añadir la máquina servidor a LDAP. En sistemas operativos Windows, es el nombre de sistema Windows. En sistemas basados en UNIX, es el nombre de sistema principal TCP/IP. Especifique NULL para registrar el servidor DB2 en el sistema local.

piInstance
Entrada. Especifique el nombre de instancia del servidor DB2. El nombre de instancia se debe especificar si se especifica el nombre de sistema para registrar un servidor remoto. Especifique NULL para registrar la instancia actual (tal como está definida por la variable de entorno DB2SYSTEM).

iNodeType
Entrada. Especifique el tipo de nodo para el servidor de bases de datos. Los valores válidos son:

- SQLF_NT_SERVER
- SQLF_NT_MPP
- SQLF_NT_DCS

iOsType
Entrada. Especifica el tipo de sistema operativo de la máquina servidor. Si no se especifica un tipo de sistema operativo, se utilizará el tipo de sistema operativo local para un servidor local y no se utilizará ningún tipo de sistema operativo para un servidor remoto.

iProtocol
Entrada. Especifique la información de protocolo en la estructura db2LdapProtocolInfo.

piComment
Entrada. Describe el servidor DB2. Se puede entrar cualquier comentario que ayude a describir el servidor registrado en el directorio de red. La longitud máxima es de 30 caracteres. No se permite ningún retorno de carro o carácter de salto de línea.

piBindDN
Entrada. Especifique el nombre distinguido (DN) LDAP del usuario. El DN LDAP de usuario debe tener suficiente autorización para crear y actualizar el objeto del directorio LDAP. Si no se especifica el nombre DN LDAP del usuario, se utilizarán las credenciales del usuario de la conexión actual.

piPassword
Entrada. Contraseña de cuenta.

Parámetros de la estructura de datos db2LdapProtocolInfo

iType Entrada. Especifique el tipo de protocolo que soporta el servidor. Si el servidor es compatible con más de un protocolo, es necesario realizar varios procesos de registro (cada uno con un nombre de nodo y tipo de protocolo diferente). Los valores válidos son:

SQL_PROTOCOL_TCPIP
Para el soporte de TCP/IPv4 o TCP/IPv6

SQL_PROTOCOL_TCPIP4
Para el soporte de TCP/IPv4

SQL_PROTOCOL_TCPIP6
Para el soporte de TCP/IPv6

SQL_PROTOCOL_SOCKS
Para TCP/IP con SOCKS de seguridad

SQL_PROTOCOL_SOCKS4
Para TCP/IPv4 con SOCKS de seguridad

SQL_PROTOCOL_NPIPE
Para el soporte Conexiones con nombre de Windows

piHostName
Entrada. Especifique el nombre de sistema principal TCP/IP o la dirección IP. La dirección IP puede ser una dirección IPv4 o IPv6. Si se especifica una dirección IP, debe coincidir con el tipo de protocolo seleccionado. Por ejemplo, si SQL_PROTOCOL_TCPIP4 está seleccionado, la dirección IP especificada debe ser una dirección IPv4.

piServiceName
Entrada. Especifique el nombre de servicio o el número de puerto TCP/IP.

piNetbiosName
Entrada. Especifique el nombre de estación de trabajo de NetBIOS. El nombre de NetBIOS debe especificarse para el soporte de NetBIOS.

piNetworkID
Entrada. Especifique el ID de red. El ID de red se debe especificar para el soporte de APPC/APPN.

piPartnerLU
Entrada. Especifique el nombre de LU asociada para la máquina servidor DB2. La LU asociada debe especificarse para el soporte APPC/APPN.

piTPName
Entrada. Especifique el nombre de programa de transacción. El nombre de programa de transacción debe especificarse para el soporte de APPC/APPN.

piMode
Entrada. Especifique el nombre de modalidad. La modalidad debe especificarse para el soporte de APPC/APPN.

iSecurityType
Entrada. Especifique el nivel de seguridad de APPC. Los valores válidos son:

- SQL_CPIC_SECURITY_NONE (valor por omisión)
- SQL_CPIC_SECURITY_SAME
- SQL_CPIC_SECURITY_PROGRAM

piLanAdapterAddress
Entrada. Especifique la dirección del adaptador de red. Este parámetro sólo es necesario para el soporte de APPC. Para APPN, se puede establecer en NULL.

piChangePasswordLU
Entrada. Especifique el nombre de la LU asociada que se debe utilizar al cambiar la contraseña para el servidor de bases de datos de sistema principal.

piIpxAddress
Entrada. Especifique la dirección IPX completa. La dirección IPX se debe especificar para el soporte de IPX/SPX.

Notas de uso

Registre el servidor DB2 una vez para cada protocolo al que el servidor dé soporte, especificando cada vez un nombre de nodos exclusivo.

Si se especifica cualquier parámetro de configuración de protocolo al registrar un servidor DB2 localmente, alterará temporalmente el valor especificado en el archivo de configuración del gestor de bases de datos.

Sólo se puede registrar un servidor DB2 remoto en LDAP. Se debe especificar el nombre de sistema y el nombre de instancia del servidor remoto, junto con la información de protocolo para el servidor remoto.

Cuando se registra un servidor de bases de datos de sistema principal, se debe especificar SQLF_NT_DCS como valor del parámetro iNodeType.

db2LdapUncatalogDatabase - Desregistrar base de datos del servidor LDAP

Elimina una entrada de base de datos en LDAP (Lightweight Directory Access Protocol).

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2LdapUncatalogDatabase (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LdapUncatalogDatabaseStruct
{
   char *piAlias;
   char *piBindDN;
   char *piPassword;
} db2LdapUncatalogDatabaseStruct;
```

Parámetros de la API db2LdapUncatalogDatabase

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParamStruct.

pParamStruct
Entrada. Puntero a la estructura db2LdapUncatalogDatabaseStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2LdapUncatalogDatabaseStruct

piAlias
Entrada. Especifique un alias para la entrada de base de datos. Es parámetro es obligatorio.

piBindDN
Entrada. Especifique el nombre distinguido (DN) LDAP del usuario. El DN LDAP de usuario debe tener suficiente autorización para suprimir el objeto del directorio LDAP. Si no se especifica el nombre DN LDAP del usuario, se utilizarán las credenciales del usuario de la conexión actual.

piPassword
Entrada. Contraseña de cuenta.

---

db2LdapUncatalogNode - Suprimir alias para nombre de nodo del servidor LDAP

Elimina una entrada de nodo de LDAP (Lightweight Directory Access Protocol).

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2LdapUncatalogNode (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LdapUncatalogNodeStruct
{
   char *piAlias;
   char *piBindDN;
   char *piPassword;
} db2LdapUncatalogNodeStruct;
```

Parámetros de la API db2LdapUncatalogNode

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParamStruct
Entrada. Puntero a la estructura db2LdapUncatalogNodeStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2LdapUncatalogNodeStruct

piAlias
Entrada. Especifique el alias del nodo que se debe descatalogar de LDAP.

piBindDN
Entrada. Especifique el nombre distinguido (DN) LDAP del usuario. El DN LDAP de usuario debe tener suficiente autorización para suprimir el objeto del directorio LDAP. Si no se especifica el nombre DN LDAP del usuario, se utilizarán las credenciales del usuario de la conexión actual.

piPassword
Entrada. Contraseña de cuenta.

db2LdapUpdate - Actualizar los atributos del servidor DB2 en el servidor LDAP

Actualiza la información de protocolo de comunicación para el servidor DB2 en LDAP (Lightweight Directory Access Protocol).

Nota: NetBIOS ya no está soportado. SNA, incluyendo sus API APPC, APPN y CPI-C, tampoco está soportado. Si utiliza estos protocolos, debe volver a catalogar los nodos y bases de datos utilizando un protocolo soportado como, por ejemplo, TCP/IP. Las referencias a estos protocolos se deben pasar por alto.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2LdapUpdate (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LdapUpdateStruct
{
   char *piNodeName;
   char *piComment;
   unsigned short                      iNodeType;
   db2LdapProtocolInfo iProtocol;
   char *piBindDN;
   char *piPassword;
} db2LdapUpdateStruct;
```

```
typedef SQL_STRUCTURE db2LdapProtocolInfo
{
   char iType;
   char *piHostName;
   char *piServiceName;
   char *piNetbiosName;
   char *piNetworkId;
   char *piPartnerLU;
   char *piTPName;
   char *piMode;
   unsigned short                       iSecurityType;
   char *piLanAdapterAddress;
   char *piChangePasswordLU;
   char *piIpxAddress;
} db2LdapProtocolInfo;
```

Parámetros de la API db2LdapUpdate

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParamStruct.

pParamStruct
Entrada. Puntero a la estructura db2LdapUpdateStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2LdapUpdateStruct

piNodeName
Entrada. Especifique el nombre de nodo que represente al servidor DB2 en LDAP.

piComment
Entrada. Especifique una nueva descripción para el servidor DB2. La longitud máxima es de 30 caracteres. No se permite ningún retorno de carro o carácter de salto de línea.

iNodeType
Entrada. Especifique un tipo de nodo nuevo. Los valores válidos son:

- SQLF_NT_SERVER
- SQLF_NT_MPP
- SQLF_NT_DCS
- SQL_PARM_UNCHANGE

iProtocol
Entrada. Especifique la información de protocolo actualizada en la estructura db2LdapProtocolInfo.

piBindDN
Entrada. Especifique el nombre distinguido (DN) LDAP del usuario. El DN LDAP de usuario debe tener suficiente autorización para crear y actualizar el objeto del directorio LDAP. Si no se especifica el nombre DN LDAP del usuario, se utilizarán las credenciales del usuario de la conexión actual.

piPassword
Entrada. Contraseña de cuenta.

Parámetros de la estructura de datos db2LdapProtocolInfo

iType Entrada. Especifique el tipo de protocolo que soporta el servidor. Si el

servidor es compatible con más de un protocolo, es necesario realizar varios procesos de registro (cada uno con un nombre de nodo y tipo de protocolo diferente). Los valores válidos son:

SQL_PROTOCOL_TCPIP
Para el soporte de TCP/IPv4 o TCP/IPv6

SQL_PROTOCOL_TCPIP4
Para el soporte de TCP/IPv4

SQL_PROTOCOL_TCPIP6
Para el soporte de TCP/IPv6

SQL_PROTOCOL_SOCKS
Para TCP/IP con SOCKS de seguridad

SQL_PROTOCOL_SOCKS4
Para TCP/IPv4 con SOCKS de seguridad

SQL_PROTOCOL_NPIPE
Para el soporte Conexiones con nombre de Windows

piHostName
Entrada. Especifique el nombre de sistema principal TCP/IP o la dirección IP. La dirección IP puede ser una dirección IPv4 o IPv6. Si se especifica una dirección IP, debe coincidir con el tipo de protocolo seleccionado. Por ejemplo, si SQL_PROTOCOL_TCPIP4 está seleccionado, la dirección IP especificada debe ser una dirección IPv4.

piServiceName
Entrada. Especifique el nombre de servicio o el número de puerto TCP/IP.

piNetbiosName
Entrada. Especifique el nombre de estación de trabajo de NetBIOS. El nombre de NetBIOS debe especificarse para el soporte de NetBIOS.

piNetworkID
Entrada. Especifique el ID de red. El ID de red se debe especificar para el soporte de APPC/APPN.

piPartnerLU
Entrada. Especifique el nombre de LU asociada para la máquina servidor DB2. La LU asociada debe especificarse para el soporte APPC/APPN.

piTPName
Entrada. Especifique el nombre de programa de transacción. El nombre de programa de transacción debe especificarse para el soporte de APPC/APPN.

piMode
Entrada. Especifique el nombre de modalidad. La modalidad debe especificarse para el soporte de APPC/APPN.

iSecurityType
Entrada. Especifique el nivel de seguridad de APPC. Los valores válidos son:

- SQL_CPIC_SECURITY_NONE (valor por omisión)
- SQL_CPIC_SECURITY_SAME
- SQL_CPIC_SECURITY_PROGRAM

piLanAdapterAddress
Entrada. Especifique la dirección del adaptador de red. Este parámetro sólo es necesario para el soporte de APPC. Para APPN, se puede establecer en NULL.

piChangePasswordLU
Entrada. Especifique el nombre de la LU asociada que se debe utilizar al cambiar la contraseña para el servidor de bases de datos de sistema principal.

piIpxAddress
Entrada. Especifique la dirección IPX completa. La dirección IPX se debe especificar para el soporte de IPX/SPX.

---

db2LdapUpdateAlternateServerForDB - Actualizar el servidor alternativo de la base de datos en el servidor LDAP

Actualiza el servidor alternativo para una base de datos en LDAP (Lightweight Directory Access Protocol).

Autorización

Acceso de lectura/grabación al servidor LDAP.

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2LdapUpdateAlternateServerForDB (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LdapUpdateAltServerStruct
{
   char *piDbAlias;
   char *piNode;
   char *piGWNode;
   char *piBindDN;
   char *piPassword;
} db2LdapUpdateAltServerStruct;
```

Parámetros de la API db2LdapUpdateAlternateServerForDB

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2LdapUpdateAltServerStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2LdapUpdateAltServerStruct

piDbAlias
Entrada. Serie que contiene un alias para la base de datos que se debe actualizar.

piNode
Entrada. Serie que contiene el nombre de nodo alternativo. Este nombre de nodo debe existir en LDAP.

piGWNode
Entrada. Serie que contiene el nombre de nodo de pasarela alternativo. Este nombre de nodo debe existir en LDAP. El cliente de ejecución de IBM Data Server utiliza este parámetro para conectar con el sistema principal a través de la pasarela.

piBindDN
Entrada. Especifica el nombre distinguido (DN) LDAP del usuario. El DN de LDAP del usuario debe dispone de suficiente autorización para crear y actualizar objetos en el directorio LDAP. Si no se especifica el DN de LDAP del usuario, se utilizarán las credenciales del usuario actual.

piPassword
Entrada. Contraseña de cuenta.

db2Load - Cargar datos en una tabla

Carga datos en una tabla DB2. Los datos que residen en el servidor pueden estar en forma de archivo, cursor, cinta o conexión con nombre. Los datos que residen en un cliente conectado remotamente pueden estar en forma archivo totalmente calificado, cursor o conexión con nombre. Aunque es más rápido que el programa de utilidad de importación, el programa de utilidad de carga no da soporte a la carga de datos a nivel de la jerarquía o la carga a un apodo.

Autorización

Una de las siguientes:

- *sysadm*
- *dbadm*
- autorización de carga en la base de datos y:
  - privilegio INSERT en la tabla cuando se invoca el programa de utilidad de carga en modalidad INSERT, en modalidad TERMINATE (para terminar una operación de inserción de carga anterior) o en modalidad RESTART (para reiniciar una operación de inserción de carga anterior)
  - privilegio INSERT y DELETE en la tabla cuando el programa de utilidad de carga se invoca en modalidad REPLACE, en modalidad TERMINATE (para terminar una operación de sustitución de carga anterior) o en modalidad RESTART (para reiniciar una operación de sustitución de carga anterior)
  - Privilegio INSERT en la tabla de excepción, si dicha tabla se utiliza como parte de la operación de carga.

Nota: En general, todos los procesos de carga y todos los procesos de servidor DB2 son propiedad del propietario de la instancia. Todos estos procesos utilizan la identificación del propietario de la instancia para acceder a los archivos necesarios.

Por consiguiente, el propietario de instancia debe tener acceso de lectura para los archivos de entrada, independientemente de quién invoque el mandato.

Conexión necesaria

Base de datos. Si se ha habilitado la conexión implícita, se establece una conexión con la base de datos por omisión. El acceso del programa de utilidad a servidores de bases de datos Linux, UNIX o Windows desde clientes Linux, UNIX o Windows debe ser una conexión directa a través del motor y no a través de un entorno de pasarela o de bucle de retorno de DB2 Connect.

Instancia. No se necesita una conexión explícita. Si se ha establecido una conexión a la base de datos, se intenta una conexión implícita a la instancia local.

Archivo de inclusión de la API

db2ApiDf.h

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2Load (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LoadStruct
{
   struct sqlu_media_list *piSourceList;
   struct sqlu_media_list *piLobPathList;
   struct sqldcol *piDataDescriptor;
   struct sqlchar *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piLocalMsgFileName;
   char *piTempFilesPath;
   struct sqlu_media_list *piVendorSortWorkPaths;
   struct sqlu_media_list *piCopyTargetList;
   db2int32 *piNullIndicators;
   struct db2LoadIn *piLoadInfoIn;
   struct db2LoadOut *poLoadInfoOut;
   struct db2PartLoadIn *piPartLoadInfoIn;
   struct db2PartLoadOut *poPartLoadInfoOut;
   db2int16 iCallerAction;
   struct sqlu_media_list *piXmlPathList;
   struct sqllob *piLongActionString;
} db2LoadStruct;

typedef SQL_STRUCTURE db2LoadUserExit
{
   db2Char iSourceUserExitCmd;
   struct db2Char *piInputStream;
   struct db2Char *piInputFileName;
   struct db2Char *piOutputFileName;
   db2Uint16 *piEnableParallelism;
} db2LoadUserExit;

typedef SQL_STRUCTURE db2LoadIn
{
   db2Uint64 iRowcount;
   db2Uint64 iRestartcount;
   char *piUseTablespace;
   db2Uint32 iSavecount;
   db2Uint32 iDataBufferSize;
   db2Uint32 iSortBufferSize;
```

```
   db2Uint32 iWarningcount;
   db2Uint16 iHoldQuiesce;
   db2Uint16 iCpuParallelism;
   db2Uint16 iDiskParallelism;
   db2Uint16 iNonrecoverable;
   db2Uint16 iIndexingMode;
   db2Uint16 iAccessLevel;
   db2Uint16 iLockWithForce;
   db2Uint16 iCheckPending;
   char iRestartphase;
   char iStatsOpt;
   db2Uint16 *piXmlParse;
   db2DMUXmlValidate *piXmlValidate;
   db2Uint16 iSetIntegrityPending;
   struct db2LoadUserExit *piSourceUserExit;
} db2LoadIn;

typedef SQL_STRUCTURE db2LoadOut
{
   db2Uint64 oRowsRead;
   db2Uint64 oRowsSkipped;
   db2Uint64 oRowsLoaded;
   db2Uint64 oRowsRejected;
   db2Uint64 oRowsDeleted;
   db2Uint64 oRowsCommitted;
} db2LoadOut;

typedef SQL_STRUCTURE db2PartLoadIn
{
   char *piHostname;
   char *piFileTransferCmd;
   char *piPartFileLocation;
   struct db2LoadNodeList *piOutputNodes;
   struct db2LoadNodeList *piPartitioningNodes;
   db2Uint16 *piMode;
   db2Uint16 *piMaxNumPartAgents;
   db2Uint16 *piIsolatePartErrs;
   db2Uint16 *piStatusInterval;
   struct db2LoadPortRange *piPortRange;
   db2Uint16 *piCheckTruncation;
   char *piMapFileInput;
   char *piMapFileOutput;
   db2Uint16 *piTrace;
   db2Uint16 *piNewline;
   char *piDistfile;
   db2Uint16 *piOmitHeader;
   SQL_PDB_NODE_TYPE *piRunStatDBPartNum;
} db2PartLoadIn;

typedef SQL_STRUCTURE db2LoadNodeList
{
   SQL_PDB_NODE_TYPE *piNodeList;
   db2Uint16 iNumNodes;
} db2LoadNodeList;

typedef SQL_STRUCTURE db2LoadPortRange
{
   db2Uint16 iPortMin;
   db2Uint16 iPortMax;
} db2LoadPortRange;

typedef SQL_STRUCTURE db2PartLoadOut
{
   db2Uint64 oRowsRdPartAgents;
   db2Uint64 oRowsRejPartAgents;
   db2Uint64 oRowsPartitioned;
   struct db2LoadAgentInfo *poAgentInfoList;
```

```
   db2Uint32 iMaxAgentInfoEntries;
   db2Uint32 oNumAgentInfoEntries;
} db2PartLoadOut;

typedef SQL_STRUCTURE db2LoadAgentInfo
{
   db2int32 oSqlcode;
   db2Uint32 oTableState;
   SQL_PDB_NODE_TYPE oNodeNum;
   db2Uint16 oAgentType;
} db2LoadAgentInfo;

SQL_API_RC SQL_API_FN
  db2gLoad (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gLoadStruct
{
   struct sqlu_media_list *piSourceList;
   struct sqlu_media_list *piLobPathList;
   struct sqldcol *piDataDescriptor;
   struct sqlchar *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piLocalMsgFileName;
   char *piTempFilesPath;
   struct sqlu_media_list *piVendorSortWorkPaths;
   struct sqlu_media_list *piCopyTargetList;
   db2int32 *piNullIndicators;
   struct db2gLoadIn *piLoadInfoIn;
   struct db2LoadOut *poLoadInfoOut;
   struct db2gPartLoadIn *piPartLoadInfoIn;
   struct db2PartLoadOut *poPartLoadInfoOut;
   db2int16 iCallerAction;
   db2Uint16 iFileTypeLen;
   db2Uint16 iLocalMsgFileLen;
   db2Uint16 iTempFilesPathLen;
   struct sqlu_media_list *piXmlPathList;
   struct sqllob *piLongActionString;
} db2gLoadStruct;

typedef SQL_STRUCTURE db2gLoadIn
{
   db2Uint64 iRowcount;
   db2Uint64 iRestartcount;
   char *piUseTablespace;
   db2Uint32 iSavecount;
   db2Uint32 iDataBufferSize;
   db2Uint32 iSortBufferSize;
   db2Uint32 iWarningcount;
   db2Uint16 iHoldQuiesce;
   db2Uint16 iCpuParallelism;
   db2Uint16 iDiskParallelism;
   db2Uint16 iNonrecoverable;
   db2Uint16 iIndexingMode;
   db2Uint16 iAccessLevel;
   db2Uint16 iLockWithForce;
   db2Uint16 iCheckPending;
   char iRestartphase;
   char iStatsOpt;
   db2Uint16 iUseTablespaceLen;
   db2Uint16 iSetIntegrityPending;
   db2Uint16 *piXmlParse;
   db2DMUXmlValidate *piXmlValidate;
   struct db2LoadUserExit *piSourceUserExit;
```

```
} db2gLoadIn;

typedef SQL_STRUCTURE db2gPartLoadIn
{
   char *piHostname;
   char *piFileTransferCmd;
   char *piPartFileLocation;
   struct db2LoadNodeList *piOutputNodes;
   struct db2LoadNodeList *piPartitioningNodes;
   db2Uint16 *piMode;
   db2Uint16 *piMaxNumPartAgents;
   db2Uint16 *piIsolatePartErrs;
   db2Uint16 *piStatusInterval;
   struct db2LoadPortRange *piPortRange;
   db2Uint16 *piCheckTruncation;
   char *piMapFileInput;
   char *piMapFileOutput;
   db2Uint16 *piTrace;
   db2Uint16 *piNewline;
   char *piDistfile;
   db2Uint16 *piOmitHeader;
   void *piReserved1;
   db2Uint16 iHostnameLen;
   db2Uint16 iFileTransferLen;
   db2Uint16 iPartFileLocLen;
   db2Uint16 iMapFileInputLen;
   db2Uint16 iMapFileOutputLen;
   db2Uint16 iDistfileLen;
} db2gPartLoadIn;


/* Definitions for iUsing value of db2DMUXmlValidate structure               */
#define DB2DMU_XMLVAL_XDS              1    /* Usa XDS                       */
#define DB2DMU_XMLVAL_SCHEMA           2    /* Usa un esquema especificado   */
#define DB2DMU_XMLVAL_SCHEMALOC_HINTS  3    /* Usa sugerencias schemaLocation */
#define DB2DMU_XMLVAL_ORIGSCHEMA       4    /* Usa esquema con el que se ha
                                               validado el documento original-
                                               mente (carga desde el cursor sólo)*/
```

Parámetros de la API db2Load

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2LoadStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2LoadStruct

piSourceList
Entrada. Puntero a una estructura sqlu_media_list utilizado para proporcionar una lista de archivos fuente, dispositivos, proveedores, áreas de interconexión de memoria o sentencias SQL.

La información proporcionada en esta estructura depende del valor del campo media_type. Los valores válidos (definidos en el archivo de cabecera sqlutil del directorio de inclusión) son:

SQLU_SQL_STMT
Si el campo media_type tiene establecido este valor, el llamador proporciona una consulta SQL mediante el campo pStatement del

campo de destino. El campo pStatement es del tipo sqlu_statement_entry. El campo de sesiones debe establecerse en el valor 1, ya que el programa de utilidad de carga solamente acepta una consulta SQL por carga.

SQLU_SERVER_LOCATION
Si el campo media_type tiene establecido este valor, el llamador proporciona información mediante estructuras sqlu_location_entry. El campo de sesiones indica el número de estructuras sqlu_location_entry proporcionadas. Se utiliza para archivos, dispositivos y conexiones con nombre.

SQLU_CLIENT_LOCATION
Si el campo media_type tiene establecido este valor, el llamador proporciona información mediante estructuras sqlu_location_entry. El campo de sesiones indica el número de estructuras sqlu_location_entry proporcionadas. Se utiliza para archivos y conexiones con nombre totalmente calificados. Tenga en cuenta que este media_type solamente es válido si se llama a la API mediante un cliente conectado remotamente.

SQLU_TSM_MEDIA
Si el campo media_type tiene establecido este valor, se utiliza la estructura sqlu_vendor, donde el nombre de archivo es el identificador exclusivo para los datos a cargar. Solamente deberá haber una entrada sqlu_vendor, independientemente del valor de las sesiones. El campo de sesiones indica el número de sesiones TSM a iniciar. El programa de utilidad de carga iniciará las sesiones con distintos números de secuencia, pero con los mismos datos en la única entrada sqlu_vendor.

SQLU_OTHER_MEDIA
Si el campo media_type tiene establecido este valor, se utiliza la estructura sqlu_vendor, donde shr_lib es el nombre de biblioteca compartida y el nombre de archivo es el identificador exclusivo para los datos a cargar. Solamente deberá haber una entrada sqlu_vendor, independientemente del valor de las sesiones. El campo de sesiones indica el número de otras sesiones de proveedor a iniciar. El programa de utilidad de carga iniciará las sesiones con distintos números de secuencia, pero con los mismos datos en la única entrada sqlu_vendor.

SQLU_REMOTEFETCH
Si el campo media_type tiene establecido este valor, el llamador proporciona información mediante una estructura sqlu_remotefetch_entry. El campo de sesiones debe establecerse en el valor 1.

piLobPathList
Entrada. Puntero a una estructura sqlu_media_list. Para los tipos de archivo IXF, ASC y DEL, una lista de víasde acceso o dispositivos totalmente calificados para identificar la ubicación de los archivos LOB individuales que deben cargarse. Los nombres de archivo se encuentran en los archivos IXF, ASC o DEL y se añaden a las vías de acceso proporcionadas.

La información proporcionada en esta estructura depende del valor del campo media_type. Los valores válidos (definidos en el archivo de cabecera sqlutil del directorio de inclusión) son:

SQLU_LOCAL_MEDIA
Si tiene establecido este valor, el llamador proporciona información mediante estructuras sqlu_media_entry. El campo de sesiones indica el número de estructuras sqlu_media_entry proporcionadas.

SQLU_TSM_MEDIA
Si tiene establecido este valor, se utiliza la estructura sqlu_vendor, donde el nombre de archivo es el identificador exclusivo para los datos a cargar. Solamente deberá haber una entrada sqlu_vendor, independientemente del valor de las sesiones. El campo de sesiones indica el número de sesiones TSM a iniciar. El programa de utilidad de carga iniciará las sesiones con distintos números de secuencia, pero con los mismos datos en la única entrada sqlu_vendor.

SQLU_OTHER_MEDIA
Si tiene establecido este valor, se utiliza la estructura sqlu_vendor, donde shr_lib es el nombre de biblioteca compartida y el nombre de archivo es el identificador exclusivo para los datos a cargar. Solamente deberá haber una entrada sqlu_vendor, independientemente del valor de las sesiones. El campo de sesiones indica el número de otras sesiones de proveedor a iniciar. El programa de utilidad de carga iniciará las sesiones con distintos números de secuencia, pero con los mismos datos en la única entrada sqlu_vendor.

piDataDescriptor
Entrada. Puntero a una estructura sqldcol que contiene información sobre las columnas que se seleccionan para cargar desde el archivo externo.

Si el parámetro piFileType se ha establecido en SQL_ASC, el campo dcolmeth de esta estructura debe establecerse en SQL_METH_L. El usuario especifica las ubicaciones inicial y final para cada columna que se debe cargar.

Si el tipo de archivo es SQL_DEL, dcolmeth puede ser SQL_METH_P o SQL_METH_D. Si es SQL_METH_P, el usuario debe proporcionar la posición de columna origen. Si es SQL_METH_D, la primera columna del archivo se carga en la primera columna de la tabla, y así sucesivamente.

Si el tipo de archivo es SQL_IXF, dcolmeth puede ser SQL_METH_P, SQL_METH_D o SQL_METH_N. Son aplicables las reglas para archivos DEL, excepto en que SQL_METH_N indica que los nombres de columna de archivo deben proporcionarse en la estructura sqldcol.

piActionString
En desuso, sustituido por piLongActionString.

piLongActionString
Entrada. Puntero a una estructura *sqllob* que contiene un campo de 4 bytes de longitud, seguido por una matriz de caracteres que especifica una acción que afecta a la tabla.

La matriz de caracteres tiene este formato:

```
"INSERT|REPLACE KEEPDICTIONARY|REPLACE RESETDICTIONARY|RESTART|TERMINATE
INTO tbname [(column_list)]
[FOR EXCEPTION e_tbname]"
```

INSERT
Añade los datos cargados a la tabla sin cambiar los datos de tabla existentes.

REPLACE
Suprime todos los datos existentes de la tabla e inserta los datos cargados. La definición de tabla y las definiciones de índice no se modifican.

RESTART
Reinicia una operación de carga interrumpida anteriormente. La operación de carga continuará automáticamente desde el último punto de coherencia de la fase de carga, creación o supresión.

TERMINATE
Termina una operación de carga interrumpida anteriormente y retrotrae la operación hasta el momento en que se empezó, incluso si se habían pasado puntos de coherencia. Los estados de los espacios de tablas implicados en la operación vuelven a ser normales y todos los objetos de tabla pasan a estar coherentes (puede que los objetos de índice se marquen como no válidos, en cuyo caso tendrá lugar automáticamente una reconstrucción de índice en el siguiente acceso). Si los espacios de tablas en los que reside la tabla no están en estado de pendiente de carga, esta opción no afecta al estado de los espacios de tabla.

La opción de terminación de carga no eliminará un estado de pendiente de copia de seguridad de los espacios de tabla.

tbname
El nombre de la tabla en la que deben cargarse los datos. La tabla no puede ser una tabla de sistema o una tabla declarada temporal. Se puede especificar un alias o el nombre de tabla totalmente calificado o no calificado. Un nombre de tabla calificado tiene el formato esquema.nombretabla. Si se especifica un nombre de tabla no calificado, la tabla se calificará con CURRENT SCHEMA.

(column_list)
Una lista de nombres de columnas de tabla en la que deben insertarse los datos. Los nombres de columnas deben estar separados por comas. Si un nombre contiene espacios o caracteres en minúsculas, debe ir entre comillas.

FOR EXCEPTION e_tbname
Especifica la tabla de excepción en la que se copiarán las filas erróneas. La tabla de excepción se utiliza para almacenar copias de filas que violan reglas de índice, restricciones de rango y políticas de seguridad.

NORANGEEXC
Indica que si se rechaza una fila debido a una violación del rango, no se insertará en la tabla de excepción.

NOUNIQUEEXC
Indica que si se rechaza una fila porque viola una restricción exclusiva, no se insertará en la tabla de excepción.

piFileType
Entrada. Serie que indica el formato de la fuente de datos de entrada. Los formatos externos soportados (definidos en sqlutil) son:

SQL_ASC
ASCII no delimitado.

SQL_DEL
ASCII delimitado, para intercambio con dBase, BASIC y los programas IBM Personal Decision Series, y muchos otros gestores de bases de datos y gestores de archivos.

SQL_IXF
Versión de PC del formato de intercambio integrado, el método favorito de exportar datos de una tabla para poder cargarlos más tarde en la misma tabla o en la tabla de otro gestor de bases de datos.

SQL_CURSOR
Una consulta SQL. La estructura sqlu_media_list transferida mediante el parámetro piSourceList es de tipo SQLU_SQL_STMT o SQLU_REMOTEFETCH, y hace referencia a una consulta SQL o a un nombre de tabla.

piFileTypeMod
Entrada. Puntero a la estructura sqlchar, seguido de una matriz de caracteres que especifican una o varias opciones de proceso. Si este puntero es NULL, o si la estructura a la que apunta tiene cero caracteres, esta acción se interpreta como selección de una especificación por omisión.

No todas las opciones pueden utilizarse con todos los tipos de archivos soportados. Consulte el enlace relacionado ″Modificadores de tipo de archivo para el programa de utilidad de carga.″

piLocalMsgFileName
Entrada. Serie que contiene el nombre de un archivo local en el que deben escribirse los mensajes de salida.

piTempFilesPath
Entrada. Serie que contiene el nombre de vía de acceso que debe utilizarse en el servidor para archivos temporales. Se crean archivos temporales para almacenar mensajes, puntos de coherencia e información de fase de supresión.

piVendorSortWorkPaths
Entrada. Puntero a la estructura sqlu_media_list que especifica los directorios de trabajo de Clasificación de proveedores.

piCopyTargetList
Entrada. Punterio que apunta a una estructura sqlu_media_list utilizada (si debe crearse una imagen de copia) para proporcionar una lista de vías de acceso destino, dispositivos o una biblioteca compartida en la que debe grabarse la imagen de copia.

Los valores proporcionados en esta estructura dependen del valor del campo media_type. Los valores válidos para este parámetro (definidos en el archivo de cabecera sqlutil del directorio de inclusión) son:

SQLU_LOCAL_MEDIA
Si la copia debe grabarse en soporte local, establezca media_type en este valor y proporcione información sobre los objetivos de las estructuras sqlu_media_entry. El campo de sesiones especifica el número de estructuras sqlu_media_entry proporcionadas.

SQLU_TSM_MEDIA
Si la copia debe grabarse en TSM, utilice este valor. No se necesita información adicional.

SQLU_OTHER_MEDIA
Si debe utilizarse un producto de proveedor, utilice este valor y proporcione información adicional mediante una estructura sqlu_vendor. Establezca el campo shr_lib de esta estructura en el nombre de biblioteca compartida del producto de proveedor. Proporcione solamente una entrada sqlu_vendor, independientemente del valor de las sesiones. El campo de sesiones especifica el número de estructuras sqlu_media_entry proporcionadas. El programa de utilidad de carga iniciará las sesiones con distintos números de secuencia, pero con los mismos datos proporcionados en la única entrada sqlu_vendor.

piNullIndicators
Entrada. Sólo para archivos ASC. Matriz de enteros que indica si los datos de la columna pueden contener valores nulos o no. Hay una correspondencia biunívoca entre los elementos de esta matriz y las columnas que se cargan desde el archivo de datos. Es decir, el número de elementos debe ser igual al campo dcolnum del parámetro piDataDescriptor. Cada elemento de la matriz contiene un número que identifica una ubicación del archivo de datos que debe utilizarse como campo de indicador NULL, o un cero que indica que la columna de tabla no puede ser nula. Si el elemento no es cero, la ubicación identificada del archivo de datos debe contener una Y o una N. Una Y indica que los datos de la columna de tabla son NULL y N indica que los datos de la columna de tabla no son NULL.

piLoadInfoIn
Entrada. Puntero a la estructura db2LoadIn.

poLoadInfoOut
Salida. Puntero a la estructura db2LoadOut.

piPartLoadInfoIn
Entrada. Puntero a la estructura db2PartLoadIn.

poPartLoadInfoOut
Salida. Puntero a la estructura db2PartLoadOut.

iCallerAction
Entrada. Acción solicitada por el llamador. Los valores válidos (definidos en el archivo de cabecera sqlutil del directorio de inclusión) son:

SQLU_INITIAL
Llamada inicial. Este valor (o SQLU_NOINTERRUPT) debe utilizarse en la primera llamada a la API.

SQLU_NOINTERRUPT
Llamada inicial. No suspender el proceso. Este valor (o SQLU_INITIAL) debe utilizarse en la primera llamada a la API.

Si se devuelve la llamada inicial o cualquier llamada siguiente y requiere que la aplicación llamador realice alguna acción antes de completar la operación de carga solicitada, la acción del llamador debe establecerse en uno de los siguientes valores:

SQLU_CONTINUE
Continúa el proceso. Este valor sólo puede utilizarse en llamadas posteriores a la API, después de que la llamada inicial haya devuelto el control al programa de utilidad que solicitaba una entrada del usuario (por ejemplo, para responder a una condición de fin de cinta). Especifica que la acción del usuario solicitada por

el programa de utilidad se ha completado y el programa de utilidad puede continuar procesando la petición inicial.

SQLU_TERMINATE
Termina el proceso. Provoca que el programa de utilidad de carga abandone de forma prematura, dejando los espacios de tablas que se están cargando en estado LOAD_PENDING. Esta opción deberá especificarse si no debe realizarse más proceso de los datos.

SQLU_ABORT
Termina el proceso. Provoca que el programa de utilidad de carga abandone de forma prematura, dejando los espacios de tablas que se están cargando en estado LOAD_PENDING. Esta opción deberá especificarse si no debe realizarse más proceso de los datos.

SQLU_RESTART
Reiniciar el proceso.

SQLU_DEVICE_TERMINATE
Interrumpir un solo dispositivo. Esta opción deberá especificarse si el programa de utilidad debe dejar de leer datos del dispositivo, pero debe realizarse más proceso de los datos.

piXmlPathList
Entrada. Puntero a sqlu_media_list cuyo campo media_type se ha establecido en SQLU_LOCAL_MEDIA y cuya estructura sqlu_media_entry lista vías de acceso del cliente donde se encuentran los archivos xml.

Parámetros de la estructura de datos db2LoadUserExit

iSourceUserExitCmd
Entrada. El nombre totalmente calificado de un ejecutable que se utilizará para ofrecer datos al programa de utilidad. Por motivos de seguridad, el ejecutable debe colocarse dentro del directorio `sqllib/bin` en el servidor. Este parámetro es obligatorio si la estructura piSourceUserExit no es NULL.

Los campos piInputStream, piInputFileName, piOutputFileName y piEnableParallelism son opcionales.

piInputStream
Entrada. Una corriente de bytes genérica que se pasará directamente a la aplicación de salida de usuario mediante STDIN. Tiene control completo sobre qué datos se incluyen en esta corriente de bytes y en qué formato. El programa de utilidad de carga simplemente llevará esta corriente de bytes al servidor y la pasará a la aplicación de salida de usuario alimentando el STDIN del proceso (no convertirá la página de códigos ni modificará la corriente de bytes). La aplicación de salida de usuario leerá los argumentos de STDIN y utilizará los datos como esté planeado.

Un atributo importante de esta característica es la capacidad de ocultar la información confidencial (tal como ID de usuario y contraseñas).

piInputFileName
Entrada. Contiene el nombre de un archivo del extremo cliente totalmente calificado, cuyo contenido se pasará a la aplicación de salida de usuario alimentando el STDIN del proceso.

piOutputFileName
Entrada. El nombre totalmente calificado de un archivo del lado del servidor. Las corrientes STDOUT y STDERR del proceso que está

ejecutando la aplicación de salida de usuario se transmitirán en modalidad continua a este archivo. Cuando piEnableParallelism es TRUE, se crean varios archivos (uno por instancia de salida de usuario) y se añade a cada nombre de archivo un valor de número de nodo numérico de 3 dígitos, por ejemplo <nombrearchivo>.000).

piEnableParallelism
Entrada. Distintivo que indica que el programa de utilidad deberá intentar paralelizar la invocación de la aplicación de salida de usuario.

Parámetros de la estructura de datos db2LoadIn

iRowcount
Entrada. Número de registros físicos a cargar. Permite a un usuario cargar solamente las primeras filas rowcnt de un archivo.

iRestartcount
Entrada. Reservado para una utilización futura.

piUseTablespace
Entrada. Si se están reconstruyendo los índices, se construye una copia duplicada del índice en el espacio de tablas iUseTablespaceName y se copia al espacio de tablas original al final de la carga. Con esta opción sólo se pueden utilizar espacios de tablas temporales del sistema. Si entonces no se especifica, el índice duplicado se creará en el mismo espacio de tablas que el objeto de índice.

Si la copia duplicada se crea en el espacio de tablas como objeto de índice, la copia del objeto de índice duplicado sobre el antiguo objeto de índice es instantánea. Si la copia duplicada está en un espacio de tablas diferente del objeto de índice, se realiza una copia física. Esto podría implicar una E/S y tiempo considerables. La copia se produce mientras la tabla está fuera de línea al final de una carga.

Este campo se ignora si iAccessLevel es SQLU_ALLOW_NO_ACCESS.

Esta opción se pasa por alto si el usuario no especifica INDEXING MODE REBUILD o INDEXING MODE AUTOSELECT. Esta opción también se pasará por alto si se elige INDEXING MODE AUTOSELECT y la carga elige actualizar el índice de forma incremental.

iSavecount
Número de registros a cargar antes de establecer un punto de coherencia. Este valor se convierte en una cuenta de páginas y se redondea por exceso a los intervalos del tamaño de extensión. Dado que se emite un mensaje en cada punto de coherencia, se deberá seleccionar esta opción si la operación de carga se supervisará utilizando db2LoadQuery - Cargar consulta. Si el valor de savecount no es suficientemente alto, la sincronización de las actividades realizadas en cada punto de coherencia influirá en el rendimiento.

El valor por omisión es 0, que significa que no se establecerán puntos de coherencia, a menos que sean necesarios.

iDataBufferSize
El número de páginas de 4 KB (independientemente del grado de paralelismo) que se deben utilizar como espacio de almacenamiento intermedio para transferir datos dentro del programa de utilidad. Si el valor especificado es menor que el mínimo algorítmico, se utilizará el mínimo necesario y no se devolverá ningún aviso.

Esta memoria se asigna directamente desde la pila del programa de utilidad, cuyo tamaño puede modificarse mediante el parámetro de configuración de base de datos util_heap_sz.

Si no se especifica un valor, el programa de utilidad calcula un valor por omisión inteligente en la ejecución. El valor por omisión se basa en un porcentaje del espacio libre disponible en la pila del programa de utilidad en el tiempo de creación de instancias del cargador, así como en algunas características de la tabla.

iSortBufferSize
Entrada. Esta opción especifica un valor que prevalece sobre el parámetro de configuración de base de datos SORTHEAP durante una operación de carga. Sólo es relevante al cargar tablas con índices y sólo cuando el parámetro iIndexingMode no se especifica como SQLU_INX_DEFERRED. El valor especificado no puede exceder el valor de SORTHEAP. Este parámetro es útil para acelerar la memoria de clasificación que LOAD utiliza sin cambiar el valor de SORTHEAP, que afectaría al proceso general de consulta.

iWarningcount
Entrada. Detiene la operación de carga después de warningcnt avisos. Establezca este parámetro si no se esperan avisos, pero se desea que se verifique si se están utilizando el archivo y la tabla correctos. Si el archivo de carga o la tabla de destino se especifican de modo incorrecto, el programa de utilidad de carga generará un aviso por cada fila que intente cargar, lo que hará que la carga falle. Si warningcnt es 0 o no se especifica esta opción, la operación de carga continuará independientemente del número de avisos emitidos.

Si la operación de carga se detiene porque se ha sobrepasado el umbral de avisos, se puede iniciar otra operación de carga en modalidad RESTART. La operación de carga continuará automáticamente desde el último punto de coherencia. Alternativamente, se puede iniciar otra operación de carga en modalidad REPLACE, empezando al principio del archivo de entrada.

iHoldQuiesce
Entrada. Un distintivo cuyo valor se establece en TRUE si el programa de utilidad ha de dejar la tabla en estado exclusivo inmovilizado tras la carga, y se establece FALSE si no ha de dejarlo.

iCpuParallelism
Entrada. Número de procesos o hebras que el programa de utilidad de carga creará para analizar, convertir y formatear registros al crear objetos de tabla. Este parámetro está diseñado para aprovechar el paralelismo entre particiones. Es especialmente útil al cargar datos clasificados previamente, porque se conserva el orden de registro de los datos fuente. Si el valor de este parámetro es cero, el programa de utilidad de carga utiliza un valor inteligente por omisión en la ejecución. Nota: Si se utiliza este parámetro con tablas que contienen campos LOB o LONG VARCHAR, su valor se convierte en uno, independientemente del número de CPU de sistema o del valor especificado por el usuario.

iDiskParallelism
Entrada. Número de procesos o hebras que el programa de utilidad de carga creará para grabar datos en los contenedores de espacios de tablas. Si no se especifica un valor, el programa de utilidad selecciona un valor por omisión inteligente basándose en el número de contenedores de espacios de tablas y en las características de la tabla.

iNonrecoverable
Entrada. Establézcalo en SQLU_NON_RECOVERABLE_LOAD si la transacción de carga debe marcarse como no recuperable y no será posible recuperarla mediante una acción subsiguiente de recuperación en avance. El programa de utilidad de recuperación en avance saltará la transacción y marcará la tabla en la que se estaban cargando datos como ″no válida″. El programa de utilidad también ignorará las transacciones subsiguientes para la tabla. Una vez completado el avance, lo que único que puede hacerse con la tabla es descartarla. Con esta opción, los espacios de tablas no se ponen en estado de pendiente de copia de seguridad a continuación de la operación de carga y durante la operación de carga no se tiene que realizar una copia de los datos cargados. Establézcalo en SQLU_RECOVERABLE_LOAD si la transacción de carga debe marcarse como recuperable.

iIndexingMode
Entrada. Especifica la modalidad de indexado. Los valores válidos (definidos en el archivo de cabecera sqlutil del directorio de inclusión) son:

SQLU_INX_AUTOSELECT
LOAD elige entre las modalidades de indexado REBUILD e INCREMENTAL.

SQLU_INX_REBUILD
Reconstruir índices de tabla.

SQLU_INX_INCREMENTAL
Ampliar los índices existentes.

SQLU_INX_DEFERRED
No actualizar los índices de tabla.

iAccessLevel
Entrada. Especifica el nivel de acceso. Los valores válidos son:

SQLU_ALLOW_NO_ACCESS
Especifica que la carga bloquea la tabla de forma exclusiva.

SQLU_ALLOW_READ_ACCESS
Especifica que los datos originales de la tabla (la parte no delta) deberá ser visible para los lectores mientras la carga está en curso. Esta opción solamente es válida para las adiciones de carga, por ejemplo una inserción de carga, y se ignorará para la sustitución de carga.

iLockWithForce
Entrada. Distintivo booleano. Si se establece en TRUE, la carga forzará las demás aplicaciones como sea necesario para asegurar que obtiene bloqueos de tabla de inmediato. Esta opción requiere la misma autorización que el mandato FORCE APPLICATIONS (SYSADM o SYSCTRL).

Las cargas SQLU_ALLOW_NO_ACCESS podrían forzar las aplicaciones que están en conflicto al comienzo de la operación de carga. Al inicio de la carga, el programa de utilidad puede forzar las aplicaciones que intentan consultar o modificar la tabla.

Las cargas SQLU_ALLOW_READ_ACCESS podrían forzar las aplicaciones que están en conflicto al comienzo o al final de la operación de carga. Al inicio de la carga, el programa de utilidad de carga puede forzar las

aplicaciones que intentan modificar la tabla. Al final de la carga, el programa de utilidad de carga puede forzar las aplicaciones que intentan consultar o modificar la tabla.

iCheckPending
Este parámetro queda en desuso a partir de la Versión 9.1. En su lugar, utilice el parámetro iSetIntegrityPending.

iRestartphase
Entrada. Reservado. El valor válido es un caráctere de un espacio ' '.

iStatsOpt
Entrada. La granularidad de las estadísticas que se deben recopilar. Los valores válidos son:

SQLU_STATS_NONE
No hay estadísticas para reunir.

SQLU_STATS_USE_PROFILE
Se reunen estadísticas basándose en el perfil definido para la tabla actual. Este perfil debe crearse utilizando el mandato RUNSTATS. Si no existe un perfil para la tabla actual, se devuelve un aviso y no se reúnen estadísticas.

iSetIntegrityPending
Entrada. Especifica que se coloque la tabla en estado de Pendiente de establecer integridad. Si se especifica el valor SQLU_SI_PENDING_CASCADE_IMMEDIATE, el estado de Pendiente de establecer integridad se aplicará de inmediato a todas las tablas dependientes y descendentes. Si se especifica el valor SQLU_SI_PENDING_CASCADE_DEFERRED, la aplicación del estado Pendiente de establecer integridad a las tablas dependientes quedará diferidad hasta que se compruebe si ha habido violaciones de integridad en la tabla. SQLU_SI_PENDING_CASCADE_DEFERRED es el valor por omisión si no se especifica la opción.

piSourceUserExit
Entrada. Un puntero a la estructura db2LoadUserExit.

piXmlParse
Entrada. Tipo de análisis que debe realizarse para los documentos XML. Los valores válidos que se encuentran en el archivo de cabecera db2ApiDf del directorio de inclusión son:

DB2DMU_XMLPARSE_PRESERVE_WS
Los espacios en blanco deben conservarse.

DB2DMU_XMLPARSE_STRIP_WS
Los caracteres en blanco deben eliminarse.

piXmlValidate
Entrada. Puntero a la estructura db2DMUXmlValidate. Indica que debe llevarse a cabo la validación de esquemas XML para los documentos XML.

```
/* Validar estructura XML
typedef SQL_STRUCTURE db2DMUXmlValidate
{
   db2Uint16                       iUsing;      /* Lo que se debe utilizar  */
                                                /* para la validación       */
   struct db2DMUXmlValidateXds    *piXdsArgs;   /* Argumentos para          */
                                                /* XMLVALIDATE USING XDS    */
   struct db2DMUXmlValidateSchema *piSchemaArgs; /* Argumentos para         */
                                                /* XMLVALIDATE USING SCHEMA */
} db2DMUXmlValidate;
```

Parámetros de la estructura de datos db2LoadOut

oRowsRead
Salida. Número de registros leídos durante la operación de carga.

oRowsSkipped
Salida. Número de registros que se han omitido antes de que comience la operación de carga.

oRowsLoaded
Salida. Número de filas cargadas en la tabla destino.

oRowsRejected
Salida. Número de registros que no se han podido cargar.

oRowsDeleted
Salida. Número de filas duplicadas suprimidas.

oRowsCommitted
Salida. Número total de registros procesados: el número de registros cargados satisfactoriamente y confirmados en la base de datos, más el número de registros omitidos y rechazados.

Parámetros de la estructura de datos db2PartLoadIn

piHostname
Entrada. Nombre del sistema principal para el parámetro iFileTransferCmd. Si es NULL, el nombre de sistema principal tomará ″nohost″ por omisión. Este parámetro está en desuso.

piFileTransferCmd
Entrada. Parámetro del mandato de transferencia de archivos. Si no es necesario, debe establecerse en NULL. Este parámetro está en desuso. En su lugar, utilice el parámetro piSourceUserExit.

piPartFileLocation
Entrada. En las modalidades PARTITION_ONLY, LOAD_ONLY y LOAD_ONLY_VERIFY_PART, este parámetro puede utilizarse para especificar la ubicación de los archivos particionados. Esta ubicación debe existir en cada partición de base de datos especificada por la opción piOutputNodes.

Para el tipo de archivo SQL_CURSOR, este parámetro no puede ser NULL y la ubicación no hace referencia a una vía de acceso, si no a un nombre de archivo totalmente calificado. Este será el nombre de archivo base totalmente calificado de los archivos particionados que se crean en cada partición de base de datos para la modalidad PARTITION_ONLY, o la ubicación de los archivos que se leerán de cada partición de base de datos para la modalidad LOAD_ONLY. Para la modalidad PARTITION_ONLY, pueden crearse múltiples archivos con el nombre base especificado si hay columnas LOB en la tabla de destino. Para los tipos de archivo que no sean SQL_CURSOR, si el valor de este parámetro es NULL, tomará por omisión el directorio actual.

piOutputNodes
Entrada. La lista de particiones de base de datos de salida de carga. NULL indica todos los nodos en los que está definida la tabla de destino.

piPartitioningNodes
Entrada. La lista de nodos de particiones. NULL indica el valor por omisión.

piMode

Entrada. Especifica la modalidad de carga para bases de datos con particiones. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

- DB2LOAD_PARTITION_AND_LOAD

Los datos se distribuyen (posiblemente en paralelo) y se cargan simultáneamente en las particiones de base de datos correspondientes.

- DB2LOAD_PARTITION_ONLY

Los datos se distribuyen (posiblemente en paralelo) y la salida se graba en archivos en una ubicación especificada en cada partición de base de datos que se carga. Para los tipos de archivo que no sean SQL_CURSOR, el nombre del archivo de salida en cada partición de base de datos tendrá el formato nombrearchivo.xxx, donde nombrearchivo es el nombre del primer archivo de entrada especificado por piSourceList y xxx es el número de partición de base de datos. Para el tipo de archivo SQL_CURSOR, el nombre del archivo de salida en cada partición de base de datos estará determinado por el parámetro piPartFileLocation. Consulte el parámetro piPartFileLocation para obtener información sobre cómo especificar la ubicación del archivo de partición de base de datos en cada partición de base de datos.

Nota: Esta modalidad no puede utilizarse para CLI LOAD.

DB2LOAD_LOAD_ONLY

Se supone que los datos ya están distribuidos; se omite el proceso de distribución y los datos se cargan simultáneamente en las particiones de base de datos correspondientes. Para los tipos de archivo que no sean SQL_CURSOR, se espera que el nombre del archivo de entrada en cada partición de base de datos tenga el formato nombrearchivo.xxx, donde nombrearchivo es el nombre del primer archivo especificado por piSourceList y xxx es el número de partición de base de datos de 13 dígitos. Para el tipo de archivo SQL_CURSOR, el nombre del archivo de entrada en cada partición de base de datos estará determinado por el parámetro piPartFileLocation. Consulte el parámetro piPartFileLocation para obtener información sobre cómo especificar la ubicación del archivo de partición de base de datos en cada partición de base de datos.

Nota: Esta modalidad no puede utilizarse al cargar un archivo de datos ubicado en un cliente remoto, ni para CLI LOAD.

DB2LOAD_LOAD_ONLY_VERIFY_PART

Se supone que los datos ya están distribuidos, pero el archivo de datos no contiene una cabecera de partición de base de datos. Se omite el proceso de distribución y los datos se cargan simultáneamente en las particiones de base de datos correspondientes. Durante la operación de carga, se comprueba cada fila para verificar que está en la partición de base de datos correcta. Las filas que contienen violaciones de la partición de base de datos se colocan en un archivo de vuelco si se ha especificado el modificador de tipo de archivo de vuelco. De lo contrario, se descartan las filas. Si existen violaciones de la partición de base de datos en una partición de base de datos de concreta que se carga,

se escribirá un solo aviso en el archivo de mensajes de carga para esa partición de base de datos. Se espera que el nombre del archivo de entrada en cada partición de base de datos tenga el formato nombrearchivo.xxx, donde nombrearchivo es el nombre del primer archivo especificado por piSourceList y xxx es el número de partición de base de datos de 13 dígitos.

Nota: Esta modalidad no puede utilizarse al cargar un archivo de datos ubicado en un cliente remoto, ni para CLI LOAD.

DB2LOAD_ANALYZE
Se genera una correlación de distribución óptima con distribución regular por todas las particiones de base de datos.

piMaxNumPartAgents
Entrada. El número máximo de agentes de particiones. Un valor NULL indica el valor por omisión, que es 25.

piIsolatePartErrs
Entrada. Indica cómo la partición de carga reaccionará ante los errores que se produzcan en particiones de base de datos individuales. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2LOAD_SETUP_ERRS_ONLY
En esta modalidad, los errores que se produzcan en una partición de base de datos durante la configuración, tales como problemas para acceder a una partición de base de datos o problemas para acceder a un espacio de tablas o una tabla de una partición de base de datos, provocarán que la operación de carga se detenga en las particiones de base de datos anómalas pero que continúe en las particiones de base de datos restantes. Los errores que se produzcan en una partición de base de datos mientras se cargan datos provocarán que falle toda la operación y se retrotraiga hasta el último punto de coherencia de cada partición de base de datos.

DB2LOAD_LOAD_ERRS_ONLY
En esta modalidad, los errores que se produzcan en una partición de base de datos durante la configuración provocarán que falle toda la operación de carga. Cuando se produzca un error mientras se cargan datos, las particiones de base de datos con errores se retrotraerán hasta el último punto de coherencia. La operación de carga continuará en las particiones de base de datos restantes hasta que se produzca una anomalía o hasta que se hayan cargado todos los datos. En las particiones de base de datos en las que se hayan cargado todos los datos, los datos no serán visibles después de la operación de carga. La transacción terminará anormalmente debido a los errores de las demás particiones de base de datos. Los datos de todas las particiones de base de datos permanecerán invisibles hasta que se realice una operación de reinicio de carga. Esto hará que los datos recién cargados sean visibles en las particiones de base de datos en las que se haya completado la operación de carga y reanudará la operación de carga en las particiones de base de datos que hayan sufrido un error.

Nota: Esta modalidad no puede utilizarse cuando iAccessLevel está establecido en SQLU_ALLOW_READ_ACCESS y también se ha especificado un destino de copia.

DB2LOAD_SETUP_AND_LOAD_ERRS
En esta modalidad, los errores a nivel de partición de base de datos ocurridos durante la configuración o la carga de datos provocan que el proceso se detenga solamente en las particiones de base de datos afectadas. Al igual que en la modalidad DB2LOAD_LOAD_ERRS_ONLY, cuando se produzcan errores de partición de base de datos mientras se cargan datos, los datos de todas las particiones de base de datos permanecerán invisibles hasta que se realice una operación de reinicio de carga.

Nota: Esta modalidad no puede utilizarse cuando iAccessLevel está establecido en SQLU_ALLOW_READ_ACCESS y también se ha especificado un destino de copia.

DB2LOAD_NO_ISOLATION
Cualquier error durante la operación de carga provoca que la transacción termine anormalmente. Si este parámetro es NULL, tomará por omisión DB2LOAD_LOAD_ERRS_ONLY, a menos que iAccessLevel esté establecido en SQLU_ALLOW_READ_ACCESS y también se haya especificado un destino de copia, en cuyo caso el valor por omisión es DB2LOAD_NO_ISOLATION.

piStatusInterval
Entrada. Especifica el número de megabytes (MB) de datos a cargar antes de generar un mensaje de situación. Los valores válidos son números enteros entre 1 y 4000. Si se especifica NULL, se utilizará un valor por omisión de 100.

piPortRange
Entrada. El rango de puertos TCP para comunicaciones internas. Si es NULL, el rango de puertos utilizado será de 6000-6063.

piCheckTruncation
Entrada. Provoca que Load compruebe el truncamiento de registros en la entrada/salida. Los valores válidos son TRUE y FALSE. Si es NULL, el valor por omisión es FALSE.

piMapFileInput
Entrada. Nombre de archivo de entrada de correlación de distribución. Si la modalidad no es ANALYZE, este parámetro deberá establecerse en NULL. Si la modalidad es ANALYZE, debe especificarse este parámetro.

piMapFileOutput
Entrada. Nombre de archivo de salida de correlación de distribución. Las reglas para piMapFileInput son también aplicables aquí.

piTrace
Entrada. Especifica el número de registros a rastrear cuando necesite revisar un vuelco de todo el proceso de conversión de datos y la salida de valores hash. Si es NULL, el número de registros toma 0 por omisión.

piNewline
Entrada. Fuerza Load a comprobar si hay caracteres de línea nueva al final de registros de datos ASC si también se especifica el modificador de tipo de archivo RECLEN. Los valores posibles son TRUE y FALSE. Si es NULL, el valor toma FALSE por omisión.

piDistfile
Entrada. Nombre del archivo de distribución de partición de base de datos. Si se especifica NULL, el valor toma por omisión ″DISTFILE″.

piOmitHeader
Entrada. Indica que no deberá incluirse una cabecera de correlación de distribución en el archivo de partición de base de datos al utilizar la modalidad DB2LOAD_PARTITION_ONLY. Los valores posibles son TRUE y FALSE. Si es NULL, el valor por omisión es FALSE.

piRunStatDBPartNum
Especifica la partición de base de datos en la que se deben reunir estadísticas. El valor por omisión es la primera partición de base de datos de la lista de particiones de base de datos de salida.

Parámetros de la estructura de datos db2LoadNodeList

piNodeList
Entrada. Matriz de números de nodo.

iNumNodes
Entrada. Número de nodos de la matriz piNodeList. Un 0 indica el valor por omisión, que es todos los nodos en los que está definida la tabla de destino.

Parámetros de la estructura de datos db2LoadPortRange

iPortMin
Entrada. Número de puerto inferior.

iPortMax
Entrada. Número de puerto superior.

Parámetros de la estructura de datos db2PartLoadOut

oRowsRdPartAgents
Salida. Número total de filas leídas por todos los agentes de particiones.

oRowsRejPartAgents
Salida. Número total de filas rechazadas por todos los agentes de particiones.

oRowsPartitioned
Salida. Número total de filas particionadas por todos los agentes de particiones.

poAgentInfoList
Salida. Durante una operación de carga en una base de datos particionada, pueden estas implicados los siguientes entes de proceso de carga: agentes de carga, agentes de particiones, agentes de preparticiones, agentes de mandatos de transferencia de archivos y agentes de carga a archivo (estos están descritos en la Guía de movimiento de datos). La finalidad del parámetro de salida poAgentInfoList es devolver al llamador información sobre cada agente de carga que ha participado en una operación de carga. Cada entrada de la lista contiene la siguiente información:

oAgentType
Un código que indica qué clase de agente de carga describe la entrada.

oNodeNum
Número de la partición de base de datos en la que se ejecuta el agente.

oSqlcode
El sqlcode final resultante del proceso del agente.

oTableState
El estado final de la tabla en la partición de base de datos en la que el agente se ha ejecutado (pertinente solamente a los agentes de carga).

Es responsabilidad del llamador de la API asignar memoria para esta lista antes de llamar a la API. El llamador también deberá indicar el número de entradas para las que asignaron memoria en el parámetro iMaxAgentInfoEntries. Si el llamador establece poAgentInfoList en NULL o establece iMaxAgentInfoEntries en 0, no se devolverá información sobre los agentes de carga.

iMaxAgentInfoEntries
Entrada. El número máximo de entradas de información de agente asignadas por el usuario para poAgentInfoList. Por lo general, establecer este parámetro en 3 veces el número de particiones de base de datos implicadas en la operación de carga deberá ser suficiente.

oNumAgentInfoEntries
Salida. El número real de entradas de información de agente generadas por la operación de carga. Este número de entradas se devolverá al usuario en el parámetro poAgentInfoList siempre que iMaxAgentInfoEntries sea mayor o igual que oNumAgentInfoEntries. Si iMaxAgentInfoEntries es menor que oNumAgentInfoEntries, el número de entradas devueltas en poAgentInfoList es igual a iMaxAgentInfoEntries.

Parámetros de la estructura de datos db2LoadAgentInfo

oSqlcode
Salida. El sqlcode final resultante del proceso del agente.

oTableState
Salida. La finalidad de este parámetro de salida es no informar de cada posible estado de la tabla tras la operación de carga, si no que su finalidad sería informar solamente de un pequeño subconjunto de posibles estados de tabla para ofrecer al llamador una idea general de lo sucedido en la tabla durante el proceso de carga. Este valor solamente es pertinente para los agentes de carga. Los valores posibles son:

DB2LOADQUERY_NORMAL
Indica que la carga se ha completado satisfactoriamente en la partición de base de datos y que la tabla ha salido del estado LOAD IN PROGRESS (o LOAD PENDING). En este caso, la tabla podría estar aún en estado SET INTEGRITY PENDING debido a la necesidad de más proceso de restricciones, pero no se informará de ello ya que es algo normal.

DB2LOADQUERY_UNCHANGED
Indica que el trabajo de carga ha terminado anormalmente el proceso debido a un error pero aún no ha cambiado el estado de la tabla en la partición de base de datos desde el estado en que estuviera antes de llamar a db2Load. No es necesario realizar un reinicio de carga o terminar la operación en tales particiones de base de datos.

DB2LOADQUERY_LOADPENDING
Indica que el trabajo de carga ha terminado anormalmente durante el proceso pero ha dejado la tabla de la partición de base de datos en estado LOAD PENDING, indicando que el trabajo de carga en esa partición de base de datos debe terminarse o reiniciarse.

oNodeNum
Salida. Número de la partición de base de datos en la que se ejecuta el agente.

oAgentType
Salida. Tipo de agente. Los valores válidos (definidos en el archivo de cabecera db2ApiDf del directorio de inclusión) son:

- DB2LOAD_LOAD_AGENT
- DB2LOAD_PARTITIONING_AGENT
- DB2LOAD_PRE_PARTITIONING_AGENT
- DB2LOAD_FILE_TRANSFER_AGENT
- DB2LOAD_LOAD_TO_FILE_AGENT

Parámetros específicos de la estructura de datos db2gLoadStruct

iFileTypeLen
Entrada. Especifica la longitud, en bytes, del parámetro iFileType.

iLocalMsgFileLen
Entrada. Especifica la longitud, en bytes, del parámetro iLocalMsgFileName.

iTempFilesPathLen
Entrada. Especifica la longitud, en bytes, del parámetro iTempFilesPath.

piXmlPathList
Entrada. Puntero a sqlu_media_list cuyo campo media_type se ha establecido en SQLU_LOCAL_MEDIA y cuya estructura sqlu_media_entry lista vías de acceso del cliente donde se encuentran los archivos xml.

Parámetros específicos de la estructura de datos db2gLoadIn

iUseTablespaceLen
Entrada. Longitud, en bytes, del parámetro piUseTablespace.

piXmlParse
Entrada. Tipo de análisis que debe realizarse para los documentos XML. Los valores válidos que se encuentran en el archivo de cabecera db2ApiDf del directorio de inclusión son:

DB2DMU_XMLPARSE_PRESERVE_WS
Los espacios en blanco deben conservarse.

DB2DMU_XMLPARSE_STRIP_WS
Los caracteres en blanco deben eliminarse.

piXmlValidate
Entrada. Puntero a la estructura db2DMUXmlValidate. Indica que debe llevarse a cabo la validación de esquemas XML para los documentos XML.

```
/* Validar estructura XML
typedef SQL_STRUCTURE db2DMUXmlValidate
{
   db2Uint16                         iUsing;      /* Lo que se debe utilizar  */
                                                  /* para la validación       */
   struct db2DMUXmlValidateXds    *piXdsArgs;    /* Argumentos para          */
                                                  /* XMLVALIDATE USING XDS    */
   struct db2DMUXmlValidateSchema *piSchemaArgs; /* Argumentos para          */
                                                  /* XMLVALIDATE USING SCHEMA */
} db2DMUXmlValidate;
```

Parámetros específicos de la estructura de datos db2gPartLoadIn

piReserved1
Reservado para una utilización futura.

iHostnameLen
Entrada. Longitud, en bytes, del parámetro piHostname.

iFileTransferLen
Entrada. Longitud, en bytes, del parámetro piFileTransferCmd.

iPartFileLocLen
Entrada. Longitud, en bytes, del parámetro piPartFileLocation.

iMapFileInputLen
Entrada. Longitud, en bytes, del parámetro piMapFileInput.

iMapFileOutputLen
Entrada. Longitud, en bytes, del parámetro piMapFileOutput.

iDistfileLen
Entrada. Longitud, en bytes, del parámetro piDistfile.

Notas de uso

Los datos se cargan en la secuencia que aparecen en el archivo de entrada. Si se desea una secuencia determinada, se deberán clasificar los datos antes de intentar una carga.

El programa de utilidad de carga crea índices basándose en las definiciones existentes. Las tablas de excepción se utilizan para manejar duplicados en claves exclusivas. El programa de utilidad no fuerza la integridad de referencia, no realiza ninguna comprobación de restricciones ni actualiza las tablas de resumen que son dependientes de las tablas que se están cargando. Las tablas que incluyen restricciones de referencia o comprobación se colocan en estado de pendiente de establecer integridad. Las tablas de resumen que se definen con REFRESH IMMEDIATE y que son dependientes de tablas que se están cargando, también se colocan en estado de pendiente de establecer integridad. Emita la sentencia SET INTEGRITY para sacar las tablas del estado de pendiente de establecer integridad. Las operaciones de carga no se pueden llevar a cabo en tablas de resumen reproducidas.

Para el agrupamiento en clúster de índices, los datos deben clasificarse en el índice de clúster antes de la carga. No es necesario ordenar los datos al cargar en una tabla de clústeres de múltiples dimensiones (MDC).

db2LoadQuery - Obtener el estado de una operación de carga

Comprueba el estado de una operación de carga durante el proceso.

Autorización

Ninguna

Conexión necesaria

Base de datos

Archivo de inclusión de la API

db2ApiDf.h

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2LoadQuery (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LoadQueryStruct
{
   db2Uint32 iStringType;
   char *piString;
   db2Uint32 iShowLoadMessages;
   struct db2LoadQueryOutputStruct *poOutputStruct;
   char *piLocalMessageFile;
} db2LoadQueryStruct;

typedef SQL_STRUCTURE db2LoadQueryOutputStruct
{
   db2Uint32 oRowsRead;
   db2Uint32 oRowsSkipped;
   db2Uint32 oRowsCommitted;
   db2Uint32 oRowsLoaded;
   db2Uint32 oRowsRejected;
   db2Uint32 oRowsDeleted;
   db2Uint32 oCurrentIndex;
   db2Uint32 oNumTotalIndexes;
   db2Uint32 oCurrentMPPNode;
   db2Uint32 oLoadRestarted;
   db2Uint32 oWhichPhase;
   db2Uint32 oWarningCount;
   db2Uint32 oTableState;
} db2LoadQueryOutputStruct;

typedef SQL_STRUCTURE db2LoadQueryOutputStruct64
{
   db2Uint64 oRowsRead;
   db2Uint64 oRowsSkipped;
   db2Uint64 oRowsCommitted;
   db2Uint64 oRowsLoaded;
   db2Uint64 oRowsRejected;
   db2Uint64 oRowsDeleted;
   db2Uint32 oCurrentIndex;
   db2Uint32 oNumTotalIndexes;
   db2Uint32 oCurrentMPPNode;
   db2Uint32 oLoadRestarted;
   db2Uint32 oWhichPhase;
   db2Uint32 oWarningCount;
   db2Uint32 oTableState;
} db2LoadQueryOutputStruct64;

typedef SQL_STRUCTURE db2LoadQueryStruct64
{
   db2Uint32 iStringType;
   char *piString;
   db2Uint32 iShowLoadMessages;
   struct db2LoadQueryOutputStruct64 *poOutputStruct;
   char *piLocalMessageFile;
} db2LoadQueryStruct64;

SQL_API_RC SQL_API_FN
  db2gLoadQuery (
   db2Uint32 versionNumber,
   void * pParmStruct,
```

```
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gLoadQueryStruct
{
   db2Uint32 iStringType;
   db2Uint32 iStringLen;
   char *piString;
   db2Uint32 iShowLoadMessages;
   struct db2LoadQueryOutputStruct *poOutputStruct;
   db2Uint32 iLocalMessageFileLen;
   char *piLocalMessageFile;
} db2gLoadQueryStruct;

typedef SQL_STRUCTURE db2gLoadQueryStru64
{
   db2Uint32 iStringType;
   db2Uint32 iStringLen;
   char *piString;
   db2Uint32 iShowLoadMessages;
   struct db2LoadQueryOutputStruct64 *poOutputStruct;
   db2Uint32 iLocalMessageFileLen;
   char *piLocalMessageFile;
} db2gLoadQueryStru64;
```

Parámetros de la API db2LoadQuery

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2LoadQueryStruct. Si la versión es la Versión 9 o superior, es un puntero a la estructura db2LoadQueryStruct64. Si no, es un puntero a la estructura db2LoadQueryStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2LoadQueryStruct

iStringType
Entrada. Especifica un tipo para piString. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2LOADQUERY_TABLENAME
Especifica un nombre de tabla para su utilización por la API db2LoadQuery.

piString
Entrada. Especifica un nombre de vía de acceso de archivos temporales o un nombre de tabla, dependiendo del valor de iStringType.

iShowLoadMessages
Entrada. Especifica el nivel de los mensajes que deben ser devueltos por el programa de utilidad de carga. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2LOADQUERY_SHOW_ALL_MSGS
Devuelve todos los mensajes de carga.

DB2LOADQUERY_SHOW_NO_MSGS
No devuelve mensajes de carga.

DB2LOADQUERY_SHOW_NEW_MSGS
Devuelve solamente los mensajes que se han generado desde la última llamada a esta API.

poOutputStruct
Salida. Puntero a la estructura db2LoadQueryOutputStruct, que contiene información de resumen sobre la carga. Su valor se establece en NULL si no es necesario un título.

piLocalMessageFile
Entrada. Especifica el nombre de un archivo local donde se colocan los mensajes de salida.

Parámetros de la estructura de datos db2LoadQueryOutputStruct

oRowsRead
Salida. Número de registros que el programa de utilidad ha leído hasta este momento.

oRowsSkipped
Salida. Número de registros que se han omitido antes de que comenzara la operación de carga.

oRowsCommitted
Salida. Número de filas que se han cargado en la tabla de destino hasta este momento.

oRowsLoaded
Salida. Número de filas cargadas en la tabla destino hasta este momento.

oRowsRejected
Salida. Número de filas que se han rechazado de la tabla de destino hasta este momento.

oRowsDeleted
Salida. Número de filas que se han suprimido de la tabla de destino hasta este momento (durante la fase de supresión).

oCurrentIndex
Salida. Índice que se está creando actualmente (durante la fase de creación).

oNumTotalIndexes
Salida. Número total de índices que se deben crear (durante la fase de creación).

oCurrentMPPNode
Salida. Indica qué servidor de particiones de base de datos se está consultando (solamente para la modalidad del entorno de bases de datos particionadas).

oLoadRestarted
Salida. Distintivo cuyo valor es TRUE si la operación de carga que se está consultando es una operación de reinicio de carga.

oWhichPhase
Salida. Indica la fase actual de la operación de carga que se está consultando. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2LOADQUERY_LOAD_PHASE
Fase de carga.

DB2LOADQUERY_BUILD_PHASE
Fase de creación.

DB2LOADQUERY_DELETE_PHASE
Fase de supresión.

DB2LOADQUERY_INDEXCOPY_PHASE
Fase de copia de índice.

oWarningCount
Salida. Número total de avisos devueltos hasta ahora.

oTableState
Salida. Los estados de tabla. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2LOADQUERY_NORMAL
No existe ningún estado de tabla que afecte a la tabla.

DB2LOADQUERY_SI_PENDING
La tabla tiene restricciones, que aún no se han verificado. Utilice el mandato SET INTEGRITY para sacar la tabla el estado DB2LOADQUERY_SI_PENDING. El programa de utilidad de carga coloca una tabla en el estado DB2LOADQUERY_SI_PENDING cuando comienza una carga en una tabla que tiene restricciones.

DB2LOADQUERY_LOAD_IN_PROGRESS
Existe un proceso activo de carga en curso en la tabla.

DB2LOADQUERY_LOAD_PENDING
Una carga ha estado activa en esta tabla pero se ha cancelado anormalmente antes de la carga se pudiese confirmar. Emita un mandato para terminar carga, reiniciar carga o sustituir carga para sacar la tabla del estado DB2LOADQUERY_LOAD_PENDING.

DB2LOADQUERY_REORG_PENDING
Es necesario realizar una reorganización para que la tabla pueda ser accesible.

DB2LOADQUERY_READ_ACCESS
Los datos de la tabla son accesibles para las consultas con acceso de lectura. Las cargas realizadas mediante la opción DB2LOADQUERY_READ_ACCESS colocan la tabla en estado Read Access Only (Acceso de sólo lectura)

DB2LOADQUERY_NOTAVAILABLE
No se puede acceder a la tabla. La tabla solamente se puede descartar o se puede restaurar a partir de una copia de seguridad. La recuperación en avance mediante una carga no recuperable colocará la tabla en un estado de no disponibilidad.

DB2LOADQUERY_NO_LOAD_RESTART
La tabla se encuentra en un estado de carga parcial que no permitirá un reinicio de la carga. La tabla también estará en estado Pendiente de carga. Emita una instrucción de finalización de carga (load terminate) o de sustitución de carga (load replace) para sacar la tabla del estado Carga no reiniciable. La tabla puede situarse en estado DB2LOADQUERY_NO_LOAD_RESTART durante una operación de avance. Esto puede ocurrir si realiza una operación de avance hasta un momento anterior al final de una operación de carga, o si realiza una operación de avance hasta una operación de

carga cancelada, pero no realiza la operación de avance hasta el final de la operación de finalización de carga o de reinicio de carga.

DB2LOADQUERY_TYPE1_INDEXES
La tabla utiliza actualmente índices de tipo 1. Los índices se pueden convertir al tipo 2 mediante la opción CONVERT al utilizar el programa de utilidad REORG sobre los índices.

Parámetros de la estructura de datos db2LoadQueryOutputStruct64

oRowsRead
Salida. Número de registros que el programa de utilidad ha leído hasta este momento.

oRowsSkipped
Salida. Número de registros que se han omitido antes de que comenzara la operación de carga.

oRowsCommitted
Salida. Número de filas que se han cargado en la tabla de destino hasta este momento.

oRowsLoaded
Salida. Número de filas cargadas en la tabla destino hasta este momento.

oRowsRejected
Salida. Número de filas que se han rechazado de la tabla de destino hasta este momento.

oRowsDeleted
Salida. Número de filas que se han suprimido de la tabla de destino hasta este momento (durante la fase de supresión).

oCurrentIndex
Salida. Índice que se está creando actualmente (durante la fase de creación).

oNumTotalIndexes
Salida. Número total de índices que se deben crear (durante la fase de creación).

oCurrentMPPNode
Salida. Indica qué servidor de particiones de base de datos se está consultando (solamente para la modalidad del entorno de bases de datos particionadas).

oLoadRestarted
Salida. Distintivo cuyo valor es TRUE si la operación de carga que se está consultando es una operación de reinicio de carga.

oWhichPhase
Salida. Indica la fase actual de la operación de carga que se está consultando. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2LOADQUERY_LOAD_PHASE
Fase de carga.

DB2LOADQUERY_BUILD_PHASE
Fase de creación.

DB2LOADQUERY_DELETE_PHASE
Fase de supresión.

DB2LOADQUERY_INDEXCOPY_PHASE
Fase de copia de índice.

oWarningCount
Salida. Número total de avisos devueltos hasta ahora.

oTableState
Salida. Los estados de tabla. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2LOADQUERY_NORMAL
No existe ningún estado de tabla que afecte a la tabla.

DB2LOADQUERY_SI_PENDING
La tabla tiene restricciones, que aún no se han verificado. Utilice el mandato SET INTEGRITY para sacar la tabla el estado DB2LOADQUERY_SI_PENDING. El programa de utilidad de carga coloca una tabla en el estado DB2LOADQUERY_SI_PENDING cuando comienza una carga en una tabla que tiene restricciones.

DB2LOADQUERY_LOAD_IN_PROGRESS
Existe un proceso activo de carga en curso en la tabla.

DB2LOADQUERY_LOAD_PENDING
Una carga ha estado activa en esta tabla pero se ha cancelado anormalmente antes de la carga se pudiese confirmar. Emita un mandato para terminar carga, reiniciar carga o sustituir carga para sacar la tabla del estado DB2LOADQUERY_LOAD_PENDING.

DB2LOADQUERY_REORG_PENDING
Es necesario realizar una reorganización para que la tabla pueda ser accesible.

DB2LOADQUERY_READ_ACCESS
Los datos de la tabla son accesibles para las consultas con acceso de lectura. Las cargas realizadas mediante la opción DB2LOADQUERY_READ_ACCESS colocan la tabla en estado Read Access Only (Acceso de sólo lectura)

DB2LOADQUERY_NOTAVAILABLE
No se puede acceder a la tabla. La tabla solamente se puede descartar o se puede restaurar a partir de una copia de seguridad. La recuperación en avance mediante una carga no recuperable colocará la tabla en un estado de no disponibilidad.

DB2LOADQUERY_NO_LOAD_RESTART
La tabla se encuentra en un estado de carga parcial que no permitirá un reinicio de la carga. La tabla también estará en estado Pendiente de carga. Emita una instrucción de finalización de carga (load terminate) o de sustitución de carga (load replace) para sacar la tabla del estado Carga no reiniciable. La tabla puede situarse en estado DB2LOADQUERY_NO_LOAD_RESTART durante una operación de avance. Esto puede ocurrir si realiza una operación de avance hasta un momento anterior al final de una operación de carga, o si realiza una operación de avance hasta una operación de carga cancelada, pero no realiza la operación de avance hasta el final de la operación de finalización de carga o de reinicio de carga.

DB2LOADQUERY_TYPE1_INDEXES
La tabla utiliza actualmente índices de tipo 1. Los índices se pueden convertir al tipo 2 mediante la opción CONVERT al utilizar el programa de utilidad REORG sobre los índices.

Parámetros de la estructura de datos db2LoadQueryStruct64

iStringType
Entrada. Especifica un tipo para piString. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2LOADQUERY_TABLENAME
Especifica un nombre de tabla para su utilización por la API db2LoadQuery.

piString
Entrada. Especifica un nombre de vía de acceso de archivos temporales o un nombre de tabla, dependiendo del valor de iStringType.

iShowLoadMessages
Entrada. Especifica el nivel de los mensajes que deben ser devueltos por el programa de utilidad de carga. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2LOADQUERY_SHOW_ALL_MSGS
Devuelve todos los mensajes de carga.

DB2LOADQUERY_SHOW_NO_MSGS
No devuelve mensajes de carga.

DB2LOADQUERY_SHOW_NEW_MSGS
Devuelve solamente los mensajes que se han generado desde la última llamada a esta API.

poOutputStruct
Salida. Puntero a la estructura db2LoadQueryOutputStruct, que contiene información de resumen sobre la carga. Su valor se establece en NULL si no es necesario un título.

piLocalMessageFile
Entrada. Especifica el nombre de un archivo local donde se colocan los mensajes de salida.

Parámetros específicos de la estructura de datos db2gLoadQueryStruct

iStringLen
Entrada. Especifica la longitud, expresada en bytes, del parámetro piString.

iLocalMessageFileLen
Entrada. Especifica la longitud, expresada en bytes, del parámetro piLocalMessageFile.

Parámetros específicos de la estructura de datos db2gLoadQueryStru64

iStringLen
Entrada. Especifica la longitud, expresada en bytes, del parámetro piString.

iLocalMessageFileLen
Entrada. Especifica la longitud, expresada en bytes, del parámetro piLocalMessageFile.

Notas de uso

Esta API lee el estado de la operación de carga en la tabla especificada por piString, y escribe el estado en el archivo especificado por piLocalMsgFileName.

db2MonitorSwitches - Obtener o actualizar los valores de los conmutadores del supervisor

Activa o desactiva selectivamente conmutadores para grupos de datos del supervisor que deben ser recopilados por el gestor de bases de datos. Devuelve el estado actual de estos conmutadores para la aplicación emisora de la llamada.

Ámbito

Esta API puede devolver información para el servidor de particiones de base de datos de la instancia o para todas las particiones de base de datos de la instancia.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint
- sysmon

Conexión necesaria

Instancia. Si no existe ninguna conexión de instancia, se creará una conexión de instancia por omisión.

Para visualizar los valores de una instancia remota o de una instancia local diferente, es necesario conectarse primero a esa instancia.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2MonitorSwitches (
    db2Uint32 versionNumber,
    void * pParmStruct,
    struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2MonitorSwitchesData
{
   struct sqlm_recording_group *piGroupStates;
   void *poBuffer;
   db2Uint32 iBufferSize;
   db2Uint32 iReturnData;
   db2Uint32 iVersion;
   db2int32 iNodeNumber;
   db2Uint32 *poOutputFormat;
```

```
} db2MonitorSwitchesData;

SQL_API_RC SQL_API_FN
  db2gMonitorSwitches (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gMonitorSwitchesData
{
   struct sqlm_recording_group *piGroupStates;
   void *poBuffer;
   db2Uint32 iBufferSize;
   db2Uint32 iReturnData;
   db2Uint32 iVersion;
   db2int32 iNodeNumber;
   db2Uint32 *poOutputFormat;
} db2gMonitorSwitchesData;
```

Parámetros de la API db2MonitorSwitches

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct. Para utilizar la estructura tal como se describe más arriba, especifique db2Version810. Si desea utilizar una versión diferente de esta estructura, vea la lista completa de versiones soportadas en el archivo de cabecera db2ApiDf.h del directorio de inclusión. Debe utilizar la versión de la estructura db2MonitorSwitchesStruct correspondiente al número de versión que especifique.

pParmStruct
Entrada. Puntero a la estructura db2MonitorSwitchesStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2MonitorSwitchesData

piGroupStates
Entrada. Puntero a la estructura sqlm-recording-group (definida en sqlmon.h) que contiene una lista de conmutadores.

poBuffer
Puntero a un almacenamiento intermedio donde se escribirán los datos de estado del conmutador.

iBufferSize
Entrada. Especifica el tamaño del almacenamiento intermedio de salida.

iReturnData
Entrada. Distintivo que especifica si se deben escribir o no los estados actuales de los conmutadores en el almacenamiento intermedio de datos señalado por poBuffer.

iVersion
Entrada. ID de versión de los datos del supervisor de bases de datos que se deben recopilar. El supervisor de bases de datos solamente devuelve datos que estaban disponibles para la versión solicitada. Establezca este parámetro en una de las constantes siguientes:

- SQLM_DBMON_VERSION1
- SQLM_DBMON_VERSION2

- SQLM_DBMON_VERSION5
- SQLM_DBMON_VERSION5_2
- SQLM_DBMON_VERSION6
- SQLM_DBMON_VERSION7
- SQLM_DBMON_VERSION8
- SQLM_DBMON_VERSION9
- SQLM_DBMON_VERSION9_5

Nota: Si se especifica SQLM_DBMON_VERSION1 como versión, las API no se pueden ejecutar remotamente.

Nota: Las constantes SQLM_DBMON_VERSION5_2 y anteriores han quedado obsoletas y es posible que se eliminen en un futuro release de DB2.

iNodeNumber
Entrada. El servidor de particiones de base de datos que debe enviarse la petición. De acuerdo con este valor, la petición se procesará para el servidor de particiones de base de datos actual, para todos los servidores de particiones de base de datos o para un servidor de particiones de base de datos especificado por el usuario. Los valores válidos son:

- SQLM_CURRENT_NODE
- SQLM_ALL_NODES
- valor de nodo

Nota: Para instancias autónomas, se debe utilizar el valor SQLM_CURRENT_NODE.

poOutputFormat
Formato de la corriente de datos devuelta por el servidor. Puede tener uno de estos valores:

SQLM_STREAM_STATIC_FORMAT
Indica que los estados de conmutador se devuelven en estructuras de conmutadores estáticas anteriores a la Versión 7.

SQLM_STREAM_DYNAMIC_FORMAT
Indica que los conmutadores se devuelven en un formato autodescrito, similar al formato devuelto para db2GetSnapshot.

Notas de uso

Para obtener el estado de los conmutadores a nivel del gestor de bases de datos, invoque db2GetSnapshot, especificando SQLMA_DB2 para OBJ_TYPE (obtener instantánea para gestor de bases de datos).

El conmutador de indicación de fecha y hora no está disponible si iVersion es menor que SQLM_DBMON_VERSION8.

db2Prune - Suprimir las entradas del archivo histórico o archivos de anotaciones cronológicas de la vía de acceso de anotación cronológica activa

Suprime entradas del archivo histórico o archivos de anotaciones de la vía de acceso del archivo de anotaciones activo.

Autorización

Una de las siguientes:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Conexión necesaria

Base de datos. Para suprimir entradas del archivo histórico para cualquier base de datos que no sea la base de datos por omisión, se debe establecer una conexión con la base de datos antes de invocar esta API.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2Prune (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2PruneStruct
{
  char *piString;
  db2HistoryEID iEID;
  db2Uint32 iAction;
  db2Uint32 iOptions;
} db2PruneStruct;

SQL_API_RC SQL_API_FN
  db2gPrune (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gPruneStruct
{
  db2Uint32 iStringLen;
  char *piString;
  db2HistoryEID iEID;
  db2Uint32 iAction;
  db2Uint32 iOptions;
} db2gPruneStruct;
```

Parámetros de la API db2Prune

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2PruneStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2PruneStruct

piString
Entrada. Puntero a una serie que especifica una indicación de fecha y hora o un número de secuencia de anotaciones cronológicas (LSN). Se utiliza la indicación de fecha y hora o parte de ella (como mínimo aaaa o año) para seleccionar registros para la supresión. Se suprimirán todas las entradas iguales o menores que la indicación de fecha y hora. Debe facilitarse una indicación de fecha y hora válida; un valor de parámetro NULL no es válido.

Este parámetro también se puede utilizar para pasar un LSN, a fin de poder suprimir archivos de anotaciones inactivos.

iEID Entrada. Especifica un identificador exclusivo que se puede utilizar para suprimir una entrada individual del archivo histórico.

iAction
Entrada. Especifica el tipo de acción que se debe realizar. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2PRUNE_ACTION_HISTORY
Elimina entradas del archivo histórico

DB2PRUNE_ACTION_LOG
Elimina archivos de anotaciones de la vía de acceso del archivo de anotaciones activo.

iOptions
Entrada. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2PRUNE_OPTION_FORCE
Fuerza la supresión de la última copia de seguridad.

DB2PRUNE_OPTION_DELETE
Suprime los archivos de anotaciones que se eliminan del archivo histórico.

Si se establece el parámetro de configuración de base de datos **auto_del_rec_obj** en ON, la llamada a db2Prune con DB2PRUNE_OPTION_DELETE también hace que se supriman las imágenes de copia de seguridad asociadas y las imágenes de copia de seguridad de carga.

DB2PRUNE_OPTION_LSNSTRING
Especifica que el valor de piString es un LSN, que se utiliza cuando se especifica DB2PRUNE_ACTION_LOG como acción del que llama.

Parámetros específicos de la estructura de datos db2gPruneStruct

iStringLen
Entrada. Especifica la longitud, en bytes, de piString.

Notas de uso

Estas entradas con estado do_not_delete no se recortarán ni suprimirán. Puede establecer el estado de las entradas de archivo histórico de recuperación en do_not_delete utilizando el mandato UPDATE HISTORY, ADMIN_CMD con UPDATE_HISTORY o la API db2HistoryUpdate. Puede utilizar el estado do_not_delete para evitar que se recorten o supriman las entradas del archivo histórico de recuperación de claves.

Si se suprime del soporte de almacenamiento la última copia de seguridad completa de la base de datos (además de suprimirla del archivo histórico), el usuario debe asegurarse de que exista copia de seguridad de todos los espacios de tablas, incluidos el espacio de tablas de catálogo y los espacios de tablas de usuario. De no hacerlo, puede dar como resultado el no poder recuperar una base de datos o la pérdida de algunos de los datos de usuario contenidos en la base de datos.

Puede recortar las entradas del archivo histórico de base de datos de copia de seguridad instantánea utilizando db2Prune, pero no puede suprimir los objetos de recuperación físicos relacionados utilizando el parámetro DB2PRUNE_OPTION_DELETE. La manera de suprimir los objetos de copia de seguridad instantánea es utilizar el mandato db2acsutil.

Sintaxis de la API de REXX

```
PRUNE RECOVERY HISTORY BEFORE :indicación de fecha y hora [WITH FORCE OPTION]
```

Parámetros de la API de REXX

timestamp
Variable de lenguaje principal que contiene una indicación de fecha y hora. Todas las entradas con indicaciones de fecha y hora iguales o anteriores a la proporcionada se suprimen del archivo histórico.

WITH FORCE OPTION
Si se especifica este parámetro, el archivo histórico se suprime de acuerdo con la indicación de fecha y hora especificada, aunque se supriman del archivo algunas entradas pertenecientes al conjunto de restauración más reciente. Si no se especifica este parámetro, se conserva el conjunto de restauración más reciente, aunque la indicación de fecha y hora sea anterior o igual a la especificada como entrada.

db2QuerySatelliteProgress - Obtener el estado de una sesión de sincronización de satélites

Comprueba el estado de una sesión de sincronización de satélites.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2QuerySatelliteProgress (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef struct db2QuerySatelliteProgressStruct
{
   db2int32 oStep;
   db2int32 oSubstep;
   db2int32 oNumSubsteps;
   db2int32 oScriptStep;
   db2int32 oNumScriptSteps;
   char *poDescription;
   char *poError;
   char *poProgressLog;
} db2QuerySatelliteProgressStruct;
```

Parámetros de la API db2QuerySatelliteProgress

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2QuerySatelliteProgressStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2QuerySatelliteProgressStruct

oStep Salida. Paso actual de la sesión de sincronización (definido en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión).

oSubstep
Salida. Si el paso de sincronización indicado por el parámetro oStep se puede desglosar en subpasos, este parámetro indica el subpaso actual.

oNumSubsteps
Salida. Si el paso actual de la sesión de sincronización contiene subpasos (oSubstep), este parámetro indica el número total de subpasos comprendidos en el paso de sincronización.

oScriptStep
Salida. Si el subpaso actual es la ejecución de un script, este parámetro informa sobre el progreso de la ejecución del script, si esa información está disponible.

oNumScriptSteps
Salida. Si se informa sobre la ejecución de un script, este parámetro contiene el número total de pasos que intervienen en la ejecución del script.

poDescription
Salida. Descripción del estado de la sesión de sincronización del satélite.

poError
Salida. Si la sesión de sincronización es errónea, este parámetro proporciona una descripción del error.

poProgressLog
Salida. Este parámetro devuelve el archivo de anotaciones completo de la sesión de sincronización del satélite.

db2ReadLog - Extraer registros de anotaciones cronológicas

Extrae registros de anotaciones cronológicas de las anotaciones cronológicas de base de datos DB2 y del gestor de anotaciones cronológicas para obtener información del estado de anotación cronológica actual. Esta API sólo se puede utilizar con bases de datos recuperables. Una base de datos es recuperable si los parámetros de configuración de la base de datos logarchmeth1 y/o logarchmeth2 no están establecidos en OFF.

Autorización

Una de las siguientes:

- sysadm
- dbadm

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2ReadLog (
       db2Uint32 versionNumber,
       void * pDB2ReadLogStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ReadLogStruct
{
   db2Uint32 iCallerAction;
   SQLU_LSN *piStartLSN;
   SQLU_LSN *piEndLSN;
   char *poLogBuffer;
   db2Uint32 iLogBufferSize;
   db2Uint32 iFilterOption;
   db2ReadLogInfoStruct *poReadLogInfo;
} db2ReadLogStruct;

typedef SQL_STRUCTURE db2ReadLogInfoStruct
{
   SQLU_LSN initialLSN;
   SQLU_LSN firstReadLSN;
   SQLU_LSN nextStartLSN;
   db2Uint32 logRecsWritten;
   db2Uint32 logBytesWritten;
   SQLU_LSN firstReusedLSN;
   db2Uint32 timeOfLSNReuse;
```

```
   db2TimeOfLog currentTimeValue;
} db2ReadLogInfoStruct;

typedef SQL_STRUCTURE db2TimeOfLog
{
   db2Uint32 seconds;
   db2Uint32 accuracy;
} db2TimeOfLog;
```

Parámetros de la API db2ReadLog

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pDB2ReadLogStruct.

pDB2ReadLogStruct
Entrada. Puntero a la estructura db2ReadLogStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2ReadLogStruct

iCallerAction
Entrada. Especifica la acción que se debe realizar.

DB2READLOG_READ
Leer el archivo de anotaciones de la base de datos desde el número de secuencia de anotaciones inicial hasta el número de secuencia de anotaciones final y devolver registros de anotaciones comprendidos en este rango.

DB2READLOG_READ_SINGLE
Leer un registro de anotaciones individual (propagable o no) identificado por el número de secuencia de anotaciones inicial.

DB2READLOG_QUERY
Consultar el archivo de anotaciones de la base de datos. Los resultados de la consulta se devuelven a través de la estructura db2ReadLogInfoStruct.

piStartLsn
Entrada. El número de secuencia de anotación inicial especifica la dirección de byte relativa inicial para la lectura del archivo de anotaciones. Este valor debe ser el inicio de un registro de anotaciones real.

piEndLsn
Entrada. El número de secuencia de anotación final especifica la dirección de byte relativa final para la lectura del archivo de anotaciones. Este valor debe ser mayor que el parámetro startLsn y no es necesario que sea el final de un registro de anotaciones real.

poLogBuffer
Salida. El almacenamiento intermedio donde se almacenan secuencialmente todos los registros de anotaciones propagables dentro del rango especificado. Este almacenamiento intermedio debe ser lo suficientemente grande para contener un solo registro de anotaciones. Como directriz, este almacenamiento intermedio deberá ser de un mínimo de 32 bytes. Su tamaño máximo depende del tamaño del rango solicitado. Cada registro de anotaciones del almacenamiento intermedio lleva como prefijo un número de secuencia de anotaciones (LSN) de seis bytes que representa el LSN del siguiente registro de anotaciones.

iLogBufferSize
Entrada. Especifica el tamaño del almacenamiento intermedio de anotaciones, expresado en bytes.

iFilterOption
Entrada. Especifica el nivel de filtrado para registros de anotaciones que se debe utilizar al leer esos registros. Los valores válidos son:

DB2READLOG_FILTER_OFF
Leer todos los registros de anotaciones comprendidos dentro del rango de LSN especificado.

DB2READLOG_FILTER_ON
Sólo lee los registros de anotaciones cronológicas en el rango de LSN determinado que estén marcados como propagables. Comportamiento habitual de la API de lectura asíncrona de archivos de anotaciones. Los registros de anotaciones cronológicas que se devuelven cuando se utiliza este valor están documentados en el tema ″Registros de anotaciones cronológicas de DB2″. Todos los demás registros de anotaciones cronológicas están destinados solamente al uso interno de IBM y por lo tanto no están documentados.

poReadLogInfo
Salida. Estructura que detalla información sobre la llamada y la anotación de la base de datos.

Parámetros de la estructura de datos db2ReadLogInfoStruct

initialLSN
Es el primer LSN utilizado o que será utilizado por la base de datos desde que se activó.

firstReadLSN
Es el primer LSN existente en el parámetro poLogBuffer.

nextStartLSN
Es el inicio del siguiente registro de anotaciones que el llamador debe leer. Debido a que algunos registros de anotaciones pueden ser filtrados y no ser devueltos en el parámetro poLogBuffer, la utilización de este LSN como inicio de la lectura siguiente en lugar del final del último registro de anotaciones contenido en el parámetro poLogBuffer evita que se exploren de nuevo registros de anotaciones que ya se han filtrado.

logRecsWritten
Número de registros de anotaciones escritos en el parámetro poLogBuffer.

logBytesWritten
Número total de bytes de datos escritos en el parámetro poLogBuffer.

firstReusedLSN
Primer LSN que se debe reutilizar debido a una operación de restauración o avance de una base de datos.

timeOfLSNReuse
Número de segundos transcurridos desde el 1 de enero de 1970 en que se reutilizó el LSN representado por firstReusedLSN.

currentTimeValue
La hora actual de acuerdo con la base de datos.

Parámetros de la estructura de datos db2TimeOfLog

segundos
Número de segundos desde el 1 de enero de 1970.

accuracy
Contador de alta precisión que permite a los llamadores distinguir el orden de los sucesos al comparar indicaciones horarias producidas dentro del mismo segundo.

Notas de uso

Si la acción solicitada es leer el archivo de anotaciones, debe proporcionar un rango de números de secuencia de registros de anotaciones y un almacenamiento intermedio para contener los registros. Esta API lee el archivo de anotaciones cronológicas secuencialmente, dentro de los límites indicados por el rango de LSN solicitado, y devuelve registros de anotaciones cronológicas asociados a tablas definidas con la cláusula DATA CAPTURE CHANGES, y una estructura db2ReadLogInfoStruct que contiene la información de anotación cronológica activa. Si la acción solicitada es una consulta del archivo de anotaciones de la base de datos (acción que se indica especificando el valor DB2READLOG_QUERY), la API devuelve una estructura db2ReadLogInfoStruct con la información de anotación cronológica activa actual.

Para utilizar el Lector de anotaciones asíncronas, primero consulte las anotaciones de base de datos para obtener un LSN inicial válido. Después de la consulta, la estructura db2ReadLogInfoStruct contendrá un LSN inicial válido (en el miembro initialLSN) para utilizarlo en una llamada de lectura. El valor utilizado como LSN final en una lectura puede ser uno de estos valores:

- Un valor mayor que initialLSN
- FFFF FFFF FFFF, que es interpretado por el lector de anotaciones asíncronas como el final del archivo de anotaciones actual.

Los registros de anotaciones propagables que se leen dentro del rango comprendido entre el LSN inicial y el LSN final se devuelven en el almacenamiento intermedio de anotaciones. Un registro de anotaciones no contiene su LSN; el LSN reside en el almacenamiento intermedio, delante del registro de anotaciones real. Encontrará descripciones de los diversos registros de anotaciones cronológicas de DB2 devueltos por db2ReadLog en la sección Registros de anotaciones cronológicas de DB2.

Para leer el siguiente registro de anotaciones secuencial después de la lectura inicial, utilice el campo nextStartLSN devuelto en la estructura db2ReadLogStruct. Emita de nuevo la llamada con este nuevo LSN inicial y un LSN final válido. A continuación, se leerá el próximo bloque de registros. Un código de sqlca igual a SQLU_RLOG_READ_TO_CURRENT significa que el lector de anotaciones cronológicas ha leído hasta el final la anotación cronológica activa actual.

Esta API lee datos de las anotaciones cronológicas de DB2. En dichas anotaciones cronológicas no se impone el control de acceso basado en etiquetas (Label-based access control - LBAC). Por lo tanto, una aplicación que llame a esta API puede obtener acceso a los datos de tablas si el llamador tiene autorización suficiente para llamar a la API y puede comprender el formato de los registros de anotaciones cronológicas.

La API db2ReadLog trabaja sobre la conexión de base de datos actual. Si se crean varias conexiones de base de datos con el mismo proceso, utilice las API de acceso concurrente para gestionar los múltiples contextos.

Si se llama la API db2ReadLog desde una aplicación puede producirse un error cuando la aplicación se desconecta si no se confirma o retrotrae antes de la desconexión:

- Se puede producir un error CLI0116E si la API db2ReadLog se llama desde una aplicación de CLI.
- Se puede producir un error SQL0428N si la API db2ReadLog se llama desde una aplicación de SQL incorporado escrita en C.

Solución 1: Para las aplicaciones de SQL no incorporado, active la modalidad de confirmación automática antes de invocar la API db2ReadLog.

Solución 2: Emita una sentencia COMMIT o ROLLBACK después de invocar la API db2ReadLog y antes de desconectarse de la base de datos.

db2ReadLogNoConn - Leer las anotaciones cronológicas de la base de datos sin una conexión de base de datos

Extrae registros de anotaciones cronológicas de las anotaciones cronológicas de base de datos DB2 y consulta el Gestor de anotaciones cronológicas para obtener información sobre el estado actual de la anotación cronológica. Antes de utilizar esta API, llame a la API db2ReadLogNoConnInit para asignar la memoria que se transfiere como parámetro de entrada a esta API. Después de invocar esta API, llame a la API db2ReadLogNoConnTerm para desasignar la memoria.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2ReadLogNoConn (
       db2Uint32 versionNumber,
       void * pDB2ReadLogNoConnStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ReadLogNoConnStruct
{
   db2Uint32 iCallerAction;
   SQLU_LSN *piStartLSN;
   SQLU_LSN *piEndLSN;
   char *poLogBuffer;
   db2Uint32 iLogBufferSize;
   char *piReadLogMemPtr;
   db2ReadLogNoConnInfoStruct *poReadLogInfo;
} db2ReadLogNoConnStruct;
```

```
typedef SQL_STRUCTURE db2ReadLogNoConnInfoStruct
{
   SQLU_LSN firstAvailableLSN;
   SQLU_LSN firstReadLSN;
   SQLU_LSN nextStartLSN;
   db2Uint32 logRecsWritten;
   db2Uint32 logBytesWritten;
   db2Uint32 lastLogFullyRead;
   db2TimeOfLog currentTimeValue;
} db2ReadLogNoConnInfoStruct;
```

Parámetros de la API db2ReadLogNoConn

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura pasada como segundo parámetro, pDB2ReadLogNoConnStruct.

pDB2ReadLogNoConnStruct
Entrada. Puntero a la estructura db2ReadLogNoConnStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2ReadLogNoConnStruct

iCallerAction
Entrada. Especifica la acción que se debe realizar. Los valores válidos son:

DB2READLOG_READ
Leer el archivo de anotaciones de la base de datos desde el número de secuencia de anotaciones inicial hasta el número de secuencia de anotaciones final y devolver registros de anotaciones comprendidos en este rango.

DB2READLOG_READ_SINGLE
Leer un registro de anotaciones individual (propagable o no) identificado por el número de secuencia de anotaciones inicial.

DB2READLOG_QUERY
Consultar el archivo de anotaciones de la base de datos. Los resultados de la consulta se devuelven a través de la estructura db2ReadLogNoConnInfoStruct.

piStartLSN
Entrada. El número de secuencia de anotación inicial especifica la dirección de byte relativa inicial para la lectura del archivo de anotaciones. Este valor debe ser el inicio de un registro de anotaciones real.

piEndLSN
Entrada. El número de secuencia de anotación final especifica la dirección de byte relativa final para la lectura del archivo de anotaciones. Este valor debe ser mayor que piStartLsn y no tiene que ser el final de un registro de anotaciones real.

poLogBuffer
Salida. El almacenamiento intermedio donde se almacenan secuencialmente todos los registros de anotaciones propagables dentro del rango especificado. Este almacenamiento intermedio debe ser lo suficientemente grande para contener un solo registro de anotaciones. Como directriz, este almacenamiento intermedio deberá ser de un mínimo de 32 bytes. Su tamaño máximo depende del tamaño del rango solicitado.

Cada registro de anotaciones del almacenamiento intermedio lleva como prefijo un número de secuencia de anotaciones (LSN) de seis bytes que representa el LSN del siguiente registro de anotaciones.

iLogBufferSize
Entrada. Especifica el tamaño del almacenamiento intermedio de anotaciones, expresado en bytes.

piReadLogMemPtr
Entrada. Bloque de memoria de tamaño iReadLogMemoryLimit que se ha asignado en la llamada de inicialización. Esta memoria contiene datos permanente que la API requiere en cada invocación. Este bloque de memoria no debe reasignarse ni modificarse de ningún modo por parte del llamador.

poReadLogInfo
Salida. Puntero de la estructura db2ReadLogNoConnInfoStruct.

Parámetros de la estructura de datos db2ReadLogNoConnInfoStruct

firstAvailableLSN
Primer LSN disponible que está contenido en los archivos de anotaciones disponibles.

firstReadLSN
Primer LSN leído en esta llamada.

nextStartLSN
Siguiente LSN legible.

logRecsWritten
Número de registros de anotaciones escritos en el campo del almacenamiento intermedio de anotaciones, poLogBuffer.

logBytesWritten
Número de bytes escritos en el campo del almacenamiento intermedio de anotaciones, poLogBuffer.

lastLogFullyRead
Número que indica el último de archivo de anotaciones que se leyó por completo.

currentTimeValue
Reservado para una utilización futura.

Notas de uso

La API db2ReadLogNoConn necesita un bloque de memoria que se debe asignar mediante la API db2ReadLogNoConnInit. El bloque de memoria se debe pasar como parámetro de entrada a todas las llamadas subsiguientes a la API db2ReadLogNoConn, y no debe ser alterado.

Cuando se solicita una lectura secuencial del archivo de anotaciones, la API necesita un rango de números de secuencia de archivos de anotaciones (LSN) y la memoria asignada. La API devolverá una secuencia de registros de anotaciones de acuerdo con la opción de filtro especificada al inicializar la API y el rango de LSN. Cuando se solicita una consulta, la estructura para leer información de archivo de anotaciones contendrá un LSN inicial válido, que se debe utilizar en una llamada de lectura. El valor utilizado como LSN final en una lectura puede ser uno de estos valores:

- Un valor mayor que el LSN inicial especificado por el llamador.
- FFFF FFFF FFFF, que el lector de archivos de anotaciones asíncrono interpreta como el final de los archivos de anotaciones disponibles.

Los registros de anotaciones propagables leídos dentro del rango de LSN inicial y final se devuelven en el almacenamiento intermedio de anotaciones. Un registro de anotaciones no contiene su LSN, si no que está contenido en el almacenamiento intermedio antes del registro de anotaciones. Encontrará descripciones de los diversos registros de anotaciones cronológicas de DB2 devueltos por db2ReadLogNoConn en la sección Registros de anotaciones cronológicas de DB2.

Después de la lectura inicial, para leer el siguiente registro de anotaciones secuencial, utilice el valor nextStartLSN devuelto en db2ReadLogNoConnInfoStruct. Emita de nuevo la llamada con ese nuevo LSN inicial y un LSN final válido, y a continuación se leerá el siguiente bloque de registros. El código SQLU_RLOG_READ_TO_CURRENT de sqlca significa que el lector de archivos de anotaciones ha leído hasta el final de los archivos de anotaciones disponibles.

Cuando ya no sea necesario utilizar la API, utilice db2ReadLogNoConnTerm para terminar la memoria.

Esta API lee datos de las anotaciones cronológicas de DB2. En dichas anotaciones cronológicas no se impone el control de acceso basado en etiquetas (Label-based access control - LBAC). Por lo tanto, una aplicación que llame a esta API puede potencialmente obtener acceso a los datos de tablas si el llamador tiene autorización suficiente para llamar a la API y puede comprender el formato de los registros de anotaciones cronológicas.

db2ReadLogNoConnInit - Inicializar la lectura de las anotaciones cronológicas de la base de datos sin una conexión de base de datos

Asigna la memoria que db2ReadLogNoConn va a utilizar para extraer registros de anotaciones cronológicas de las anotaciones cronológicas de la base de datos DB2 y consultar el Gestor de anotaciones cronológicas para obtener información de estado sobre la anotación cronológica actual.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2ReadLogNoConnInit (
       db2Uint32 versionNumber,
       void * pDB2ReadLogNoConnInitStruct,
       struct sqlca * pSqlca);
```

```
typedef SQL_STRUCTURE db2ReadLogNoConnInitStruct
{
   db2Uint32 iFilterOption;
   char *piLogFilePath;
   char *piOverflowLogPath;
   db2Uint32 iRetrieveLogs;
   char *piDatabaseName;
   char *piNodeName;
   db2Uint32 iReadLogMemoryLimit;
   char                                **poReadLogMemPtr;
} db2ReadLogNoConnInitStruct;
```

Parámetros de la API db2ReadLogNoConnInit

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segunda parámetro pDB2ReadLogNoConnInitStruct.

pDB2ReadLogNoConnInitStruct
Entrada. Puntero a la estructura db2ReadLogNoConnInitStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2ReadLogNoConnInitStruct

iFilterOption
Entrada. Especifica el nivel de filtrado para registros de anotaciones que se debe utilizar al leer esos registros. Los valores válidos son:

DB2READLOG_FILTER_OFF
Leer todos los registros de anotaciones comprendidos dentro del rango de LSN especificado.

DB2READLOG_FILTER_ON
Sólo lee los registros de anotaciones cronológicas en el rango de LSN determinado que estén marcados como propagables. Comportamiento habitual de la API de lectura asíncrona de archivos de anotaciones.

piLogFilePath
Entrada. Vía de acceso donde residen los archivos de anotaciones que se deben leer.

piOverflowLogPath
Entrada. Vía de acceso alternativa donde pueden residir los archivos de anotaciones que se deben leer.

iRetrieveLogs
Entrada. Opción que especifica si se debe invocar userexit para recuperar archivos de anotaciones que no se pueden encontrar en la vía de acceso de archivos de anotaciones ni en la vía de acceso de archivos de anotaciones de desbordamiento. Los valores válidos son:

DB2READLOG_RETRIEVE_OFF
No se debe invocar la salida de usuario para recuperar los archivos de anotaciones que faltan.

DB2READLOG_RETRIEVE_LOGPATH
Se debe invocar la salida de usuario para recuperar los archivos de anotaciones que faltan y colocarlos en la vía de acceso especificada de archivos de anotaciones.

DB2READLOG_RETRIEVE_OVERFLOW
Se debe invocar la salida de usuario para recuperar los archivos de anotaciones que faltan y colocarlos en la vía de acceso especificada de archivos de anotaciones de desbordamiento.

piDatabaseName
Entrada. Nombre de la base de datos a la que pertenecen los archivos de anotaciones de recuperación que se están leyendo. Este parámetro es necesario si se especifica la opción de recuperación indicada anteriormente. Este parámetro es necesario si se especifica la opción de recuperación anterior.

piNodeName
Entrada. Nombre del nodo al que pertenecen los archivos de anotaciones de recuperación que se están leyendo. Este parámetro es necesario si se especifica la opción de recuperación anterior.

iReadLogMemoryLimit
Entrada. Número máximo de bytes que la API puede asignar internamente.

poReadLogMemPtr
Salida. Bloque asignado por la API de memoria de tamaño iReadLogMemoryLimit. Esta memoria contiene datos permanente que la API requiere en cada invocación. Este bloque de memoria no debe reasignarse ni modificarse de ningún modo por parte del llamador.

Notas de uso

La memoria inicializada por db2ReadLogNoConnInit no se debe alterar.

Cuando ya no se utilice db2ReadLogNoConn, invoque db2ReadLogNoConnTerm para desasignar la memoria inicializada por db2ReadLogNoConnInit.

db2ReadLogNoConnTerm - Terminar la lectura de las anotaciones cronológicas de la base de datos sin una conexión de base de datos

Desasigna la memoria utilizada por la API db2ReadLogNoConn e inicializada originalmente por la API db2ReadLogNoConnInit.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2ReadLogNoConnTerm (
       db2Uint32 versionNumber,
       void * pDB2ReadLogNoConnTermStruct,
       struct sqlca * pSqlca);
```

```
typedef SQL_STRUCTURE db2ReadLogNoConnTermStruct
{
   char                                **poReadLogMemPtr;
} db2ReadLogNoConnTermStruct;
```

Parámetros de la API db2ReadLogNoConnTerm

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pDB2ReadLogNoConnTermStruct.

pDB2ReadLogNoConnTermStruct
Entrada. Puntero a la estructura db2ReadLogNoConnTermStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2ReadLogNoConnTermStruct

poReadLogMemPtr
Salida. Puntero al bloque de memoria asignado en la llamada de inicialización. Este puntero se libera y se establece en NULL.

---

db2Recover - Restaurar y avanzar una base de datos

Restaura y avanza una base de datos hasta un determinado punto del tiempo o al final de las anotaciones cronológicas.

Ámbito

En un entorno de bases de datos particionadas, esta API sólo puede invocarse desde la partición de catálogo. Si no se ha especificado ningún servidor de particiones de base de datos, afecta a todos los servidores de particiones de base de datos que están listados en el archivo db2nodes.cfg. Si se especifica un punto en el tiempo, la API afecta a todas las particiones de base de datos.

Autorización

Para recuperar una base de datos existente, una de las siguientes:

- sysadm
- sysctrl
- sysmaint

Para recuperar una base de datos nueva, una de las siguientes:

- sysadm
- sysctrl

Conexión necesaria

Para recuperar una base de datos existente, se necesita una conexión de base de datos. Esta API establece automáticamente una conexión a la base de datos especificada y finalizará la conexión cuando la operación de recuperación finalice. Instancia y base de datos, para recuperar a una base de datos nueva. La conexión de instancia es necesaria para crear la base de datos.

Archivo de inclusión de la API

db2ApiDf.h

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2Recover (
       db2Uint32 versionNumber,
       void * pDB2RecovStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2RecoverStruct
{
  char *piSourceDBAlias;
  char *piUsername;
  char *piPassword;
  db2Uint32 iRecoverCallerAction;
  db2Uint32 iOptions;
  sqlint32 *poNumReplies;
  struct sqlurf_info *poNodeInfo;
  char *piStopTime;
  char *piOverflowLogPath;
  db2Uint32 iNumChngLgOvrflw;
  struct sqlurf_newlogpath *piChngLogOvrflw;
  db2int32 iAllNodeFlag;
  db2int32 iNumNodes;
  SQL_PDB_NODE_TYPE *piNodeList;
  db2int32 iNumNodeInfo;
  char *piHistoryFile;
  db2Uint32 iNumChngHistoryFile;
  struct sqlu_histFile *piChngHistoryFile;
  char *piComprLibrary;
  void *piComprOptions;
  db2Uint32 iComprOptionsSize;
} db2RecoverStruct;

SQL_STRUCTURE sqlu_histFile
{
   SQL_PDB_NODE_TYPE nodeNum;
   unsigned short  filenameLen;
   char filename[SQL_FILENAME_SZ+1];
};

SQL_API_RC SQL_API_FN
  db2gRecover (
       db2Uint32 versionNumber,
       void * pDB2gRecoverStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gRecoverStruct
{
  char *piSourceDBAlias;
  db2Uint32 iSourceDBAliasLen;
  char *piUserName;
  db2Uint32 iUserNameLen;
  char *piPassword;
  db2Uint32 iPasswordLen;
  db2Uint32 iRecoverCallerAction;
  db2Uint32 iOptions;
  sqlint32 *poNumReplies;
  struct sqlurf_info *poNodeInfo;
  char *piStopTime;
  db2Uint32 iStopTimeLen;
  char *piOverflowLogPath;
  db2Uint32 iOverflowLogPathLen;
  db2Uint32 iNumChngLgOvrflw;
  struct sqlurf_newlogpath *piChngLogOvrflw;
```

```
  db2int32 iAllNodeFlag;
  db2int32 iNumNodes;
  SQL_PDB_NODE_TYPE *piNodeList;
  db2int32 iNumNodeInfo;
  char *piHistoryFile;
  db2Uint32 iHistoryFileLen;
  db2Uint32 iNumChngHistoryFile;
  struct sqlu_histFile *piChngHistoryFile;
  char *piComprLibrary;
  db2Uint32 iComprLibraryLen;
  void *piComprOptions;
  db2Uint32 iComprOptionsSize;
} db2gRecoverStruct;
```

Parámetros de la API db2Recover

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro pDB2RecoverStruct.

pDB2RecoverStruct
Entrada. Puntero a la estructura db2RecoverStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2RecoverStruct

piSourceDBAlias
Entrada. Serie que contiene el alias de base de datos que debe recuperarse.

piUserName
Entrada. Serie que contiene el nombre de usuario que se debe utilizar al intentar establecer una conexión. Puede ser NULL.

piPassword
Entrada. Serie que contiene la contraseña que se debe utilizar con el nombre de usuario. Puede ser NULL.

iRecoverCallerAction
Entrada. Los valores válidos son:

DB2RECOVER
Inicia la operación de recuperación. Especifica que la recuperación se ejecutará de forma desatendida y que en las situaciones que normalmente requieran la intervención del usuario se intentarán sin primero devolver el control al llamador o bien se producirá un error. Utilice esta acción de llamador, por ejemplo, si se sabe que se han montado todos los soportes de almacenamiento necesarios para la recuperación y no se desea que aparezcan mensajes de solicitud de programas de utilidad.

DB2RECOVER_RESTART
Permite el usuario ignorar una recuperación previa y volver a empezar desde el principio.

DB2RECOVER_CONTINUE
Continuar utilizando el dispositivo que ha generado el mensaje de aviso (por ejemplo, cuando se ha montado una cinta nueva).

DB2RECOVER_LOADREC_TERM
Terminar todos los dispositivos utilizados por la recuperación de carga.

DB2RECOVER_DEVICE_TERM
Dejar de utilizar el dispositivo que ha generado el mensaje de aviso (por ejemplo, cuando no hay más cintas).

DB2RECOVER_PARM_CHK_ONLY
Se utiliza para validar parámetros sin realizar una operación de recuperación. Antes de que concluya esta llamada, se interrumpe la conexión con la base de datos establecida por esta llamada y no es necesaria ninguna llamada más.

DB2RECOVER_DEVICE_TERMINATE
Elimina un determinado dispositivo de la lista de dispositivos utilizados por la operación de recuperación. Cuando un determinado soporte de almacenamiento está lleno, la recuperación devuelve un aviso al llamador. Invoque de nuevo el programa de utilidad de recuperación con esta acción de llamador para eliminar de la lista de dispositivos utilizados el dispositivo que produjo el aviso.

iOptions
Entrada. Los valores válidos son:

- DB2RECOVER_EMPTY_FLAG
No se especifican distintivos.

- DB2RECOVER_LOCAL_TIME
Indica que el valor especificado para la hora de parada por piStopTime está en hora local, no GMT. Este es el valor por omisión.

- DB2RECOVER_GMT_TIME
Este distintivo indica que el valor especificado para la hora de parada por piStopTime está en GMT (Hora Media de Greeenwich).

poNumReplies
Salida. Número de respuestas recibidas.

poNodeInfo
Salida. Información de respuesta de particiones de base de datos.

piStopTime
Entrada. Serie de caracteres que contiene una indicación fecha y hora en formato ISO. La recuperación de la base de datos se detendrá cuando se sobrepase esta indicación de fecha y hora. Especifique SQLUM_INFINITY_TIMESTAMP para avanzar todo lo posible. Puede ser NULL para DB2ROLLFORWARD_QUERY, DB2ROLLFORWARD_PARM_CHECK, y cualquiera de las acciones de llamador (DB2ROLLFORWARD_LOADREC_) de recuperación de carga.

piOverflowLogPath
Entrada. Este parámetro se utiliza para especificar la vía de acceso de anotaciones cronológicas alternativa que se debe utilizar. Además de los archivos de anotaciones cronológicas activos, los archivos de anotaciones cronológicas archivados deben moverse (por parte del usuario) a la ubicación especificada por el parámetro de configuración de vía de acceso de anotaciones para que este programa de utilidad pueda utilizarlos. Esto puede suponer un problema si el usuario no tiene suficiente espacio en la vía de acceso de anotaciones. Por este motivo se proporciona la vía de acceso de anotaciones cronológicas de desbordamiento. Durante la recuperación en avance, se busca en los archivos de anotaciones cronológicas necesarios, primero en la vía de acceso de anotaciones y

después en la vía de acceso de anotaciones de desbordamiento. Los archivos de anotaciones cronológicas necesarios para la recuperación en avance de espacios de tablas pueden llevarse a la vía de acceso de anotaciones o a la vía de acceso de anotaciones de desbordamiento. Si el llamador no especifica una vía de acceso de anotaciones de desbordamiento, el valor por omisión es la vía de acceso de anotaciones.

En un entorno de bases de datos particionadas, la vía de acceso de anotaciones cronológicas de desbordamiento debe ser una vía de acceso válida totalmente calificada; la vía de acceso por omisión es la vía de acceso de anotaciones cronológicas de desbordamiento por omisión para cada partición de base de datos. En un entorno de bases de datos de una sola partición, la vía de acceso de anotaciones cronológicas de desbordamiento puede ser relativa si el servidor es local.

iNumChngLgOvrflw
Entrada. Entornos de bases de datos particionadas solamente. El número de vías de acceso de anotaciones cronológicas de desbordamiento cambiadas. Estas nuevas vías de acceso de anotaciones cronológicas alteran temporalmente la vía de acceso de anotaciones cronológicas de desbordamiento por omisión sólo para el servidor de particiones de base de datos especificado.

piChngLogOvrflw
Entrada. Entornos de bases de datos particionadas solamente. Un puntero a una estructura que contiene los nombres totalmente calificados de vías de acceso de anotaciones cronológicas de desbordamiento cambiadas. Estas nuevas vías de acceso de anotaciones cronológicas alteran temporalmente la vía de acceso de anotaciones cronológicas de desbordamiento por omisión sólo para el servidor de particiones de base de datos especificado.

iAllNodeFlag
Entrada. Entornos de bases de datos particionadas solamente. Indica si la operación de avance se debe aplicar a todos los servidores de particiones de base de datos definidos en db2nodes.cfg. Los valores válidos son:

DB2_NODE_LIST
Aplicar a servidores de particiones de base de datos de una lista que se pasa en piNodeList.

DB2_ALL_NODES
Aplicar a todos los servidores de particiones de base de datos. piNodeList debe ser NULL. Es el valor por omisión.

DB2_ALL_EXCEPT
Aplicar a todos los servidores de particiones de base de datos excepto a los de una lista que se pasa en piNodeList.

DB2_CAT_NODE_ONLY
Aplicar sólo a la partición de catálogo. piNodeList debe ser NULL.

iNumNodes
Entrada. Especifica el número de servidores de particiones de base de datos de la matriz piNodeList.

piNodeList
Entrada. Puntero a una matriz de números de servidor de partición de base de datos en los que realizar la recuperación en avance.

iNumNodeInfo
Entrada. Define el tamaño del parámetro de salida poNodeInfo, que debe

ser lo suficientemente grande como para contener información de estado de cada partición de base de datos que se avanza. En un entorno de bases de datos de una sola partición, este parámetro debe establecerse en 1. El valor de este parámetro debe ser el mismo que el número de servidores de particiones de base de datos para los que se llama a esta API.

piHistoryFile
Archivo histórico.

iNumChngHistoryFile
Número de archivos históricos en lista.

piChngHistoryFile
Lista de archivos históricos.

piComprLibrary
Entrada. Indica el nombre de la biblioteca externa a utilizar para realizar la descompresión de la imagen de copia de seguridad, si la imagen está comprimida. El nombre debe ser una vía de acceso totalmente calificada que haga referencia a un archivo del servidor. Si el valor es un puntero nulo o un puntero a una serie vacía, DB2 intenta utilizar la biblioteca almacenada en la imagen. Si la copia de seguridad no se ha comprimido, el valor de este parámetro se pasará por alto. Si no se encuentra la biblioteca especificada, la restauración fallará.

piComprOptions
Entrada. Describe un bloque de datos binarios que se pasará a la rutina de inicialización en la biblioteca de descompresión. DB2 pasará esta serie directamente del cliente al servidor, de modo que los posibles problemas de inversión de bytes o de conversión de página de códigos los deberá manejar la biblioteca de compresión. Si el primer carácter del bloque de datos es '@', DB2 interpretará los datos restantes como el nombre de un archivo que se encuentra en el servidor. DB2 sustituirá entonces el contenido de piComprOptions y iComprOptionsSize por el contenido y tamaño de este archivo respectivamente y pasará estos nuevos valores a la rutina de inicialización.

iComprOptionsSize
Entrada. Representa el tamaño del bloque de datos pasado como piComprOptions. iComprOptionsSize será cero solamente si piComprOptions es un puntero nulo.

Parámetros de la estructura de datos sqlu_histFile

nodeNum
Entrada. Especifica para qué partición de base de datos deberá utilizarse esta entrada.

filenameLen
Entrada. Longitud del nombre de archivo en bytes.

filename
Entrada. Vía de acceso al archivo histórico para esta partición de base de datos. La vía de acceso debe finalizar con una barra inclinada.

Parámetros específicos de la estructura de datos db2gRecoverStruct

iSourceDBAliasLen
Especifica la longitud, en bytes, del parámetro piSourceDBAlias.

iUserNameLen
Especifica la longitud, en bytes, del parámetro piUsername.

iPasswordLen
Especifica la longitud, en bytes, del parámetro piPassword.

iStopTimeLen
Especifica la longitud, en bytes, del parámetro piStopTime.

iOverflowLogPathLen
Especifica la longitud, en bytes, del parámetro piOverflowLogPath.

iHistoryFileLen
Especifica la longitud, en bytes, del parámetro piHistoryFile.

iComprLibraryLen
Entrada. Especifica la longitud, en bytes, del nombre de la biblioteca especificada en el parámetro piComprLibrary. El valor se establece en cero si no se proporciona ningún nombre de biblioteca.

db2Reorg - Reorganizar un índice o una tabla

Reorganiza una tabla o todos los índices definidos en una tabla compactando la información y reconstruyendo los datos de filas o índice para eliminar los datos fragmentados.

Autorización

Una de las siguientes:

- SYSADM
- SYSCTRL
- SYSMAINT
- DBADM
- Privilegio CONTROL sobre la tabla

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2Reorg (
             db2Uint32 versionNumber,
             void * pReorgStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ReorgStruct
{
  db2Uint32 reorgType;
  db2Uint32 reorgFlags;
  db2int32 nodeListFlag;
  db2Uint32 numNodes;
  SQL_PDB_NODE_TYPE *pNodeList;
  union db2ReorgObject        reorgObject;
} db2ReorgStruct;
```

```
union db2ReorgObject
{
  struct db2ReorgTable             tableStruct;
  struct db2ReorgIndexesAll        indexesAllStruct;
};

typedef SQL_STRUCTURE db2ReorgTable
{
  char *pTableName;
  char *pOrderByIndex;
  char *pSysTempSpace;
  char *pLongTempSpace;
} db2ReorgTable;

typedef SQL_STRUCTURE db2ReorgIndexesAll
{
  char *pTableName;
  char *pIndexName;
} db2ReorgIndexesAll;

SQL_API_RC SQL_API_FN
  db2gReorg (
              db2Uint32 versionNumber,
              void * pReorgStruct,
              struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gReorgStruct
{
  db2Uint32 reorgType;
  db2Uint32 reorgFlags;
  db2int32 nodeListFlag;
  db2Uint32 numNodes;
  SQL_PDB_NODE_TYPE *pNodeList;
  union db2gReorgObject       reorgObject;
} db2gReorgStruct;

typedef SQL_STRUCTURE db2gReorgNodes
{
  SQL_PDB_NODE_TYPE nodeNum[SQL_PDB_MAX_NUM_NODE];
} db2gReorgNodes;

union db2gReorgObject
{
  struct db2gReorgTable            tableStruct;
  struct db2gReorgIndexesAll       indexesAllStruct;
};

typedef SQL_STRUCTURE db2gReorgTable
{
  db2Uint32 tableNameLen;
  char *pTableName;
  db2Uint32 orderByIndexLen;
  char *pOrderByIndex;
  db2Uint32 sysTempSpaceLen;
  char *pSysTempSpace;
  db2Uint32 longTempSpaceLen;
  char *pLongTempSpace;
} db2gReorgTable;

typedef SQL_STRUCTURE db2gReorgIndexesAll
{
  db2Uint32 tableNameLen;
  char *pTableName;
  db2Uint32 indexNameLen;
  char *pIndexName;
} db2gReorgIndexesAll;
```

Parámetros de la API db2Reorg

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro, **pReorgStruct**.

pReorgStruct
Entrada. Puntero a la estructura db2ReorgStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2ReorgStruct

reorgType
Entrada. Especifica el tipo de reorganización. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio include) son:

DB2REORG_OBJ_TABLE_OFFLINE
Reorganizar la tabla fuera de línea.

DB2REORG_OBJ_TABLE_INPLACE
Reorganizar la tabla in situ.

DB2REORG_OBJ_INDEXESALL
Reorganizar todos los índices.

DB2REORG_OBJ_INDEX
Reorganizar un índice.

reorgFlags
Entrada. Opciones de reorganización. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio include) son:

DB2REORG_OPTION_NONE
Acción por omisión.

DB2REORG_LONGLOB
Reorganizar campos long y lob, utilizado cuando se especifica DB2REORG_OBJ_TABLE_OFFLINE como **reorgType**.

DB2REORG_INDEXSCAN
Volver a organizar clústeres utilizando la exploración de índices, utilizado cuando se especifica DB2REORG_OBJ_TABLE_OFFLINE como **reorgType**.

DB2REORG_START_ONLINE
Iniciar la reorganización en línea, utilizado cuando se especifica DB2REORG_OBJ_TABLE_INPLACE como **reorgType**.

DB2REORG_PAUSE_ONLINE
Pausar una reorganización en línea, utilizado cuando se especifica DB2REORG_OBJ_TABLE_INPLACE como **reorgType**.

DB2REORG_STOP_ONLINE
Detener una reorganización en línea, utilizado cuando se especifica DB2REORG_OBJ_TABLE_INPLACE como **reorgType**.

DB2REORG_RESUME_ONLINE
Reanudar una reorganización en línea que está en pausa, utilizado cuando se especifica DB2REORG_OBJ_TABLE_INPLACE como **reorgType**.

DB2REORG_NOTRUNCATE_ONLINE
No realizar truncamiento de tabla, utilizado cuando se especifica DB2REORG_OBJ_TABLE_INPLACE como **reorgType**.

DB2REORG_ALLOW_NONE
Sin acceso de lectura o escritura a la tabla. Este parámetro no está soportado cuando se especifica DB2REORG_OBJ_TABLE_INPLACE como **reorgType**.

DB2REORG_ALLOW_WRITE
Permitir acceso de lectura y escritura a la tabla. Este parámetro no está soportado cuando se especifica DB2REORG_OBJ_TABLE_OFFLINE como **reorgType**.

DB2REORG_ALLOW_READ
Permitir sólo el acceso de lectura a la tabla.

DB2REORG_CLEANUP_NONE
No es necesaria limpieza, utilizado cuando se especifica DB2REORG_OBJ_INDEXESALL o DB2REORG_OBJ_INDEX como **reorgType**.

DB2REORG_CLEANUP_ALL
Limpiar las claves pseudosuprimidas confirmadas y las páginas pseudovacías confirmadas, utilizado cuando se especifica DB2REORG_OBJ_INDEXESALL o DB2REORG_OBJ_INDEX como **reorgType**.

DB2REORG_CLEANUP_PAGES
Limpiar las páginas pseudovacías confirmadas solamente, pero no limpiar las claves pseudosuprimidas confirmadas de páginas que no están pseudovacías, utilizado cuando se especifica DB2REORG_OBJ_INDEXESALL o DB2REORG_OBJ_INDEX como **reorgType**.

DB2REORG_CONVERT_NONE
No es necesaria la conversión, utilizado cuando se especifica DB2REORG_OBJ_INDEXESALL o DB2REORG_OBJ_INDEX como **reorgType**.

DB2REORG_CONVERT
Convertir a índice de tipo 2, utilizado cuando se especifica DB2REORG_OBJ_INDEXESALL como **reorgType**.

DB2REORG_RESET_DICTIONARY
Si el atributo COMPRESS para la tabla es YES, se crea un nuevo diccionario de compresión de filas. Todas las filas procesadas durante la reorganización están sujetas a compresión utilizando este diccionario nuevo. Este diccionario sustituye a cualquier otro diccionario anterior. Si el atributo COMPRESS para la tabla es NO y la tabla tiene un diccionario de compresión existente, el proceso de reorganización eliminará el diccionario y todas las filas de la tabla reorganizada estarán en formato no comprimido. Este parámetro solamente está soportado para el **reorgType** DB2REORG_OBJ_TABLE_OFFLINE.

DB2REORG_KEEP_DICTIONARY
Si el atributo COMPRESS para la tabla es YES y existe un diccionario, se conserva. Si el atributo COMPRESS para la tabla es YES y no existe un diccionario, se crea uno, ya que en ese caso la opción toma por omisión el valor

DB2REORG_RESET_DICTIONARY. Todas las filas procesadas por la reorganización están sujetas a compresión. Si el atributo COMPRESS para la tabla es NO, se mantendrá el diccionario (si existía uno) y todas las filas de tabla reorganizada estarán en formato no comprimido. Este parámetro solamente está soportado para el **reorgType** DB2REORG_OBJ_TABLE_OFFLINE.

nodeListFlag
Entrada. Especifica qué nodos se deben reorganizar. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio include) son:

DB2REORG_NODE_LIST
Someter a todos los nodos de la matriz de lista de nodos.

DB2REORG_ALL_NODES
Someter a todos los nodos del grupo de particiones de base de datos.

DB2REORG_ALL_EXCEPT
Someter a todos los nodos excepto los especificados por el parámetro de lista de nodos.

numNodes
Entrada. Número de nodos de la matriz de lista de nodos.

pNodeList
Puntero a la matriz de números de nodo.

reorgObject
Entrada. Especifica el tipo de objeto que debe reorganizarse.

Parámetros de unión de db2ReorgObject

tableStruct
Especifica las opciones para una reorganización de tabla.

indexesAllStruct
Especifica las opciones para una reorganización de índice.

Parámetros de la estructura de datos db2ReorgTable

pTableName
Entrada. Especifica el nombre de la tabla que debe reorganizarse.

pOrderByIndex
Entrada. Especifica el índice según el que debe ordenarse la tabla.

pSysTempSpace
Entrada. Especifica el espacio de tablas temporal del sistema en el que se crean los objetos temporales. El mandato REORG puede expandir filas en los casos en que se añade una columna a una tabla (es decir, desde ALTER TABLE ADD COLUMN) y las filas se insertaron antes de añadirse la columna. Para una tabla no particionada, este parámetro debe especificar un espacio de tablas con suficiente espacio para crear el nuevo objeto de tabla. Las tablas particionadas se reorganizan de partición de datos en partición de datos. En este caso, debe haber suficiente espacio libre en la tabla para que contenga la mayor partición de datos de la tabla.

Si no se especifica este parámetro para una tabla no particionada, se utiliza el espacio de tablas en el que reside la tabla. Si no se especifica este parámetro para una tabla particionada, el espacio de tablas en que está

ubicada cada partición de datos se utiliza para el almacenamiento temporal de esa partición de datos. En el espacio de tablas de cada partición de datos debe haber suficiente espacio libre para que quepa una copia de la partición de datos.

pLongTempSpace
Entrada. Especifica el espacio de tablas temporal para crear objetos long (columnas LONG VARCHAR y LOB) durante la reorganización de tabla. Si no se especifica el parámetro **pSysTempSpace**, no se tiene en cuenta este parámetro. Si no se especifica este parámetro, pero se ha especificado el parámetro **pSysTempSpace**, DB2 creará los objetos de datos long en el espacio de tablas especificado por el parámetro **pSysTempSpace**, a menos que los tamaños de página difieran.

Cuando los tamaños de página difieren, si se especifica **pSysTempSpace** pero no este parámetro, DB2 intentará buscar un espacio de tablas existente con un tamaño de página correspondiente en el que crear los objetos long.

Parámetros de la estructura de datos db2ReorgIndexesAll

pTableName
Entrada. Especifica el nombre de la tabla para la reorganización de índices. Si se especifica DB2REORG_OBJ_INDEX como **reorgType**, no es necesario el parámetro **pTableName** y puede ser NULL. No obstante, si se especifica el parámetro **pTableName**, debe ser la tabla en la que se define el índice.

pIndexName
Entrada. Especifica el nombre del índice que debe reorganizarse. Este parámetro se utiliza solamente cuando el parámetro **reorgType** está establecido en un valor de DB2REORG_OBJ_INDEX, de lo contrario establezca el parámetro **pIndexName** en NULL.

Parámetros específicos de la estructura de datos db2gReorgTable

tableNameLen
Entrada. Especifica la longitud, en bytes, de **pTableName**.

orderByIndexLen
Entrada. Especifica la longitud, en bytes, de **pOrderByIndex**.

sysTempSpaceLen
Entrada. Especifica la longitud, en bytes, de **pSysTempSpace**.

longTempSpaceLen
Entrada. Especifica la longitud del nombre almacenado en el parámetro **pLongTempSpace**.

Parámetros específicos de la estructura de datos db2gReorgIndexesAll

tableNameLen
Entrada. Especifica la longitud, en bytes, de **pTableName**.

indexNameLen
Entrada. Especifica la longitud, en bytes, del parámetro **pIndexName**.

Notas de uso

- El rendimiento del acceso a tabla, la exploración de índices y la eficacia de la captación previa de páginas de índice pueden resultar afectados negativamente cuando se han modificado los datos de la tabla muchas veces, fragmentándose y

deshaciendo los clústeres. Utilice REORGCHK para determinar si una tabla o sus índices son candidatos a la reorganización. Todo el trabajo se confirmará y todos los cursores abiertos se cerrarán durante el proceso de roeganización. Después de reorganizar una tabla o sus índices, utilice db2Runstats para actualizar las estadísticas y sqlarbnd para volver a vincular los paquetes que utilizan esta tabla.

- Si la tabla está distribuida entre varios nodos y la reorganización falla en cualquiera de los nodos afectados, solamente se retrotraerá la reorganización de los nodos anómalos. Si la reorganización de la tabla no es satisfactoria, no se deberán suprimir los archivos temporales. El gestor de bases de datos utiliza estos archivos para recuperar la base de datos.
- Para la reorganización de tabla, si se especifica el nombre de un índice, el gestor de bases de datos reorganiza los datos de acuerdo con el orden del índice. Para maximizar el rendimiento, especifique un índice que se utilice con frecuencia en consultas de SQL. Si no se especifica el nombre de un índice y existe un índice de clúster, los datos se ordenarán de acuerdo al índice de clúster.
- El valor PCTFREE de una tabla determina la cantidad de espacio libre designado por página. Si no se ha establecido el valor, el programa de utilidad llenará tanto espacio como sea posible en cada página.
- Para realizar una recuperación en avance del espacio de tablas después de una reorganización de tabla, se deberán habilitar tanto los espacios de tablas LONG como los espacios de datos para la recuperación en avance.
- Si la tabla contiene columnas LOB no definidas con la opción COMPACT, el objeto de almacenamiento LOB DATA puede ser significativamente mayor después de la reorganización de la tabla. Esto puede ser el resultado del orden en el que se han reorganizado las filas y los tipos de espacios de tablas utilizados (SMS/DMS).
- En la tabla siguiente se muestra el acceso a la tabla por omisión elegido basándose en el tipo de reorganización y de tabla:

Tabla 8. Acceso a tabla por omisión elegido basándose en el tipo de reorganización y tabla

Tipo de reorganización y distintivo aplicables que pueden afectar al acceso a tabla por omisión		**Modalidad de acceso elegida para cada tipo de tabla**	
reorgType	**reorgFlags (si corresponde)**	**Tabla no particionada**	**Tabla particionada**
DB2REORG_OBJ_TABLE_ OFFLINE		DB2REORG_ALLOW_ READ	DB2REORG_ALLOW_ NONE
DB2REORG_OBJ_TABLE_ INPLACE		DB2REORG_ALLOW_ WRITE	N/D
DB2REORG_OBJ_ INDEXESALL		DB2REORG_ALLOW_ READ	DB2REORG_ALLOW_ NONE
DB2REORG_OBJ_ INDEXESALL	DB2REORG_CLEANUP_ ALL, DB2REORG_CLEANUP_ PAGES	DB2REORG_ALLOW_ READ	DB2REORG_ALLOW_ READ
DB2REORG_OBJ_INDEX		N/D	DB2REORG_ALLOW_ READ
DB2REORG_OBJ_INDEX	DB2REORG_CLEANUP_ ALL, DB2REORG_CLEANUP_ PAGES	N/D	DB2REORG_ALLOW_ READ

N/A: No aplicable en este momento ya que no está soportado.

Nota: Es posible que algunas modalidades de acceso no estén soportadas en determinados tipos de tablas o índices. En estos casos, y siempre que es posible, se utiliza la modalidad de acceso menos restrictiva. (La modalidad de acceso más restrictiva es DB2REORG_ALLOW_NONE, seguida de DB2REORG_ALLOW_READ y después DB2REORG_ALLOW_WRITE, que es la menos restrictiva). Como soporte para el cambio de tipos de tabla o índice existentes, o si se proporcionan nuevos tipos de tabla o índice, el valor por

omisión puede cambiar de una modalidad de acceso más restrictiva una modalidad menosrestrictiva. La modalidad menos restrictiva elegida para el valor por omisión no irá más allá de DB2REORG_ALLOW_READ cuando **reorgType** no es DB2REORG_OBJ_TABLE_INPLACE. La modalidad de acceso por omisión se elige cuando no se especifica ninguno de los distintivos DB2REORG_ALLOW_NONE, DB2REORG_ALLOW_READ o DB2REORG_ALLOW_WRITE.

- Al reorganizar índices, utilice la opción de acceso para permitir a otras transacciones acceso solo de lectura o de lectura/grabación a la tabla. Existe un breve período de bloqueo cuando se poniendo disponibles los índices reorganizados durante el que no se permite ningún acceso a la tabla.
- Si una reorganización de índice con acceso de lectura o de escritura falla debido a que deben reconstruirse los índices, se cambiará la reorganización para no permitir ningún acceso y continuar. Se escribirá un mensaje en las anotaciones de notificación de administración y en las anotaciones de diagnóstico para avisarle sobre el cambio en la modalidad de acceso. Para la reorganización de índice de una tabla particionada, los índices que deban reconstruirse se reconstruirán fuera de línea y, a continuación, se reorganizará el índice que haya especificado, suponiendo que no estuviera ya reconstruido. Esta reorganización utiliza la modalidad de acceso que haya especificado personalmente. Se escribirá un mensaje en las anotaciones de notificación de administración y en las anotaciones de diagnóstico para avisarle de que los índices se reconstruyen fuera de línea.
- Para la reorganización de tablas que no sea in situ, si no se especifica DB2REORG_RESET_DICTIONARY ni DB2REORG_KEEP_DICTIONARY, el valor por omisión es DB2REORG_KEEP_DICTIONARY.
- Si una reorganización de índice sin acceso falla, alguno de los índices o todos ellos no estarán disponibles y se reconstruirán en el siguiente acceso a la tabla.
- Esta API no puede utilizarse con:
  - vistas o un índice basado en una extensión de índice
  - una tabla DMS mientras se esté realizando una copia de seguridad en línea de un espacio de tablas en el que resida la tabla
  - tablas temporales declaradas

db2ResetAlertCfg - Restablecer la configuración de alertas de los indicadores de salud

Restablece los valores del indicador de salud para objetos específicos en los valores por omisión actuales para ese tipo de objeto o restablece los valores por omisión actuales del indicador de salud para un tipo de objeto a los valores por omisión de instalación.

Autorización

Una de las siguientes:

- sysadm
- sysmaint
- sysctrl

Conexión necesaria

Instancia. Si no existe ninguna conexión de instancia, se creará una conexión de instancia por omisión.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2ResetAlertCfg (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ResetAlertCfgData
{
   db2Uint32 iObjType;
   char *piObjName;
   char *piDbName;
   db2Uint32 iIndicatorID;
} db2ResetAlertCfgData;
```

Parámetros de la API db2ResetAlertCfg

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2ResetAlertCfgData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2ResetAlertCfgData

iObjType
Entrada. Especifica el tipo de objeto para el que se debe restaurar la configuración. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

- DB2ALERTCFG_OBJTYPE_DBM
- DB2ALERTCFG_OBJTYPE_DATABASES
- DB2ALERTCFG_OBJTYPE_TABLESPACES
- DB2ALERTCFG_OBJTYPE_TS_CONTAINERS
- DB2ALERTCFG_OBJTYPE_DATABASE
- DB2ALERTCFG_OBJTYPE_TABLESPACE
- DB2ALERTCFG_OBJTYPE_TS_CONTAINER

piObjName
Entrada. Nombre del espacio de tablas o contenedor de espacio de tablas cuando el tipo de objeto, iObjType, es DB2ALERTCFG_OBJTYPE_TS_CONTAINER o DB2ALERTCFG_OBJTYPE_TABLESPACE. El nombre del contenedor del espacio de tablas se define como <IDnumérico-espaciotablas>.<nombre-contenedor-espaciotablas>.

piDbname
Entrada. Nombre de alias de la base de datos para la que se debe

restablecer la configuración cuando el tipo de objeto, iObjType, es DB2ALERTCFG_OBJTYPE_TS_CONTAINER, DB2ALERTCFG_OBJTYPE_TABLESPACE y DB2ALERTCFG_OBJTYPE_DATABASE.

iIndicatorID
Entrada. Indicador de salud para el que se deben aplicar los restablecimientos de la configuración.

Notas de uso

El valor por omisión actual del tipo de objeto se restaura cuando ObjType es DB2ALERTCFG_OBJTYPE_DBM, DB2ALERTCFG_OBJTYPE_DATABASES, DB2ALERTCFG_OBJTYPE_TABLESPACES, DB2ALERTCFG_OBJTYPE_TS_CONTAINERS o cuando piObjName y piDbName son ambos NULL. Si iObjType es DB2ALERTCFG_OBJTYPE_DATABASE, DB2ALERTCFG_OBJTYPE_TABLESPACE, DB2ALERTCFG_OBJTYPE_TS_CONTAINER y se especifica piDbName y piObjName (no necesario para la base de datos), entonces se restauran los valores actuales para ese objeto determinado.

db2ResetMonitor - Restaurar los datos del supervisor del sistema de base de datos

Restaura los datos del supervisor del sistema de base de datos de una base de datos especificada, o de todas las bases de datos activas, para la aplicación que emite la llamada.

Ámbito

Esta API puede afectar a una partición de base de datos determinada de la instancia o a todas las particiones de base de datos de la instancia.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint
- sysmon

Conexión necesaria

Instancia. Si no existe ninguna conexión de instancia, se creará una conexión de instancia por omisión.

Para restablecer los conmutadores de supervisor para una instancia remota (o una instancia local diferente), es necesario conectarse primero a dicha instancia.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2ResetMonitor (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ResetMonitorData
{
   db2Uint32 iResetAll;
   char *piDbAlias;
   db2Uint32 iVersion;
   db2int32 iNodeNumber;
} db2ResetMonitorData;

SQL_API_RC SQL_API_FN
  db2gResetMonitor (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gResetMonitorData
{
   db2Uint32 iResetAll;
   char *piDbAlias;
   db2Uint32 iDbAliasLength;
   db2Uint32 iVersion;
   db2int32 iNodeNumber;
} db2gResetMonitorData;
```

Parámetros de la API db2ResetMonitor

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2ResetMonitorData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2ResetMonitorData

iResetAll
Entrada. Distintivo de restauración.

piDbAlias
Entrada. Puntero al alias de la base de datos.

iVersion
Entrada. ID de versión de los datos del supervisor de bases de datos que se deben recopilar. El supervisor de bases de datos solamente devuelve datos que estaban disponibles para la versión solicitada. Establezca este parámetro en una de las constantes siguientes:

- SQLM_DBMON_VERSION1
- SQLM_DBMON_VERSION2
- SQLM_DBMON_VERSION5
- SQLM_DBMON_VERSION5_2
- SQLM_DBMON_VERSION6
- SQLM_DBMON_VERSION7
- SQLM_DBMON_VERSION8

- SQLM_DBMON_VERSION9
- SQLM_DBMON_VERSION9_5

Nota: Si se especifica SQLM_DBMON_VERSION1 como versión, las API no se pueden ejecutar remotamente.

Nota: Las constantes SQLM_DBMON_VERSION5_2 y anteriores han quedado obsoletas y es posible que se eliminen en un futuro release de DB2.

iNodeNumber
Entrada. El servidor de particiones de base de datos que debe enviarse la petición. De acuerdo con este valor, la petición se procesará para el servidor de particiones de base de datos actual, para todos los servidores de particiones de base de datos o para un servidor de particiones de base de datos especificado por el usuario. Los valores válidos son:

- SQLM_CURRENT_NODE
- SQLM_ALL_NODES
- valor de nodo

Nota: Para las instancias autónomas, se debe utilizar el valor SQLM_CURRENT_NODE.

Parámetros específicos de la estructura de datos db2gResetMonitorData

iDbAliasLength
Entrada. Especifica la longitud, en bytes, del parámetro piDbAlias.

Notas de uso

Cada proceso (conexión) tiene su propia vista privada de los datos de supervisor. Si un usuario restablece o desactiva un conmutador de supervisor, los demás usuarios no se ven afectados. Cuando una aplicación llama por primera vez a cualquier función de supervisor de base de datos, hereda los valores de conmutación por omisión del archivo de configuración del gestor de bases de datos. Estos valores pueden alterarse temporalmente con db2MonitorSwitches - Obtener/actualizar conmutadores de supervisor.

Si se restauran todas las bases de datos activas, también se restaura alguna información del gestor de bases de datos para mantener la coherencia de los datos devueltos.

Esta API no se puede utilizar para restaurar selectivamente determinados ítems de datos o grupos de supervisores. Pero se puede restaurar un grupo determinado desactivando su conmutador y luego activándolo de nuevo, mediante db2MonitorSwitches - Obtener/actualizar conmutadores de supervisor.

db2Restore - Restaurar una base de datos o un espacio de tablas

Vuelve a crear una base de datos dañada o corrompida de la que se ha hecho copia de seguridad utilizando la API db2Backup. La base de datos restaurada está en el mismo estado que estaba cuando se hizo la copia de seguridad. Este programa de utilidad también puede restaurar en una base de datos con un nombre diferente del nombre de base de datos de la imagen de copia de seguridad (además de

poder restaurar en una nueva base de datos), a excepción de una restauración instantánea donde el nombre de base de datos de la imagen de copia de seguridad debe ser el mismo.

Este programa de utilidad también se puede utilizar para restaurar bases de datos DB2 creadas en los dos releases anteriores.

Además, este programa de utilidad puede restaurar a partir de una copia de seguridad de nivel de espacios de tablas, o restaurar desde una imagen de copia de seguridad de base de datos.

Ámbito

Esta API sólo afecta a la partición de base de datos desde la que se invoca.

Autorización

Para restaurar a una base de datos existente, una de las siguientes:

- *sysadm*
- *sysctrl*
- *sysmaint*

Para restaurar a una base de datos nueva, una de las siguientes:

- *sysadm*
- *sysctrl*

Conexión necesaria

Base de datos, para restaurar a una base de datos existente. Esta API establece automáticamente una conexión a la base de datos especificada y finalizará la conexión cuando la operación de restauración finalice.

Instancia y *base de datos*, para restaurar a una base de datos nueva. La conexión de instancia es necesaria para crear la base de datos.

Para una restauración de instantánea, se necesitan conexiones de *instancia* y *base de datos*.

Para restaurar a una base de datos nueva en una instancia distinta de la actual, (tal como está definida por el valor de la variable de entorno **DB2INSTANCE**), es necesario conectarse primero a la instancia en la que residirá la base de datos nueva.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2Restore (
       db2Uint32 versionNumber,
       void * pDB2RestoreStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2RestoreStruct
{
  char *piSourceDBAlias;
```

```
  char *piTargetDBAlias;
  char oApplicationId[SQLU_APPLID_LEN+1];
  char *piTimestamp;
  char *piTargetDBPath;
  char *piReportFile;
  struct db2TablespaceStruct *piTablespaceList;
  struct db2MediaListStruct *piMediaList;
  char *piUsername;
  char *piPassword;
  char *piNewLogPath;
  void *piVendorOptions;
  db2Uint32 iVendorOptionsSize;
  db2Uint32 iParallelism;
  db2Uint32 iBufferSize;
  db2Uint32 iNumBuffers;
  db2Uint32 iCallerAction;
  db2Uint32 iOptions;
  char *piComprLibrary;
  void *piComprOptions;
  db2Uint32 iComprOptionsSize;
  char *piLogTarget;
  struct db2StoragePathsStruct *piStoragePaths;
  char *piRedirectScript;
} db2RestoreStruct;

typedef SQL_STRUCTURE db2TablespaceStruct
{
  char                         **tablespaces;
  db2Uint32 numTablespaces;
} db2TablespaceStruct;

typedef SQL_STRUCTURE db2MediaListStruct
{
  char                         **locations;
  db2Uint32 numLocations;
  char locationType;
} db2MediaListStruct;

typedef SQL_STRUCTURE db2StoragePathsStruct
{
  char                         **storagePaths;
  db2Uint32 numStoragePaths;
} db2StoragePathsStruct;

SQL_API_RC SQL_API_FN
  db2gRestore (
       db2Uint32 versionNumber,
       void * pDB2gRestoreStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gRestoreStruct
{
  char *piSourceDBAlias;
  db2Uint32 iSourceDBAliasLen;
  char *piTargetDBAlias;
  db2Uint32 iTargetDBAliasLen;
  char *poApplicationId;
  db2Uint32 iApplicationIdLen;
  char *piTimestamp;
  db2Uint32 iTimestampLen;
  char *piTargetDBPath;
  db2Uint32 iTargetDBPathLen;
  char *piReportFile;
  db2Uint32 iReportFileLen;
  struct db2gTablespaceStruct *piTablespaceList;
  struct db2gMediaListStruct *piMediaList;
  char *piUsername;
```

```
  db2Uint32 iUsernameLen;
  char *piPassword;
  db2Uint32 iPasswordLen;
  char *piNewLogPath;
  db2Uint32 iNewLogPathLen;
  void *piVendorOptions;
  db2Uint32 iVendorOptionsSize;
  db2Uint32 iParallelism;
  db2Uint32 iBufferSize;
  db2Uint32 iNumBuffers;
  db2Uint32 iCallerAction;
  db2Uint32 iOptions;
  char *piComprLibrary;
  db2Uint32 iComprLibraryLen;
  void *piComprOptions;
  db2Uint32 iComprOptionsSize;
  char *piLogTarget;
  db2Uint32 iLogTargetLen;
  struct db2gStoragePathsStruct *piStoragePaths;
  char *piRedirectScript;
  db2Uint32 iRedirectScriptLen;
} db2gRestoreStruct;

typedef SQL_STRUCTURE db2gTablespaceStruct
{
  struct db2Char *tablespaces;
  db2Uint32 numTablespaces;
} db2gTablespaceStruct;

typedef SQL_STRUCTURE db2gMediaListStruct
{
  struct db2Char *locations;
  db2Uint32 numLocations;
  char locationType;
} db2gMediaListStruct;

typedef SQL_STRUCTURE db2gStoragePathsStruct
{
  struct db2Char *storagePaths;
  db2Uint32 numStoragePaths;
} db2gStoragePathsStruct;

typedef SQL_STRUCTURE db2Char
{
   char *pioData;
   db2Uint32 iLength;
   db2Uint32 oLength;
} db2Char;
```

Parámetros de la API db2Restore

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro **pDB2RestoreStruct**.

pDB2RestoreStruct
Entrada. Puntero a la estructura db2RestoreStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2RestoreStruct

piSourceDBAlias
Entrada. Serie que contiene el alias de base de datos de la imagen de copia de seguridad de la base de datos fuente.

piTargetDBAlias
Entrada. Serie que contiene el alias de base de datos destino. Si este parámetro es nulo, se utilizará el valor del parámetro **piSourceDBAlias**.

oApplicationId
Salida. La API devuelve una serie que identifica al agente que presta servicio a la aplicación. Se puede utilizar para obtener información sobre el progreso de la operación de copia de seguridad utilizando el supervisor de bases de datos.

piTimestamp
Entrada. Serie que representa la indicación de fecha y hora de la imagen de copia de seguridad. Este campo es opcional y solamente hay una imagen de copia de seguridad en el origen especificado.

piTargetDBPath
Entrada. Serie que contiene el nombre relativo o totalmente calificado del directorio de bases de datos destino en el servidor. Se utiliza si debe crearse una base de datos nueva para la copia de seguridad restaurada; de lo contrario no se utiliza.

piReportFile
Entrada. El nombre de archivo, si se especifica, debe estar totalmente calificado.

Nota: Este parámetro se ha quedado obsoleto, pero sigue definido.

piTablespaceList
Entrada. Lista de espacios de tablas que deben restaurarse. Se utiliza al restaurar un subconjunto de espacios de tablas a partir de una imagen de copia de seguridad de espacio de tablas o de base de datos. En los casos de reconstrucción, puede ser una lista de inclusión o una lista de exclusión de espacios de tablas utilizada para reconstruir la base de datos. Vea la estructura DB2TablespaceStruct. Se aplican las siguientes restricciones:

- La base de datos debe ser recuperable (solo para casos de no reconstrucción); es decir, debe habilitarse la retención de anotaciones o salidas de usuario.
- La base de datos a la que se restaura debe ser la misma base de datos que se utilizó para crear la imagen de copia de seguridad. Es decir, no pueden añadirse espacios de tablas a una base de datos mediante la función de restauración de espacios de tabla.
- El programa de utilidad de avance asegurará que los espacios de tablas restaurados en un entorno de bases de datos particionadas estén sincronizados con las demás particiones de base de datos que contengan los mismos espacios de tabla. Si se solicita una operación de restauración de espacios de tablas y **piTablespaceList** es NULL, el programa de utilidad de restauración intentará restaurar todos los espacios de tablas de la imagen de copia de seguridad.
- Al restaurar un espacio de tablas que ha cambiado de nombre desde que se realizó la copia de seguridad, debe utilizarse el nombre de espacio de tablas nuevo en el mandato de restaurar. Si se utiliza el nombre de espacio de tablas antiguo, no se encontrará.
- En el caso de la reconstrucción, se debe ofrecer la lista para 3 de los 5 tipos de reconstrucción: `DB2RESTORE_ALL_TBSP_IN_DB_EXC`, `DB2RESTORE_ALL_TBSP_IN_IMG_EXC` y `DB2RESTORE_ALL_TBSP_IN_LIST`.

piMediaList
Entrada. Soporte para la imagen de copia de seguridad.

Para obtener más información, consulte la estructura db2MediaListStruct.

piUsername
Entrada. Serie que contiene el nombre de usuario que se debe utilizar al intentar establecer una conexión. Puede ser NULL.

piPassword
Entrada. Serie que contiene la contraseña que se debe utilizar con el nombre de usuario. Puede ser NULL.

piNewLogPath
Entrada. Serie que representa la vía de acceso a utilizar para el registro cronológico una vez se ha completado la restauración. Si este campo es nulo, se utilizará la vía de acceso de anotaciones predeterminada.

piVendorOptions
Entrada. Se utiliza para pasar información desde la aplicación a las funciones de proveedor. Esta estructura de datos debe ser plana; es decir, no se admite ningún nivel de direccionamiento indirecto. Tenga en cuenta que no se realiza la inversión de bytes y que no se buscan estos datos en la página de códigos.

iVendorOptionsSize
Entrada. Longitud, en bytes, del parámetro **piVendorOptions**, que no puede ser mayor que 65535 bytes.

iParallelism
Entrada. Grado de paralelismo (número de manipuladores de almacenamientos intermedios). El mínimo es 1. El máximo es 1024.

iBufferSize
Entrada. Tamaño del almacenamiento intermedio de copia de seguridad expresado en unidades de asignación de 4 KB (páginas). El mínimo es de 8 unidades. El tamaño especificado para una restauración debe ser un entero múltiplo del tamaño de almacenamiento intermedio utilizado para generar la imagen de copia de seguridad.

iNumBuffers
Entrada. Especifica el número de almacenamientos intermedios de restauración a utilizar.

iCallerAction
Entrada. Especifica la acción que se debe realizar. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

- DB2RESTORE_RESTORE - Iniciar la operación de restaurar.
- DB2RESTORE_NOINTERRUPT - Iniciar la restauración. Especifica que la restauración se ejecutará de forma desatendida y que en las situaciones que normalmente requieran la intervención del usuario se intentarán sin primero devolver el control al llamador o bien se producirá un error. Utilice esta acción de llamador, por ejemplo, si se sabe que se han montado todos los soportes de almacenamiento necesarios para la restauración y no se desea que aparezcan mensajes de solicitud de programas de utilidad.
- DB2RESTORE_CONTINUE - Continuar la restauración después de que el usuario haya realizado alguna acción solicitada por el programa de utilidad (por ejemplo, montar una nueva cinta).

- DB2RESTORE_TERMINATE - Interrumpir la restauración después de que el usuario no haya realizado alguna acción solicitada por el programa de utilidad.
- DB2RESTORE_DEVICE_TERMINATE - Eliminar un determinado dispositivo de la lista de dispositivos utilizados por la restauración. Cuando un determinado soporte de almacenamiento está lleno, la copia de seguridad devuelve un aviso al llamador. Invoque de nuevo la restauración con esta acción de llamador para eliminar de la lista de dispositivos utilizados el dispositivo que produjo el aviso.
- DB2RESTORE_PARM_CHK - Se utiliza para validar parámetros sin realizar una restauración. Esta opción no interrumpe la conexión con la base de datos después de finalizar la ejecución de la llamada. Después de finalizar satisfactoriamente esta llamada, el usuario debe emitir otra llamada a esta API con el parámetro **iCallerAction** establecido en el valor DB2RESTORE_CONTINUE para proseguir con la restauración.
- DB2RESTORE_PARM_CHK_ONLY - Se utiliza para validar parámetros sin realizar una restauración. Antes de que concluya esta llamada, se interrumpe la conexión con la base de datos establecida por esta llamada y no es necesaria ninguna llamada más.
- DB2RESTORE_TERMINATE_INCRE - Terminar una operación de restauración incremental antes de su finalización.
- DB2RESTORE_RESTORE_STORDEF - Llamada inicial. Se solicita la redefinición del contenedor de espacio de tablas.
- DB2RESTORE_STORDEF_NOINTERRUPT - Llamada inicial. La restauración se ejecutará sin interrupciones. Se solicita la redefinición del contenedor de espacio de tablas.

iOptions

Entrada. Mapa de bits de propiedades de restauración. Las opciones se deben combinar utilizando el operador de bits OR para producir un valor para **iOptions**. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

- DB2RESTORE_OFFLINE - Realizar una operación de restaurar fuera de línea.
- DB2RESTORE_ONLINE - Realizar una operación de restaurar en línea.
- DB2RESTORE_DB - Restaurar todos los espacios de tablas de la base de datos. Debe realizarse fuera de línea.
- DB2RESTORE_TABLESPACE - Restaurar solamente los espacios de tablas listados en el parámetro **piTablespaceList** de la imagen de copia de seguridad. Esto puede realizarse en línea o fuera de línea.
- DB2RESTORE_HISTORY - Restaurar solamente el archivo histórico.
- DB2RESTORE_COMPR_LIB - Indica que debe restaurarse la biblioteca de compresión. Esta opción no puede utilizarse simultáneamente con ningún otro tipo de proceso de restauración. Si el objeto existe en la imagen de copia de seguridad, se restaurará en el directorio de la base de datos. Si el objeto no existe en la imagen de copia de seguridad, la operación de restauración fallará.
- DB2RESTORE_LOGS - Especifica que solamente debe restaurarse el conjunto de archivos de anotaciones cronológicas que se encuentra en la imagen de copia de seguridad. Si la imagen de copia de seguridad no incluye ningún archivo de anotaciones cronológicas, la operación de restauración fallará. Si se especifica esta opción, también debe especificarse la opción **piLogTarget**.

- DB2RESTORE_INCREMENTAL - Realizar una operación de restaurar acumulativa manual.
- DB2RESTORE_AUTOMATIC - Realizar una operación de restaurar acumulativa (incremental) automática. Debe especificarse con DB2RESTORE_INCREMENTAL.
- DB2RESTORE_ROLLFWD - Poner la base de datos en estado de pendiente de avance después de haberse restaurado satisfactoriamente.
- DB2RESTORE_NOROLLFWD - No poner la base de datos en estado de pendiente de avance después de haberse restaurado satisfactoriamente. Esto no puede especificarse para las copias de seguridad realizadas en línea o para restauraciones de nivel de espacios de tablas. Si, después de una restauración satisfactoria, la base de datos está en estado de pendiente de avance, debe llamarse a la API db2Rollforward para que la base de datos pueda utilizarse de nuevo.
- DB2RESTORE_GENERATE_SCRIPT - Crear un script, que puede utilizarse para realizar una restauración redirigida. **piRedirectScript** debe contener un nombre de archivo válido. **iCallerAction** debe ser DB2RESTORE_RESTORE_STORDEF o DB2RESTORE_STORDEF_NOINTERRUPT.

Los siguientes valores solamente deben utilizarse para operaciones de reconstrucción:

- DB2RESTORE_ALL_TBSP_IN_DB - Restaura la base de datos con todos los espacios de tablas conocidos por la base de datos en el momento que se restaura la imagen. Esta reconstrucción sobrescribe la base de datos si ya existe.
- DB2RESTORE_ALL_TBSP_IN_DB_EXC - Restaura la base de datos con todos los espacios de tablas conocidos por la base de datos en el momento que se restaura la imagen, excepto aquellos especificados en la lista a la que señala el parámetro **piTablespaceList**. Esta reconstrucción sobrescribe la base de datos si ya existe.
- DB2RESTORE_ALL_TBSP_IN_IMG - Restaura la base de datos solamente con los espacios de tablas de la imagen que se restaura. Esta reconstrucción sobrescribe la base de datos si ya existe.
- DB2RESTORE_ALL_TBSP_IN_IMG_EXC - Restaura la base de datos solamente con los espacios de tablas de la imagen que se restaura, excepto aquellos especificados en la lista a la que señala el parámetro **piTablespaceList**. Esta reconstrucción sobrescribe la base de datos si ya existe.
- DB2RESTORE_ALL_TBSP_IN_LIST - Restaura la base de datos solamente con los espacios de tablas especificados en la lista a la que señala el parámetro **piTablespaceList**. Esta reconstrucción sobrescribe la base de datos si ya existe.

NOTA: Si la imagen de copia de seguridad es de una base de datos recuperable, no se puede especificar WITHOUT ROLLING FORWARD (DB2RESTORE_NOROLLFWD) con ninguna de las acciones de reconstrucción anteriores.

piComprLibrary

Entrada. Indica el nombre de la biblioteca externa a utilizar para descomprimir la imagen de copia de seguridad, si la imagen está comprimida. El nombre debe ser una vía de acceso totalmente calificada que haga referencia a un archivo del servidor. Si el valor es un puntero nulo o un puntero a una serie vacía, el sistema de bases de datos DB2

intenta utilizar la biblioteca almacenada en la imagen. Si la copia de seguridad no se ha comprimido, el valor de este parámetro se pasará por alto. Si no se encuentra la biblioteca especificada, la operación de restauración fallará.

piComprOptions
Entrada. Este parámetro de API describe un bloque de datos binarios que se pasará a la rutina de inicialización en la biblioteca de descompresión. El sistema de bases de datos DB2 pasa esta serie directamente del cliente al servidor, de modo que cualquier problema de inversión de bytes o de conversión de página de códigos los debe manejar la biblioteca de compresión. Si el primer carácter del bloque de datos es'@', los datos restantes se interpretan como el nombre de un archivo que se encuentra en el servidor. El sistema de bases de datos DB2 sustituye el contenido de los parámetros **piComprOptions** y **iComprOptionsSize** por el contenido y el tamaño de este archivo y para estos nuevos valores a la rutina de inicialización.

iComprOptionsSize
Entrada. Valor entero de cuatro bytes y sin signo que representa el tamaño del bloque de datos pasado como **piComprOptions**. El parámetro **iComprOptionsSize** será cero solamente si **piComprOptions** es un puntero nulo.

piLogTarget
Entrada. Especifica la vía de acceso completa de un directorio del servidor de bases de datos que debe utilizarse como directorio de destino para extraer archivos de anotaciones cronológicas de una imagen de copia de seguridad. Si se especifica este parámetro, los archivos de anotaciones cronológicas incluidos en la imagen de copia de seguridad se extraen en el directorio de destino. Si no se especifica este parámetro, no se extraen los archivos de anotaciones cronológicas incluidos en la imagen de copia de seguridad. Para extraer sólo los archivos de anotaciones cronológicas de la imagen de copia de seguridad, deberá pasarse el valor DB2RESTORE_LOGS al parámetro **iOptions**.

Para la restauración instantánea, se debe proporciona uno de los siguientes:

- DB2RESTORE_LOGTARGET_INCLUDE "INCLUDE"

  Restaura los volúmenes del directorio de anotaciones cronológicas a partir de la imagen de instantánea. Si se especifica esta opción y la imagen de copia de seguridad contiene directorios de anotaciones cronológicas, se restaurarán. Los directorios y archivos de anotaciones cronológicas existentes en el disco permanecerán intactos si no entran en conflicto con los directorios de anotaciones cronológicas de la imagen de copia de seguridad. Si directorios de anotaciones cronológicas existentes en el disco entran en conflicto con los directorios de anotaciones cronológicas de la imagen de copia de seguridad, se devolverá un error.
- DB2RESTORE_LOGTARGET_EXCLUDE "EXCLUDE"

  No restaura los volúmenes del directorio de anotaciones cronológicas. Si se especifica esta opción, no se restaurarán los directorios de anotaciones cronológicas a partir la imagen de copia de seguridad. Los directorios y archivos de anotaciones cronológicas existentes en el disco permanecerán intactos si no entran en conflicto con los directorios de anotaciones cronológicas de la imagen de copia de seguridad. Si se restaura una vía de acceso que pertenece a la base de datos y, debido a ello se restaura

implícitamente un directorio de anotaciones cronológicas haciendo que se sobregrabe un directorio de anotaciones cronológicas, se devolverá un error.

- DB2RESTORE_LOGTARGET_INCFORCE ″INCLUDE FORCE″

  Permite que los directorios de anotaciones cronológicas existentes se sobregraben y sustituyan al restaurar la imagen instantánea. Si se especifica esta opción y la imagen de copia de seguridad contiene directorios de anotaciones cronológicas, se restaurarán. Los directorios y archivos de anotaciones cronológicas existentes en el disco permanecerán intactos si no entran en conflicto con los directorios de anotaciones cronológicas de la imagen de copia de seguridad. Si directorios de anotaciones cronológicas existentes en el disco entran en conflicto con los directorios de anotaciones cronológicas de la imagen de copia de seguridad, se sobregrabarán por los de la imagen de copia de seguridad.

- DB2RESTORE_LOGTARGET_EXCFORCE ″EXCLUDE FORCE″

  Permite que los directorios de anotaciones cronológicas existentes se sobregraben y sustituyan al restaurar la imagen instantánea. Si se especifica esta opción, no se restaurarán los directorios de anotaciones cronológicas a partir la imagen de copia de seguridad. Los directorios y archivos de anotaciones cronológicas existentes en el disco permanecerán intactos si no entran en conflicto con los directorios de anotaciones cronológicas de la imagen de copia de seguridad. Si se restaura una vía de acceso que pertenece a la base de datos y, debido a ello se restaura implícitamente un directorio de anotaciones cronológicas haciendo que se sobregrabe un directorio de anotaciones cronológicas, la restauración seguirá adelante y sobregrabará el directorio de anotaciones cronológicas en conflicto.

donde DB2RESTORE_LOGTARGET_EXCLUDE es el valor por omisión.

piStoragePaths
Entrada. Estructura que contiene campos que describen una lista de vías de acceso de almacenamiento utilizadas para el almacenamiento automático. Debe establecerse en NULL si no se va a habilitar el almacenamiento automático para la base de datos.

piRedirectScript
Entrada. El nombre de archivo para el script de restauración de redirección que se creará en el extremo cliente. El nombre de archivo puede especificarse como relativo o completo. El campo **iOptions** debe tener establecido el bit DB2RESTORE_GENERATE_SCRIPT.

Parámetros específicos de la estructura de datos db2TablespaceStruct

tablespaces
Entrada. Puntero a la lista de espacios de tablas de los que debe hacerse copia de seguridad. Para C, la lista son series de terminación nula. En el caso genérico, es una lista de estructuras db2Char.

numTablespaces
Entrada. Número de entradas en el parámetro **tablespaces**.

Parámetros de la estructura de datos db2MediaListStruct

locations
Entrada. Puntero a la lista de ubicaciones de soporte de almacenamiento. Para C, la lista son series de terminación nula. En el caso genérico, es una lista de estructuras db2Char.

numLocations
Entrada. Número de entradas del parámetro locations.

locationType
Entrada. Carácter que indica el tipo de soporte de almacenamiento. Los valores válidos (definidos en el archivo de cabecera sqlutil, ubicado en el directorio de inclusión) son:

SQLU_LOCAL_MEDIA: 'L'
Dispositivos locales (cintas, discos, disquetes o conexiones con nombre).

SQLU_XBSA_MEDIA: 'X'
Interfaz de XBSA.

SQLU_TSM_MEDIA: 'A'
Tivoli Storage Manager.

SQLU_OTHER_MEDIA: 'O'
Biblioteca de proveedor.

SQLU_SNAPSHOT_MEDIA: 'F'
Especifica que los datos se deben restaurar a partir de una copia de seguridad instantánea.

No puede utilizar SQLU_SNAPSHOT_MEDIA con ninguno de los siguientes:

- Acciones del llamador: DB2RESTORE_RESTORE_STORDEF, DB2RESTORE_STORDEF_NOINTERRUPT, DB2RESTORE_TERMINATE_INCRE
- DB2RESTORE_REPLACE_HISTORY
- DB2RESTORE_TABLESPACE
- DB2RESTORE_COMPR_LIB
- DB2RESTORE_INCREMENTAL
- DB2RESTORE_HISTORY
- DB2RESTORE_LOGS
- **piStoragePaths** - debe ser NULO o estar vacío para poder utilizarlo
- **piTargetDBPath**
- **piTargetDBAlias**
- **piNewLogPath**
- **iNumBuffers**
- **iBufferSize**
- **piRedirectScript**
- **iRedirectScriptLen**
- **iParallelism**
- **piComprLibrary**, **iComprLibraryLen**, **piComprOptions** o **iComprOptionsSize**

- El campo **numLocations** de esta estructura debe ser 1 para la restauración instantánea

Además, no puede utilizar el parámetro SNAPSHOT con ninguna operación de restauración que implique una lista de espacios de tabla.

El comportamiento por omisión al restaurar datos de una imagen de copia de seguridad instantánea será una restauración FULL DATABASE OFFLINE de todas las vías de acceso que componen la base de datos incluyendo todos los contenedores, el directorio de volúmenes local, la vía de acceso de base de datos (**DBPATH**), las vías de acceso del registro primario y del registro de anotaciones cronológicas de reflejos de la copia instantánea más reciente si no se proporciona ninguna indicación de fecha y hora (INCLUDE LOGS es el valor por omisión para todas las copias de seguridad instantáneas a menos que se indique EXCLUDE LOGS explícitamente). Si se proporciona una indicación de fecha y hora, se restaurará la imagen de copia de seguridad instantánea.

En IBM Data Server se integra un controlador de API ACS de DB2 para el hardware de almacenamiento siguiente:

- IBM TotalStorage SAN Volume Controller
- IBM Enterprise Storage Server Model 800
- IBM System Storage DS6000
- IBM System Storage DS8000
- IBM System Storage N Series
- NetApp V-series
- NetApp FAS

Parámetros de la estructura de datos db2StoragePathsStruct

storagePaths
Entrada. Matriz de series que contienen nombres totalmente calificados de vías de acceso de almacenamiento en el servidor que se utilizarán para espacios de tablas de almacenamiento automático. En una base de datos de varias particiones, se utilizan las mismas vías de acceso de almacenamiento en todas las particiones de base de datos. Si se va a restaurar una base de datos de varias particiones con nuevas vías de acceso de almacenamiento, debe restaurarse la partición de catálogo antes de restaurar las demás particiones de base de datos.

numStoragePaths
Entrada. El número de vías de acceso de almacenamiento en el parámetro **storagePaths** de la estructura db2StoragePathsStruct.

Parámetros específicos de la estructura de datos db2gRestoreStruct

iSourceDBAliasLen
Entrada. Especifica la longitud, en bytes, del parámetro **piSourceDBAlias**.

iTargetDBAliasLen
Entrada. Especifica la longitud, en bytes, del parámetro **piTargetDBAlias**.

iApplicationIdLen
Entrada. Especifica la longitud, en bytes, del parámetro **poApplicationId**.

Deberá igual a SQLU_APPLID_LEN + 1. La constante SQLU_APPLID_LEN está definida en el archivo de cabecera sqlutil ubicado en el directorio de inclusión.

iTimestampLen
Entrada. Especifica la longitud, en bytes, del parámetro **piTimestamp**.

iTargetDBPathLen
Entrada. Especifica la longitud, en bytes, del parámetro **piTargetDBPath**.

iReportFileLen
Entrada. Especifica la longitud, en bytes, del parámetro **piReportFile**.

iUsernameLen
Entrada. Especifica la longitud, en bytes, del parámetro **piUsername**. El valor se establece en cero si no se proporciona ningún nombre de usuario.

iPasswordLen
Entrada. Especifica la longitud, en bytes, del parámetro **piPassword**. El valor se establece en cero si no se proporciona ninguna contraseña.

iNewLogPathLen
Entrada. Especifica la longitud, en bytes, del parámetro **piNewLogPath**.

iLogTargetLen
Entrada. Especifica la longitud, en bytes, del parámetro **piLogTarget**.

iRedirectScriptLen
Entrada. Número entero sin signo, de 4 bytes, que representa la longitud, en bytes, del nombre de la biblioteca especificada en **piRedirectScript**. El valor se establece en cero si no se proporciona ningún nombre de script.

Parámetros de la estructura de datos db2Char

pioData
Puntero a un almacenamiento intermedio de datos de caracteres. Si el valor es NULL, no se devolverán datos.

iLength
Entrada. Tamaño del almacenamiento intermedio **pioData**.

oLength
Salida. Número de caracteres válidos de datos contenidos en el almacenamiento intermedio **pioData**.

Notas de uso

- Para la restauración fuera de línea, este programa de utilidad se conecta a la base de datos en modalidad exclusiva. El programa de utilidad falla si hay una aplicación, incluida la aplicación que llama, ya conectada a la base de datos que se restaura. Además, la petición fallará si se está utilizando el programa de utilidad de restauración para realizar la restauración, y hay una aplicación, incluida la aplicación que realiza la llamada, ya conectada a una base de datos en la misma estación de trabajo. Si la conexión se realiza satisfactoriamente, la API bloquea las demás aplicaciones hasta que se complete la restauración.
- La copia de seguridad no sustituirá al archivo de configuración de base de datos actual a menos que haya quedado inservible. En ese caso, si se sustituye el archivo, se devuelve un mensaje de aviso.
- La copia de seguridad de la base de datos o el espacio de tablas debe haberse realizado con la API db2Backup.

- Si el valor de acción del llamador es DB2RESTORE_NOINTERRUPT, la restauración continúa sin solicitudes a la aplicación. Si el valor de acción del llamador es DB2RESTORE_RESTORE y el programa de utilidad está restaurando a una base de datos existente, el programa de utilidad devuelve el control a la aplicación con un mensaje que solicita interacción del usuario. Tras manejar la interacción del usuario, la aplicación vuelva a llamar a RESTORE DATABASE, con el valor de acción del llamador establecido para indicar si el proceso debe continuar (DB2RESTORE_CONTINUE) o terminar (DB2RESTORE_TERMINATE) en la llamada siguiente. El programa de utilidad finaliza el proceso y devuelve un SQLCODE en **sqlca**.
- Para cerrar un dispositivo cuando haya finalizado, establezca el valor de acción de llamador en DB2RESTORE_DEVICE_TERMINATE. Si, por ejemplo, un usuario está restaurando desde 3 volúmenes de cinta utilizando 2 dispositivos de cinta, y se ha restaurado una de las cintas, la aplicación obtiene el control de la API con un SQLCODE que indica el fin de cinta. La aplicación puede solicitar al usuario que monte otra cinta y, si el usuario indica ″no hay más″, vuelve a la API con el valor de acción de llamador SQLUD_DEVICE_TERMINATE para indicar el fin del dispositivo de soporte. Se terminará el controlador de dispositivo, pero el resto de los dispositivos implicados en la restauración continuará procesando su entrada hasta que se hayan restaurado todos los segmentos del conjunto de restauración (el número de segmentos del conjunto de restauración se coloca en el último dispositivo de medios durante el proceso de copia de seguridad). Esta acción del llamador puede utilizarse con dispositivos que no sean de cinta (dispositivos soportados por proveedores).
- Para realizar una comprobación de parámetros antes de volver a la aplicación, establezca el valor de la acción del llamador en DB2RESTORE_PARM_CHK.
- Establezca el valor de la acción del llamador en DB2RESTORE_RESTORE_STORDEF al realizar una restauración redirigida; se utiliza conjuntamente con la API sqlbstsc.
- Si se produce una anomalía del sistema durante una fase crítica de la restauración de una base de datos, el usuario no podrá conectarse satisfactoriamente a la base de datos hasta que se realice una restauración satisfactoria. Esta condición se detectará al intentar realizar la conexión y se devolverá un mensaje de error. Si la base de datos de la que se ha hecho copia de seguridad no está configurada para la recuperación en avance y hay un archivo de configuración actual utilizable con cualquiera de estos parámetros activados, tras la restauración, el usuario deberá realizar una nueva copia de seguridad de la base de datos, o bien inhabilitar los parámetros de retención de anotaciones y salida de usuario antes de conectarse a la base de datos.
- Aunque no se descartará la base de datos restaurada (a menos que se restaure a una base de datos no existente), si la restauración falla, no podrá utilizarse.
- Si el tipo de restauración especifica que el archivo histórico de la copia de seguridad debe restaurarse, se restaurará sobre el archivo histórico existente para la base de datos, borrando así todos los cambios realizados en el archivo histórico después de la copia de seguridad que se está restaurando. Si no se desea que ocurra esto, restaure el archivo histórico a una base de datos nueva o de prueba, de forma que el contenido pueda verse sin destruir las actualizaciones que se han realizado.
- Si, en el momento de la operación de copia de seguridad, la base de datos se había habilitado para la recuperación en avance, dicha base de datos se puede dejar en el estado que tenía antes de que se produjera el daño o la corrupción emitiendo db2Rollforward después de la ejecución satisfactoria de db2Restore. Si la base de datos puede recuperarse, tomará por omisión el estado de pendiente de avance tras la conclusión de la restauración.

- Si se pone la imagen de copia de seguridad de la base de datos fuera de línea y el llamador no desea avanzar la base de datos tras la restauración, puede utilizarse la opción DB2RESTORE_NOROLLFWD para la restauración. Esto da como resultado que la base de datos puede utilizarse inmediatamente tras la restauración. Si se pone la imagen de copia de seguridad en línea, el llamador debe avanzar por los registros correspondientes de las anotaciones al completarse la restauración.
- Para restaurar archivos de anotaciones cronológicas a partir de una imagen de copia de seguridad que los contiene, se debe especificar la opción **LOGTARGET**, suponiendo que existe una vía de acceso completamente calificada y válida en el servidor DB2. Si estas condiciones se satisfacen, el programa de utilidad de restauración grabará los archivos de anotaciones cronológicas de la imagen a la vía de acceso de destino. Si se especifica **LOGTARGET** durante la restauración de una imagen de copia de seguridad que no incluya anotaciones cronológicas, la operación de restauración devuelve un error antes de intentar restaurar datos de espacios de tabla. Una operación de restauración también falla si se especifica una vía de acceso **LOGTARGET** incorrecta o de sólo lectura.
- Si existen archivos de anotaciones cronológicas en la vía de acceso **LOGTARGET** al emitir el mandato RESTORE, se devuelve una indicación de aviso al usuario. Este aviso no se devolverá si se especifica **WITHOUT PROMPTING**.
- Durante una operación de restauración en la que se especifique **LOGTARGET**, si hay algún archivo de anotaciones cronológicas que no se pueda extraer, la operación de restauración falla y devuelve un error. Si cualquiera de los archivos de anotaciones cronológicas que se están extrayendo de la imagen de copia de seguridad tiene el mismo nombre que un archivo existente en la vía de acceso **LOGTARGET**, la operación de restauración falla y se devuelve un error. El programa de utilidad de restauración no graba encima de los archivos de anotaciones cronológicas existentes en el directorio **LOGTARGET**.
- Sólo puede restaurar el conjunto de anotaciones cronológicas guardadas de una imagen de copia de seguridad. Para indicar que sólo se deben restaurar los archivos de anotaciones cronológicas, especifique la opción **LOGS** además de la vía de acceso **LOGTARGET**. Si especifica la opción **LOGS** sin una vía de acceso **LOGTARGET**, se producirá un error. Si se produce algún problema al restaurar archivos de anotaciones cronológicas en esta modalidad, la operación de restauración finaliza inmediatamente y se devuelve un error.
- Durante una operación de restauración incremental automática, sólo se recuperan de la imagen de copia de seguridad las anotaciones cronológicas incluidas en la imagen de destino de la operación de restauración. Las anotaciones cronológicas incluidas en las imágenes intermedias a que se ha hecho referencia durante el proceso de restauración incremental no se extraen de dichas imágenes de copia de seguridad intermedias. Durante una operación de restauración incremental manual, la vía de acceso **LOGTARGET** sólo se debe especificar con el mandato RESTORE final.
- Si una copia de seguridad está comprimida, el sistema de bases de datos DB2 detecta este estado y descomprime los datos automáticamente antes de restaurarlos. Si se especifica una biblioteca en la API db2Restore, se utiliza para descomprimir los datos. Si no se especifica una biblioteca en la API db2Restore, se utiliza la biblioteca almacenada en la imagen de copia de seguridad. Además, si no hay ninguna biblioteca almacenada en la imagen de copia de seguridad, los datos no se podrán descomprimir y la operación de restauración fallará.
- Si la biblioteca de compresión se restaura desde una imagen de copia de seguridad (ya sea explícitamente especificando el tipo de restauración DB2RESTORE_COMPR_LIB o implícitamente realizando una restauración

normal de una copia de seguridad comprimida), la operación de restauración se deberá realizar en la misma plataforma en que se haya realizado la copia de seguridad. Si las plataformas son distintas, la operación de restauración fallará, aunque el sistema de bases de datos DB2 normalmente soporte operaciones de restauración de plataforma cruzada que implican dos sistemas.

- Si se restaura una base de datos que esté habilitada para almacenamiento automático, las vías de acceso de almacenamiento asociadas a esta base de datos pueden volver a definirse o permanecer como estaban. Para conservar las definiciones de vías de acceso de almacenamiento como están, no proporcione otras vías de acceso de almacenamiento como parte de la operación de restauración. De lo contrario, especifique un nuevo conjunto de vías de acceso de almacenamiento para asociarlas con la base de datos. Los espacios de tablas del almacenamiento automático se redirigirán automáticamente a las nuevas vías de acceso de almacenamiento durante la operación de restauración.

Restauración instantánea

Igual que una restauración tradicional (no instantánea), el comportamiento por omisión al restaurar una imagen de copia de seguridad instantánea será NO restaurar los directorios de anotaciones cronológicas — DB2RESTORE_LOGTARGET_EXCLUDE.

Si el gestor de DB2 detecta que algún ID de grupo del directorio de anotaciones cronológicas se comparte entre otras vías de acceso que se deben restaurar, se devolverá un error. En este caso, se deben especificar DB2RESTORE_LOGTARGET_INCLUDE o DB2RESTORE_LOGTARGET_INCFORCE ya que los directorios da anotaciones cronológicas deben formar parte de la restauración.

El gestor de DB2 realizará todo lo posible para guardar los directorios de anotaciones cronológicas existentes (primario, reflejos y desbordamiento) antes de que tenga lugar la restauración de las vías de acceso a partir de la imagen de copia de seguridad.

Si desea que los directorios de anotaciones cronológicas se restauren y el gestor de DB2 detecta que los directorios de anotaciones cronológicas preexistentes en el disco entran en conflicto con los directorios de anotaciones cronológicas de la imagen de copia de seguridad, el gestor de DB2 informará de un error. En este caso, si ha especificado DB2RESTORE_LOGTARGET_INCFORCE, se suprimirá este error y se restaurarán los directorios de anotaciones cronológicas a partir de la imagen, suprimiendo lo que existía de antemano.

Existe un caso especial cuando se especifica la opción DB2RESTORE_LOGTARGET_EXCLUDE y una vía de acceso de directorio de anotaciones cronológicas reside bajo el directorio de bases de datos (por ejemplo, /NODExxxx/SQLxxxxx/SQLOGDIR/). En este caso, una restauración sobregrabará el directorio de anotaciones cronológicas y la vía de acceso de base de datos, y se restaurará todo el contenido. Si el gestor de DB2 detecta este caso y existen archivos de anotaciones cronológicas en este directorio de anotaciones cronológicas, se informará de un error. Si especifica DB2RESTORE_LOGTARGET_EXCLUDE, se suprimirá este error y los directorios de anotaciones cronológicas de la imagen de copia de seguridad sobregrabarán los directorios de anotaciones cronológicas en conflicto en el disco.

db2Rollforward - Avanzar una base de datos

Recupera una base de datos aplicando las transacciones registradas en los archivos de anotaciones cronológicas de base de datos. Se invoca después de haber restaurado una base de datos o una copia de seguridad de espacio de tablas, o si la base de datos ha dejado fuera de línea algún espacio de tablas debido a un error de soporte de almacenamiento. La base de datos debe ser recuperable (es decir, el parámetro de configuración de base de datos **logarchmeth1** o el parámetro de configuración de base de datos **logarchmeth2** se debe establecer en un valor distinto de OFF) antes de poder recuperar la base de datos con una recuperación en avance.

Ámbito

En un entorno de bases de datos particionadas, se debe llamar a esta API desde la partición de catálogo. Las particiones se avanzan según lo que especifique en la cláusula TO:

- Una llamada de avance a un punto en el tiempo afecta a todos los servidores de particiones de base de datos que se listan en el archivo `db2nodes.cfg`.
- Una llamada de avance END OF LOGS afecta a los servidores de particiones de bases de datos que se especifican en la cláusula ON DATABASE PARTITION. Si no se especifican servidores de particiones de base de datos, la llamada de avance afecta a todos los servidores de particiones de base de datos que se listan en el archivo `db2nodes.cfg`.
- Una llamada de avance de base de datos o de espacio de tablas especificando el fin de la copia de seguridad afecta a todos los servidores de particiones de base de datos que se listan en el archivo `db2nodes.cfg`.

Si ya se han aplicado todas las transacciones de un servidor de particiones de base de datos en particular a la base de datos actual y, por lo tanto, no se debe avanzar ninguna de esas transacciones, ese servidor de particiones de base de datos se pasa por alto.

Cuando se avanza una tabla particionada a un punto en el tiempo determinado, también se deben avanzar los espacios de tablas que contienen esa tabla hasta el mismo punto en el tiempo. Sin embargo, cuando se avanza un espacio detablas, no se tienen que avanzar todas las tablas de ese espacio de tablas.

Autorización

Una de las siguientes:

- *sysadm*
- *sysctrl*
- *sysmaint*

Conexión necesaria

Ninguna. Esta API establece una conexión de base de datos.

Archivo de inclusión de la API

`db2ApiDf.h`

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2Rollforward (
       db2Uint32 versionNumber,
       void * pDB2RollforwardStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2RollforwardStruct
{
  struct db2RfwdInputStruct *piRfwdInput;
  struct db2RfwdOutputStruct *poRfwdOutput;
} db2RollforwardStruct;

typedef SQL_STRUCTURE db2RfwdInputStruct
{
  sqluint32 iVersion;
  char *piDbAlias;
  db2Uint32 iCallerAction;
  char *piStopTime;
  char *piUserName;
  char *piPassword;
  char *piOverflowLogPath;
  db2Uint32 iNumChngLgOvrflw;
  struct sqlurf_newlogpath *piChngLogOvrflw;
  db2Uint32 iConnectMode;
  struct sqlu_tablespace_bkrst_list *piTablespaceList;
  db2int32 iAllNodeFlag;
  db2int32 iNumNodes;
  SQL_PDB_NODE_TYPE *piNodeList;
  db2int32 iNumNodeInfo;
  char *piDroppedTblID;
  char *piExportDir;
  db2Uint32 iRollforwardFlags;
} db2RfwdInputStruct;

typedef SQL_STRUCTURE db2RfwdOutputStruct
{
  char *poApplicationId;
  sqlint32 *poNumReplies;
  struct sqlurf_info *poNodeInfo;
  db2Uint32 oRollforwardFlags;
} db2RfwdOutputStruct;

SQL_STRUCTURE sqlurf_newlogpath
{
   SQL_PDB_NODE_TYPE nodenum;
   unsigned short  pathlen;
   char logpath[SQL_LOGPATH_SZ+SQL_LOGFILE_NAME_SZ+1];
};

typedef SQL_STRUCTURE sqlu_tablespace_bkrst_list
{
   sqlint32 num_entry;
   struct sqlu_tablespace_entry *tablespace;
} sqlu_tablespace_bkrst_list;

typedef SQL_STRUCTURE sqlu_tablespace_entry
{
   sqluint32 reserve_len;
   char tablespace_entry[SQLU_MAX_TBS_NAME_LEN+1];
   char filler[1];
} sqlu_tablespace_entry;

SQL_STRUCTURE sqlurf_info
{
   SQL_PDB_NODE_TYPE nodenum;
   sqlint32 state;
```

```
   unsigned char   nextarclog[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   firstarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   lastarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   lastcommit[SQLUM_TIMESTAMP_LEN+1];
};

SQL_API_RC SQL_API_FN
  db2gRollforward (
       db2Uint32 versionNumber,
       void * pDB2gRollforwardStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gRollforwardStruct
{
  struct db2gRfwdInputStruct *piRfwdInput;
  struct db2RfwdOutputStruct *poRfwdOutput;
} db2gRollforwardStruct;

typedef SQL_STRUCTURE db2gRfwdInputStruct
{
  db2Uint32 iDbAliasLen;
  db2Uint32 iStopTimeLen;
  db2Uint32 iUserNameLen;
  db2Uint32 iPasswordLen;
  db2Uint32 iOvrflwLogPathLen;
  db2Uint32 iDroppedTblIDLen;
  db2Uint32 iExportDirLen;
  sqluint32 iVersion;
  char *piDbAlias;
  db2Uint32 iCallerAction;
  char *piStopTime;
  char *piUserName;
  char *piPassword;
  char *piOverflowLogPath;
  db2Uint32 iNumChngLgOvrflw;
  struct sqlurf_newlogpath *piChngLogOvrflw;
  db2Uint32 iConnectMode;
  struct sqlu_tablespace_bkrst_list *piTablespaceList;
  db2int32 iAllNodeFlag;
  db2int32 iNumNodes;
  SQL_PDB_NODE_TYPE *piNodeList;
  db2int32 iNumNodeInfo;
  char *piDroppedTblID;
  char *piExportDir;
  db2Uint32 iRollforwardFlags;
} db2gRfwdInputStruct;
```

Parámetros de la API db2Rollforward

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro.

pDB2RollforwardStruct
Entrada. Puntero a la estructura db2RollforwardStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2RollforwardStruct

piRfwdInput
Entrada. Puntero a la estructura db2RfwdInputStruct.

poRfwdOutput
Salida. Puntero a la estructura db2RfwdOutputStruct.

Parámetros de la estructura de datos db2RfwdInputStruct

iVersion
Entrada. ID de versión de los parámetros de avance. Se define como SQLUM_RFWD_VERSION.

piDbAlias
Entrada. Serie que contiene el alias de la base de datos. Es el alias de base de datos que se catalogará en el directorio de bases de datos del sistema.

iCallerAction
Entrada. Especifica la acción que se debe realizar. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2ROLLFORWARD_ROLLFWD
Avanzar hasta el punto del tiempo especificado por el parámetro piStopTime. Para el avance de base de datos, la base de datos se deja en estado de pendiente de avance. Para el avance de espacios de tablas hasta un punto del tiempo, los espacios de tablas se dejan en estado de pendiente de avance en proceso.

DB2ROLLFORWARD_STOP
Finalizar la recuperación de avance avanzando la base de datos utilizando los archivos de anotaciones disponibles y luego retrotrayéndolos. Las transacciones no confirmadas se restituyen y el estado de pendiente de avance de la base de datos o espacios de tablas se desactiva. Un sinónimo de este valor es DB2ROLLFORWARD_RFWD_COMPLETE.

DB2ROLLFORWARD_RFWD_STOP
Avanzar hasta el punto del tiempo especificado por el parámetro piStopTime y finalizar la recuperación en avance. El estado de pendiente de avance de la base de datos o espacios de tablas se desactiva. Un sinónimo de este valor es DB2ROLLFORWARD_RFWD_COMPLETE.

DB2ROLLFORWARD_QUERY
Consultar valores para nextarclog, firstarcdel, lastarcdel y lastcommit. Devolver estado de base de datos y un número de nodo.

DB2ROLLFORWARD_PARM_CHECK
Validar parámetros sin realizar el avance.

DB2ROLLFORWARD_CANCEL
Cancelar la operación de avance que se está ejecutando actualmente. La base de datos o el espacio de tablas si sitúan en estado de pendiente de recuperación.

Nota: Esta opción no se puede utilizar mientras se está ejecutando la operación de avance. Puede utilizarse si la operación de avance queda en pausa (es decir, en espera de una detención (STOP)) o si se ha producido una anomalía del sistema durante la operación de avance. Debe utilizarse con precaución.
Puede que la recuperación en avance de bases de datos requiera una recuperación de carga utilizando dispositivos de cinta. La API de avance devolverá un mensaje de aviso si es necesaria la intervención del usuario en un dispositivo. Puede llamarse de nuevo a la API con una de las tres siguientes acciones de llamador:

DB2ROLLFORWARD_LOADREC_CONT
Continuar utilizando el dispositivo que ha generado el mensaje de aviso (por ejemplo, cuando se ha montado una cinta nueva).

DB2ROLLFORWARD_DEVICE_TERM
Dejar de utilizar el dispositivo que ha generado el mensaje de aviso (por ejemplo, cuando no hay más cintas).

DB2ROLLFORWARD_LOAD_REC_TERM
Terminar todos los dispositivos utilizados por la recuperación de carga.

piStopTime
Entrada. Serie de caracteres que contiene una indicación fecha y hora en formato ISO. La recuperación de la base de datos se detendrá cuando se sobrepase esta indicación de fecha y hora. Especifique SQLUM_INFINITY_TIMESTAMP para avanzar todo lo posible. Puede ser NULL para DB2ROLLFORWARD_QUERY, DB2ROLLFORWARD_PARM_CHECK y para cualquiera de las acciones de llamador de recuperación de carga (B2ROLLFORWARD_LOADREC_xxx).

piUserName
Entrada. Serie que contiene el nombre de usuario de la aplicación. Puede ser NULL.

piPassword
Entrada. Serie que contiene la contraseña del nombre de usuario especificado (si lo hay). Puede ser NULL.

piOverflowLogPath
Entrada. Este parámetro se utiliza para especificar la vía de acceso de anotaciones cronológicas alternativa que se debe utilizar. Además de los archivos de anotaciones cronológicas activos, el usuario debe mover los archivos de anotaciones cronológicas archivados a la vía de acceso de anotaciones cronológicas para que este programa de utilidad pueda utilizarlos. Esto puede representar un problema si la base de datos no tiene espacio suficiente en la vía de acceso de anotaciones. Por este motivo se proporciona la vía de acceso de anotaciones cronológicas de desbordamiento. Durante la recuperación en avance, se busca en los archivos de anotaciones necesarios, primero en la vía de acceso de anotaciones y luego en la vía de acceso de anotaciones de desbordamiento. Los archivos de anotaciones cronológicas necesarios para la recuperación en avance de espacios de tablas pueden especificarse en la vía de acceso de anotaciones cronológicas o en la vía de acceso de anotaciones cronológicas de desbordamiento. Si el llamador no especifica una vía de acceso de anotaciones de desbordamiento, el valor por omisión es la vía de acceso de anotaciones. En un entorno de bases de datos particionadas, la vía de acceso de anotaciones cronológicas de desbordamiento debe ser una vía de acceso completamente calificada válida; la vía de acceso por omisión es la vía de acceso de anotaciones cronológicas de desbordamiento por omisión de cada nodo. En un entorno de bases de datos de una sola partición, la vía de acceso de anotaciones cronológicas de desbordamiento puede ser relativa si el servidor es local.

iNumChngLgOvrflw
Entrada. Entornos de bases de datos particionadas solamente. El número de vías de acceso de anotaciones cronológicas de desbordamiento cambiadas. Estas nuevas vías de acceso de anotaciones cronológicas alteran

temporalmente la vía de acceso de anotaciones cronológicas de desbordamiento por omisión sólo para el servidor de particiones de base de datos especificado.

piChngLogOvrflw
Entrada. Entornos de bases de datos particionadas solamente. Un puntero a una estructura que contiene los nombres totalmente calificados de vías de acceso de anotaciones cronológicas de desbordamiento cambiadas. Estas nuevas vías de acceso de anotaciones cronológicas alteran temporalmente la vía de acceso de anotaciones cronológicas de desbordamiento por omisión sólo para el servidor de particiones de base de datos especificado.

iConnectMode
Entrada. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2ROLLFORWARD_OFFLINE
Avance fuera de línea. Este valor debe especificarse para recuperación en avance de bases de datos.

DB2ROLLFORWARD_ONLINE
Avance en línea.

piTablespaceList
Entrada. Puntero a una estructura que contiene los nombres de los espacios de tablas que deben recuperarse en avance hasta el final de las anotaciones o hasta un punto del tiempo específico. Si no se especifica, se seleccionarán los espacios de tablas que necesiten recuperación en avance.

Para tablas particionadas, el avance de punto del tiempo (PIT) de un espacio de tablas que contenga cualquier parte de una tabla particionada también debe recuperar en avance todos los demás espacios de tablas en los que resida dicha tabla hasta el mismo punto del tiempo. El avance hasta el final de las anotaciones cronológicas de un solo espacio de tablas que contenga una parte de una tabla particionada sigue estando permitida.

Si una tabla particionada tiene particiones de datos conectadas, desconectadas o descartadas, el avance de tipo PIT debe incluir también todos los espacios de tablas de estas particiones de datos. Para determinar si una tabla particionada tiene particiones de datos conectadas, desconectadas o descartadas, consulte el campo de estado (Status) de la tabla de catálogo SYSDATAPARTITIONS.

Dado que una tabla particionada puede residir en varios espacios de tablas, generalmente es necesario realizar el avance en varios espacios de tablas. Los datos que se recuperan por medio de la recuperación de tablas descartadas se graban en el directorio de exportación especificado en el parámetro piExportDir. Es posible realizar el avance de todas las tablas en un solo mandato, o realizar operaciones de avance repetidas para subconjuntos de los espacios de tablas implicados. Se escribirá un aviso en las anotaciones de notificación si la API db2Rollforward no ha especificado el conjunto completo de espacios de tablas necesarios para recuperar todos los datos de la tabla. Se devolverá al usuario un aviso con detalles completos de todas las particiones no recuperadas en el mandato encontrado en las anotaciones de notificación de administración.

El hecho de permitir la operación de avance de un subconjunto de los espacios de tablas facilita el tratamiento de los casos en los que la cantidad de datos que deben recuperarse no cabe en un solo directorio de exportación.

iAllNodeFlag
Entrada. Entornos de bases de datos particionadas solamente. Indica si la operación de avance se debe aplicar a todos los servidores de particiones de base de datos definidos en db2nodes.cfg. Los valores válidos son:

DB2_NODE_LIST
Aplicar a servidores de particiones de base de datos de una lista que se pasa en piNodeList.

DB2_ALL_NODES
Aplicar a todos los servidores de particiones de base de datos. Es el valor por omisión. El parámetro piNodeList no debe establecerse en NULL si se utiliza este valor.

DB2_ALL_EXCEPT
Aplicar a todos los servidores de particiones de base de datos excepto a los de una lista que se pasa en piNodeList.

DB2_CAT_NODE_ONLY
Aplicar sólo a la partición de catálogo. El parámetro piNodeList no debe establecerse en NULL si se utiliza este valor.

iNumNodes
Entrada. Especifica el número de servidores de particiones de base de datos de la matriz piNodeList.

piNodeList
Entrada. Puntero a una matriz de servidores de particiones de base de datos en los que realizar la recuperación en avance.

iNumNodeInfo
Entrada. Define el tamaño del parámetro de salida poNodeInfo, que debe ser lo suficientemente grande como para contener información de estado de cada partición de base de datos que se avanza. En un entorno de bases de datos de una sola partición, este parámetro debe establecerse en 1. El valor de este parámetro debe ser el mismo que el número de servidores de particiones de base de datos para los que se llama a esta API.

piDroppedTblID
Entrada. Serie que contiene el ID de la tabla descartada cuya recuperación se está intentando. Para tablas particionadas, el ID de tabla descartada (drop-table-id) identifica la tabla en conjunto, a fin de que todas las particiones de datos de la tabla puedan recuperarse en un solo mandato de avance.

piExportDir
Entrada. Nombre del directorio al que se exportarán los datos de la tabla descartada.

iRollforwardFlags
Entrada. Especifica los distintivos de avance. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2ROLLFORWARD_EMPTY_FLAG
No se especifican distintivos.

DB2ROLLFORWARD_LOCAL_TIME
Permite al usuario el avance hasta un punto del tiempo que es la hora local del usuario en vez de la hora GMT. Esto facilita a los usuarios el avance hasta un punto del tiempo específico en sus

máquinas locales y elimina potenciales errores de usuario causados por la conversión de la hora local a la hora GMT.

DB2ROLLFORWARD_NO_RETRIEVE
Controla qué archivos de anotaciones cronológicas se avanzarán en la máquina de espera, al permitir que el usuario inhabilite la recuperación de las anotaciones cronológicas archivadas. Al controlar el avance de los archivos de anotaciones cronológicas, el usuario puede asegurarse que la máquina de espera estará X horas por detrás de la máquina de producción, para evitar que el usuario afecte a ambos sistemas. Esta opción resulta de utilidad si el sistema de espera no tiene acceso al archivador, por ejemplo, si TSM es el archivador, sólo permite a la máquina original recuperar los archivos. También eliminará la posibilidad de que el sistema de espera recupere un archivo de anotaciones incompleto mientras el sistema de producción está archivando un archivo y el sistema de espera está recuperando el mismo archivo.

DB2ROLLFORWARD_END_OF_BACKUP
Especifica que la base de datos se debe avanzar hasta la *hora de recuperación mínima*.

Parámetros de la estructura de datos db2RfwdOutputStruct

poApplicationId
Salida. ID de la aplicación.

poNumReplies
Salida. Número de respuestas recibidas.

poNodeInfo
Salida. Información de respuesta de particiones de base de datos.

oRollforwardFlags
Salida. Distintivos de salida de avance. Los valores válidos son:

DB2ROLLFORWARD_OUT_LOCAL_TIME
Indica al usuario que la indicación de fecha y hora de la última transacción confirmada se visualiza en la hora local en lugar de en UTC. La hora local se basa en la hora local del servidor, no en la del cliente. En un entorno de base de datos particionada, la hora local se basa en la hora local de la partición del catálogo.

Parámetros de la estructura de datos sqlurf_newlogpath

nodenum
Entrada. Número de la partición de base de datos detallada por esta estructura.

pathlen
Entrada. Longitud total del campo logpath.

logpath
Entrada. Una vía de acceso totalmente calificada que debe utilizarse para un nodo específico de la operación de avance.

Parámetros de la estructura de datos sqlu_tablespace_bkrst

num_entry
Entrada. El número de estructuras contenidas en la lista a la que señala el parámetro de espacio de tablas.

tablespace
Entrada. Puntero a una lista de estructuras sqlu_tablespace_entry.

Parámetros de la estructura de datos sqlu_tablespace_entry

reserve_len
Entrada. Especifica la longitud del parámetro tablespace_entry, en bytes.

tablespace_entry
Entrada. El nombre del espacio de tablas en el que debe realizarse la operación de avance.

filler Rellenador utilizado para la alineación correcta de la estructura de datos en la memoria.

Parámetros de la estructura de datos sqlurf_info

nodenum
Salida. Número de la partición de base de datos para la que esta estructura contiene información.

state Salida. El estado actual de la base de datos o de los espacios de tablas que se han incluido en el avance en una partición de base de datos.

nextarclog
Salida. Si la operación de avance ha finalizado, este campo estará vacío. Si la operación de avance aún no ha finalizado, será el nombre del próximo archivo de anotaciones que se procesará para el avance.

firstarcdel
Salida. El primer archivo de anotaciones reproducido por el avance.

lastarcdel
Salida. El último archivo de anotaciones reproducido por el avance.

lastcommit
Salida. La hora de la última transacción confirmada.

Parámetros específicos de la estructura de datos db2gRfwdInputStruct

iDbAliasLen
Entrada. Especifica la longitud del alias de base de datos, en bytes.

iStopTimeLen
Entrada. Especifica la longitud del parámetro de hora de detención (stop time), en bytes. El valor se establece en cero si no se proporciona ninguna hora de detención.

iUserNameLen
Entrada. Especifica la longitud del nombre de usuario, en bytes. El valor se establece en cero si no se proporciona ningún nombre de usuario.

iPasswordLen
Entrada. Especifica la longitud, en bytes, de la contraseña. El valor se establece en cero si no se proporciona ninguna contraseña.

iOverflowLogPathLen
Entrada. Especifica la longitud de la vía de acceso de anotaciones de desbordamiento, en bytes. El valor se establece en cero si no se proporciona ninguna vía de acceso de anotaciones de desbordamiento.

iDroppedTblIDLen
Entrada. Especifica la longitud del ID de tabla descartada (parámetro piDroppedTblID), en bytes. El valor se establece en cero si no se proporciona ningún ID de tabla descartada.

iExportDirLen
Entrada. Especifica la longitud del directorio de exportación de tabla descartada (parámetro piExportDir), en bytes. El valor se establece en cero si no se proporciona ningún directorio de exportación de tabla descartada.

Notas de uso

El gestor de bases de datos utiliza la información almacenada en los archivos de anotaciones cronológicas activos y archivados para reconstruir las transacciones realizadas en la base de datos desde su última copia de seguridad.

La acción realizada cuando se llama a esta API depende del distintivo rollforward_pending de la base de datos antes de la llamada. Puede consultarse utilizando db2CfgGet - Obtener parámetros de configuración. El distintivo rollforward_pending se establece en DATABASE si la base de datos está en estado de pendiente de avance. Se establece en TABLESPACE si uno o varios espacios de tablas se encuentran en estado SQLB_ROLLFORWARD_PENDING o SQLB_ROLLFORWARD_IN_PROGRESS. El distintivo rollforward_pending se establece en NO si ni la base de datos ni ninguno de los espacios de tablas necesita operación de avance.

Si la base de datos se encuentra en estado de pendiente de avance cuando se llama a esta API, la base de datos se someterá a operación de avance. Las espacios de tablas vuelven al estado normal después de una operación de avance de base de datos satisfactoria, a menos que un estado anómalo provoque que uno o varios espacios de tablas queden fuera de línea. Si el distintivo rollforward_pending se establece en TABLESPACE, la operación de avance sólo se realizará en los espacios de tablas que estén en estado de pendiente de avance o en los espacios de tablas solicitados por nombre.

Nota: Si la operación de avance de espacios de tablas termina anormalmente, los espacios de tablas que se estaban sometiendo a la operación de avance se situarán en estado SQLB_ROLLFORWARD_IN_PROGRESS. En la próxima invocación de ROLLFORWARD DATABASE, sólo se procesarán los espacios de tablas en estado SQLB_ROLLFORWARD_IN_PROGRESS. Si el conjunto de nombres de espacio de tablas seleccionados no incluye todos los espacios de tablas cuyo estado es SQLB_ROLLFORWARD_IN_PROGRESS, los espacios de tablas que no sean necesarios se situarán en estado SQLB_RESTORE_PENDING.

Si la base de datos no se encuentra en estado de pendiente de avance y no se especifica ningún punto del tiempo, los espacios de tablas que se encuentren en estado de avance en proceso se avanzarán hasta el final de las anotaciones. Si ningún espacio de tablas se encuentra en estado de avance en proceso, los espacios de tablas que se encuentren en estado de pendiente de avance se avanzarán hasta el final de las anotaciones.

Esta API lee los archivos de anotaciones empezando por el archivo de anotaciones que coincide con la imagen de copia de seguridad. El nombre de este archivo de anotaciones puede determinarse llamando a esta API con la acción de llamador DB2ROLLFORWARD_QUERY antes de realizar la operación de avance de los archivos de anotaciones.

Las transacciones contenidas en los archivos de anotaciones se reaplican a la base de datos. Las anotaciones se procesan durante todo el tiempo durante el que haya información disponible, o hasta la hora especificada por el parámetro de hora de detención (stop time).

La recuperación se detiene cuando se produce alguno de los siguientes eventos:

- No se encuentran más archivos de anotaciones
- Una indicación fecha y hora del archivo de anotaciones cronológicas sobrepasa la indicación de fecha y hora de finalización especificada por el parámetro de hora de detención
- Se produce un error al leer el archivo de anotaciones.

Es posible que algunas transacciones no puedan recuperare. El valor devuelto en lastcommit indica la indicación de fecha y hora de la última transacción confirmada que se ha aplicada a la base de datos.

Si la causa de la recuperación de la base de datos ha sido un error humano o de aplicación, el usuario puede especificar un valor de indicación de fecha y hora en piStopTime, indicando que la recuperación debe detenerse antes de la hora del error. Esto se aplica sólo al avance de bases de datos completas y a la recuperación en avance de espacios de tablas hasta un punto del tiempo. También permite detener la recuperación antes de que se produzca un error de lectura de anotaciones, determinado durante un intento de recuperación fallido anterior.

Si el distintivo rollforward_recovery está establecido en DATABASE, la base de datos no estará disponible para el uso hasta que termine la recuperación en avance. La finalización se realiza llamando a la API con la acción de llamador DB2ROLLFORWARD_STOP o DB2ROLLFORWARD_RFWRD_STOP para sacar a la base de datos del estado de pendiente de avance. Si el distintivo rollforward_recovery es TABLESPACE, la base de datos está disponible para el uso. Sin embargo, los espacios de tablas que se encuentren en estado SQLB_ROLLFORWARD_PENDING y SQLB_ROLLFORWARD_IN_PROGRESS no estarán disponibles hasta que se llame a la API para realizar la recuperación en avance de espacios de tablas. Si se realiza el avance de espacios de tablas hasta un punto del tiempo, los espacios de tablas se sitúan en estado de pendiente de copia de seguridad después de una operación de avance satisfactoria.

Si la opción RollforwardFlags está establecida en DB2ROLLFORWARD_LOCAL_TIME, todos los mensajes devueltos al usuario también estarán en la hora local. Todas las horas se convierten en el servidor y, en un entorno de bases de datos particionadas, en la partición de base de datos de catálogo. La serie de indicación de fecha y hora se convierte a GMT en el servidor, de manera que la hora es local para el huso horario del servidor, no del cliente. Si el cliente está en una zona horaria y el servidor en otra, debe utilizarse la hora local del servidor. Esto es diferente de la opción de la hora local del Centro de control, que es local para el cliente. Si la serie de indicación de fecha y hora está próxima al cambio de hora del reloj a causa del horario de verano, es importante saber si la hora de detención es anterior o posterior al cambio de hora y especificarlo correctamente.

db2Runstats - Actualizar estadísticas para tablas e índices

Actualiza estadísticas sobre las características de una tabla y/o los índices o vistas de estadísticas asociados. Estas características incluyen, entre otros elementos, el número de registros, el número de páginas y el promedio de longitud de registro. El optimizador utiliza estas estadísticas al determinar las vías de acceso a los datos.

Cuando se utiliza sobre tablas, es conveniente invocar este programa de utilidad cuando se han realizado muchas actualizaciones en una tabla, después de reorganizar una tabla o después de crear un nuevo índice.

Las estadísticas están basadas en la porción de la tabla que reside en la partición de base de datos donde se ejecuta la API. Las estadísticas de tabla globales se obtienen multiplicando los valores obtenidos en una partición de base de datos por el número de particiones que albergan la tabla completa. Las estadísticas globales se almacenan en las tablas de catálogo. No es necesario que la partición de base de datos desde donde se invoca la API contenga una porción de la tabla:

- Si la API se invoca desde una partición de base de datos donde reside una porción de la tabla, el programa de utilidad se ejecuta en esta partición de base de datos.
- Si la API se invoca desde una partición de base de datos que no contiene una porción de la tabla, la petición se envía a la primera partición del grupo de particiones que contenga una porción de la tabla. A continuación, el programa de utilidad se ejecuta en esta partición de base de datos. Al recopilar estadísticas para una vista de estadísticas, se recopilan estadísticas de todas las particiones de base de datos.

Cuando se utiliza sobre vistas estadísticas, es conveniente invocar este programa de utilidad cuando cambios en tablas subyacentes han afectado significativamente a las filas devueltas por una vista. Estas vistas se deben haber habilitado para su utilización en la optimización de consultas, mediante ″ALTER VIEW ... ENABLE QUERY OPTIMIZATION.″

Ámbito

Esta API se puede invocar desde cualquier servidor de particiones de base de datos definido en el archivo db2nodes.cfg. Puede utilizarse para actualizar los catálogos en la partición de base de datos de catálogo.

Autorización

Cuando se utiliza sobre tablas, una de las autorizaciones o privilegios siguientes:

- sysadm
- sysctrl
- sysmaint
- Privilegio CONTROL sobre la tabla
- LOAD

Cuando se utiliza sobre vistas estadísticas, una de las autorizaciones o privilegios siguientes:

- sysadm
- sysctrl
- sysmaint

- dbadm
- Privilegio CONTROL sobre la vista

Además, el usuario debe tener la autorización o privilegio apropiado para acceder a las filas de la vista. Específicamente, para cada tabla, vista o apodo referenciado en la definición de la vista, el usuario debe tener una de las autorizaciones o privilegios siguientes:

- sysadm o dbadm
- Privilegio CONTROL
- Privilegio SELECT

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2Runstats (
               db2Uint32 versionNumber,
               void * data,
               struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2RunstatsData
{
  double iSamplingOption;
  unsigned char *piTablename;
  struct db2ColumnData        **piColumnList;
  struct db2ColumnDistData    **piColumnDistributionList;
  struct db2ColumnGrpData     **piColumnGroupList;
  unsigned char               **piIndexList;
  db2Uint32 iRunstatsFlags;
  db2int16 iNumColumns;
  db2int16 iNumColdist;
  db2int16 iNumColGroups;
  db2int16 iNumIndexes;
  db2int16 iParallelismOption;
  db2int16 iTableDefaultFreqValues;
  db2int16 iTableDefaultQuantiles;
  db2Uint32 iSamplingRepeatable;
  db2Uint32 iUtilImpactPriority;
} db2RunstatsData;

typedef SQL_STRUCTURE db2ColumnData
{
  unsigned char *piColumnName;
  db2int16 iColumnFlags;
} db2ColumnData;

typedef SQL_STRUCTURE db2ColumnDistData
{
  unsigned char *piColumnName;
  db2int16 iNumFreqValues;
  db2int16 iNumQuantiles;
} db2ColumnDistData;

typedef SQL_STRUCTURE db2ColumnGrpData
{
  unsigned char               **piGroupColumnNames;
  db2int16 iGroupSize;
```

```
  db2int16 iNumFreqValues;
  db2int16 iNumQuantiles;
} db2ColumnGrpData;

SQL_API_RC SQL_API_FN
  db2gRunstats (
                db2Uint32 versionNumber,
                void * data,
                struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gRunstatsData
{
  double iSamplingOption;
  unsigned char *piTablename;
  struct db2gColumnData       **piColumnList;
  struct db2gColumnDistData   **piColumnDistributionList;
  struct db2gColumnGrpData    **piColumnGroupList;
  unsigned char               **piIndexList;
  db2Uint16 *piIndexNamesLen;
  db2Uint32 iRunstatsFlags;
  db2Uint16 iTablenameLen;
  db2int16 iNumColumns;
  db2int16 iNumColdist;
  db2int16 iNumColGroups;
  db2int16 iNumIndexes;
  db2int16 iParallelismOption;
  db2int16 iTableDefaultFreqValues;
  db2int16 iTableDefaultQuantiles;
  db2Uint32 iSamplingRepeatable;
  db2Uint32 iUtilImpactPriority;
} db2gRunstatsData;

typedef SQL_STRUCTURE db2gColumnData
{
  unsigned char *piColumnName;
  db2Uint16 iColumnNameLen;
  db2int16 iColumnFlags;
} db2gColumnData;

typedef SQL_STRUCTURE db2gColumnDistData
{
  unsigned char *piColumnName;
  db2Uint16 iColumnNameLen;
  db2int16 iNumFreqValues;
  db2int16 iNumQuantiles;
} db2gColumnDistData;

typedef SQL_STRUCTURE db2gColumnGrpData
{
  unsigned char               **piGroupColumnNames;
  db2Uint16 *piGroupColumnNamesLen;
  db2int16 iGroupSize;
  db2int16 iNumFreqValues;
  db2int16 iNumQuantiles;
} db2gColumnGrpData;
```

Parámetros de la API db2Runstats

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como los datos del segundo parámetro.

data Entrada. Puntero a la estructura db2RunstatsData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2RunstatsData

iSamplingOption
Entrada. Indica que deben recopilarse estadísticas sobre una muestra de datos de una tabla o vista. iSamplingOption representa el tamaño de la muestra expresado como porcentaje P. Este valor debe ser un número positivo que sea menor o igual que 100, pero puede estar comprendido entre 1 y 0. Por ejemplo, el valor 0,01 representa un porcentaje de una centésima, de forma que por termino medio se recopilarían datos en una 1 fila de cada 10.000. DB2 tratará un valor de 0 ó 100 como si no se especificara el muestreo, independientemente de si se ha especificado DB2RUNSTATS_SAMPLING_SYSTEM. DB2 tratará un valor superior a 100 o inferior a 0 como un error (SQL1197N). Los dos tipos de muestreo posibles son BERNOULLI y SYSTEM. La especificación del tipo de muestreo está controlada por el valor de DB2RUNSTATS_SAMPLING_SYSTEM en iRunstatsFlags.

piTablename
Entrada. Puntero al nombre totalmente calificado de la tabla o vista estadística para la que se deben recopilar estadísticas. El nombre puede ser un alias. Para los tipos de fila, piTablename debe ser el nombre de la tabla raíz de la jerarquía.

piColumnList
Entrada. Matriz de elementos db2ColumnData. Cada elemento de esta matriz está formado por dos subelementos como máximo:

- una serie que representa el nombre de la columna para la que se deben recopilar estadísticas
- un campo de distintivos que indican opciones estadísticas para la columna

Si iNumColumns es cero, piColumnList no se tiene en cuenta si se proporciona.

piColumnDistributionList
Entrada. Matriz de elementos db2ColumnDistData. Estos elementos se proporcionan cuando se desea recopilar estadísticas para una columna o columnas determinadas. Cada elemento de esta matriz está compuesto por hasta tres subelementos:

- una serie que representa el nombre de la columna para la que se deben recopilar estadísticas de distribución
- el número de valores frecuentes que se deben recopilar.
- el número de valores cuantiles que se deben recopilar

Se recopilan estadísticas básicas de columna para las columnas que aparecen en piColumnDistributionList y que no aparecen en piColumnList. Esto produce el mismo efecto que si esas columnas se hubieran incluido en piColumnList. Si iNumColdist es cero, se omite piColumnDistributionList.

piColumnGroupList
Entrada. Matriz de elementos db2ColumnGrpData. Estos elementos se proporcionan cuando se recopilan estadísticas para un grupo de columnas. Es decir, los valores de cada columna del grupo correspondientes a cada fila se concatenan y tratan como un valor individual. Cada db2ColumnGrpData está formada por 3 campos de enteros y una matriz de series de caracteres. El primer campo de enteros representa el número de series de caracteres de la matriz de series de caracteres piGroupColumns. Cada serie de esta matriz contiene un nombre de

columna. Por ejemplo, si se deben recopilar estadísticas de combinación de columnas para los grupos de columnas (c1,c2) y para (c3,c4,c5), existen dos elementos db2ColumnGrpData en piGroupColumns.

El primer elemento db2ColumnGrpData es como sigue: piGroupSize = 2 y la matriz de series contiene 2 elementos: c1 y c2.

El segundo elemento db2ColumnGrpData es como sigue: piGroupSize = 3 y la matriz de series contiene 3 elementos: c3, c4 y c5.

El segundo y tercer campo entero representan el número de valores frecuentes y el número de cuantiles respectivamente cuando se recopilan estadísticas de distribución para grupos de columnas. Esta funcionalidad no está disponible actualmente.

Se recopilan estadísticas básicas de columna para las columnas que aparecen en piColumnGroupList y que no aparecen en piColumnList. Esto produce el mismo efecto que si esas columnas se hubieran incluido en piColumnList. Si iNumColGroups es cero, se ignora piColumnGroupList.

piIndexList
Entrada. Matriz de series. Cada serie contiene un nombre de índice totalmente calificado. Si NumIndexes es cero, se ignora piIndexList.

iRunstatsFlags
Entrada. Campo de máscara de bits utilizado para especificar opciones estadísticas. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2RUNSTATS_ALL_COLUMNS
Recopilar estadísticas para todas las columnas de la tabla o vista estadística. Esta opción se puede especificar en combinación con listas de columnas, distribución de columnas, grupos de columnas o de estructuras de índice. Esto es útil si desea recopilar estadísticas para todas las columnas de la tabla o vista, y al mismo tiempo proporcionar opciones estadísticas para determinadas columnas.

DB2RUNSTATS_KEY_COLUMNS
Recopilar estadísticas solamente para las columnas que conforman todos los índices definidos en la tabla. Esta opción no se puede utilizar para vistas estadísticas. Para tablas, esta opción se puede especificar en combinación con listas de columnas, distribución de columnas, grupos de columnas o de estructuras de índice. Esto es útil si desea recopilar estadísticas para todas las columnas de clave de la tabla y al mismo tiempo recopilar estadísticas para algunas columnas no de clave, o si desea proporcionar opciones estadísticas para determinadas columnas de clave. Por definición, las columnas de tipo XML no son columnas de clave y no se incluirán para la recopilación de estadísticas cuando el valor del parámetro iRunstatsFlags sea DB2RUNSTATS_KEY_COLUMNS.

DB2RUNSTATS_DISTRIBUTION
Recopilar estadísticas de distribución. Esta opción solamente se puede utilizar con DB2RUNSTATS_ALL_COLUMNS y DB2RUNSTATS_KEY_COLUMNS. Cuando se utiliza con DB2RUNSTATS_ALL_COLUMNS, se recopilan estadísticas de distribución para todas las columnas de la tabla o vista estadística. Cuando se utiliza con DB2RUNSTATS_KEY_COLUMNS, se recopilan estadísticas para todas las columnas que conforman

todos los índices definidos en la tabla. Cuando se utiliza al mismo tiempo con DB2RUNSTATS_ALL_COLUMNS y DB2RUNSTATS_KEY_COLUMNS, se recopilan estadísticas básicas para todas las columnas de la tabla y se recopilan estadísticas de distribución solamente para las columnas que conforman todos los índices definidos en la tabla.

DB2RUNSTATS_ALL_INDEXES
Recopilar estadísticas para todos los índices definidos en la tabla. Esta opción no se puede utilizar para vistas estadísticas.

DB2RUNSTATS_EXT_INDEX
Recopilar estadísticas de índice detalladas. Esta opción se debe especificar con DB2RUNSTATS_ALL_INDEXES o con una lista explícita de nombres de índice (piIndexList y iNumIndexes > 0). Esta opción no se puede utilizar para vistas estadísticas.

DB2RUNSTATS_EXT_INDEX_SAMPLED
Recopilar estadísticas de índice detalladas utilizando métodos de ejemplo. Esta opción se debe especificar con DB2RUNSTATS_ALL_INDEXES o con una lista explícita de nombres de índice (piIndexList y iNumIndexes > 0). Se ignorará DB2RUNSTATS_EXT_INDEX si se especifica al mismo tiempo. Esta opción no se puede utilizar para vistas estadísticas.

DB2RUNSTATS_ALLOW_READ
Permite el acceso de solo lectura mientras se recopilan las estadísticas. El valor por omisión es permitir el acceso de lectura y escritura.

DB2RUNSTATS_SAMPLING_SYSTEM
Recopilar estadísticas para un porcentaje de las páginas de datos según lo especificado por el usuario en el parámetro iSamplingOption. El tipo de muestreo SYSTEM trata cada página individualmente, incluyendo la página con una probabilidad de P/100 (donde P es el valor de iSamplingOption) y excluyéndola con una probabilidad de 1-P/100. De esta manera, si SamplingOption es el valor 10, lo que representa una muestra del 10 por ciento, cada página se incluirá con una probabilidad de 0,1 y se excluirá con una probabilidad de 0,9.

El tipo de muestreo SYSTEM no se puede especificar para vistas estadísticas. Solamente se puede utilizar el muestreo BERNOULLI para muestrear datos de vista.

Si no se especifica DB2RUNSTATS_SAMPLING_SYSTEM, DB2 asume que se va a utilizar el muestreo BERNOULLI como método de muestreo. El muestreo BERNOULLI considera cada fila individualmente, incluyendo la fila con la probabilidad P/100 (donde P es el valor de iSamplingOption) y excluyéndola con la probabilidad 1-P/100.

Tanto en el muestreo SYSTEM como BERNOULLI, a menos que se especifique el distintivo DB2RUNSTATS_SAMPLING_REPEAT, cada ejecución de la recopilación de estadísticas produce habitualmente una muestra diferente de la tabla o vista estadística.

DB2RUNSTATS_SAMPLING_REPEAT
Especifica que se ha pasado un valor generador mediante el parámetro iSamplingRepeatable. El valor de iSamplingRepeatable

se utilizará como valor generador para producir la muestra de datos. También se debe especificar el parámetro iSamplingOption para indicar la frecuencia de muestreo.

DB2RUNSTATS_USE_PROFILE
Recopilar estadísticas para una tabla o vista estadística utilizando un perfil estadístico previamente registrado en los catálogos de la tabla o vista. Si se especifica la opción USE PROFILE mediante este distintivo definido en la máscara de bits iRunstatsFlags, no se tienen en cuenta las demás opciones contenidas en db2RunstatsData.

DB2RUNSTATS_SET_PROFILE
Generar y almacenar un perfil en los catálogos para registrar las opciones estadísticas especificadas y recopilar estadísticas utilizando esas mismas opciones.

DB2RUNSTATS_SET_PROFILE_ONLY
Generar y almacenar un perfil en los catálogos para registrar las opciones estadísticas especificadas, pero sin recopilar estadísticas para la tabla o vista.

DB2RUNSTATS_UNSET_PROFILE
Si se suprime la definición de un perfil de estadísticas, se eliminará el perfil de estadísticas de los catálogos del sistema estableciendo SYSCAT.STATISTICS_PROFILE en NULL. Si no existe un perfil de estadísticas, el intento de suprimir la definición dará como resultado un error (SQLCODE -2315).

DB2RUNSTATS_UPDATE_PROFILE
Modificar un perfil estadístico existente en los catálogos y recopilar estadísticas utilizando las opciones contenidas en el perfil actualizado.

DB2RUNSTATS_UPDA_PROFILE_ONLY
Modificar un perfil estadístico existente en los catálogos, pero sin recopilar estadísticas para la tabla o vista.

DB2RUNSTATS_EXCLUDING_XML
No recopilar estadísticas para columnas de tipo XML. Se recopilarán estadísticas para todas las columnas especificadas cuyo tipo no sea XML. Esta opción tiene preferencia sobre todos los demás métodos donde se especifican columnas XML.

iNumColumns
Entrada. Número de elementos especificados en la lista piColumnList.

iNumColdist
Entrada. Número de elementos especificados en la lista piColumnDistributionList.

iNumColGroups
Entrada. Número de elementos especificados en la lista piColumnGroupList.

iNumIndexes
Entrada. Número de elementos especificados en la lista piIndexList.

iParallelismOption
Entrada. Reservado para una utilización futura. El valor válido es 0.

iTableDefaultFreqValues
Entrada. Especifica el número predefinido de valores frecuentes que se deben recopilar para la tabla o vista. Los valores válidos son:

n Se recopilarán n valores frecuentes a menos que se especifique otra cosa a nivel de columna.

0 No se recopilarán valores frecuentes a menos que se especifique otra cosa a nivel de columna.

-1 Se utiliza el parámetro de configuración por omisión NUM_FREQVALUES de la base de datos para indicar el número de valores frecuentes que se deben recopilar.

iTableDefaultQuantiles
Entrada. Especifica el número por omisión de valores cuantiles que se deben recopilar para la tabla o vista. Los valores válidos son:

n Se recopilarán n cuantiles a menos que se especifique otra cosa a nivel de columna.

0 No se recopilarán cuantiles a menos que se especifique otra cosa a nivel de columna.

-1 Se utiliza el parámetro de configuración por omisión NUM_QUANTILES de la base de datos para indicar el número de cuantiles que se deben recopilar.

iSamplingRepeatable
Entrada. Valor entero no negativo que representa el valor generador que se debe utilizar en el muestreo de tablas o vistas. Si se pasa un generador negativo se producirá un error (SQL1197N).

Se debe definir el distintivo DB2RUNSTATS_SAMPLING_REPEAT para utilizar este valor generador. Esta opción se utiliza en combinación con el parámetro iSamplingOption para generar la misma muestra de datos en recopilaciones de datos estadísticos subsiguientes. La muestra puede todavía variar entre peticiones repetibles si la actividad realizada en la tabla o vista produce cambios en los datos de la tabla o vista desde la última vez que se ejecutó una petición repetible. Además, el método utilizado para obtener la muestra (BERNOULLI o SYSTEM) debe también ser el mismo para asegurar resultados coherentes.

iUtilImpactPriority
Entrada. Prioridad para la invocación de runstats. Los valores válidos deben estar dentro del rango de 0 a 100, donde 70 representa la modalidad no regulada de invocación y 100 representa la prioridad más alta posible. Esta opción no se puede utilizar para vistas estadísticas.

Parámetros de la estructura de datos db2ColumnData

piColumnName
Entrada. Puntero a una serie que representa un nombre de columna.

iColumnFlags
Entrada. Campo de máscara de bits que se utiliza para especificar opciones de estadísticas para la columna. Los valores válidos son:

DB2RUNSTATS_COLUMN_LIKE_STATS
Recopilar estadísticas LIKE para la columna.

Parámetros de la estructura de datos db2ColumnDistData

piColumnName
Entrada. Puntero a una serie que representa un nombre de columna.

iNumFreqValues
Entrada. Número de valores frecuentes que se deben recopilar para la columna. Los valores válidos son:

n Recopilar n valores frecuentes para la columna.

-1 Utilizar el número predefinido de valores frecuentes para la tabla, tal como iTableDefaultFreqValues si está definido, o el parámetro de configuración NUM_FREQVALUES.

iNumQuantiles
Entrada. Número de cuantiles que se deben recopilar para la columna. Los valores válidos son:

n Recopilar n cuantiles para la columna.

-1 Utilizar el número predefinido de cuantiles para la tabla, tal como iTableDefaultQuantiles si está definido, o el parámetro de configuración NUM_QUANTILES de la base de datos.

Parámetros de la estructura de datos db2ColumnGrpData

piGroupColumnNames
Entrada. Matriz de series. Cada serie representa un nombre de columna que forma parte del grupo de columnas para el que se deben recopilar estadísticas.

iGroupSize
Entrada. Número de columnas en el grupo de columnas. Los valores válidos son:

n El grupo de columnas consta de n columnas.

iNumFreqValues
Entrada. Reservado para una utilización futura.

iNumQuantiles
Entrada. Reservado para una utilización futura.

Parámetros específicos de la estructura de datos db2gRunstatsData

piIndexNamesLen
Entrada. Matriz de valores que representan la longitud, en bytes, de cada nombre de columna contenido en la lista de nombres de columna. Si NumIndexes es cero, se omite piIndexNamesLen.

iTablenameLen
Entrada. Valor que representa la longitud, en bytes, del nombre de tabla o vista.

Parámetros específicos de la estructura de datos db2gColumnData

iColumnNameLen
Entrada. Valor que representa la longitud, en bytes, del nombre de columna.

Parámetros específicos de la estructura de datos db2gColumnDistData

iColumnNameLen
Entrada. Valor que representa la longitud, en bytes, del nombre de columna.

Parámetros específicos de la estructura de datos db2gColumnGrpData

piGroupColumnNamesLen
Entrada. Matriz de valores que representan la longitud, en bytes, de cada nombre de columna contenido en la lista de nombres de columna.

Notas de uso

Utilice db2Runstats para actualizar estadísticas:

- Para tablas que se han modificado muchas veces (por ejemplo, si se han realizado muchas actualizaciones, o se ha insertado o suprimido una cantidad importante de datos)
- Para tablas que se han reorganizado
- Cuando se ha creado un índice nuevo.
- Para vistas basadas en tablas que se han modificado sustancialmente, a fin de cambiar las filas devueltas por la vista.

Después de actualizar estadísticas, se pueden crear nuevas vías de acceso para la tabla volviendo a vincular los paquetes mediante sqlabndx - Bind.

Si se solicitan estadísticas de índice, y no se han ejecutado nunca estadísticas para la tabla donde reside el índice, se calculan estadísticas tanto para la tabla como para los índices.

Si la API db2Runstats está recogiendo solamente estadísticas para índices, se conservan las estadísticas de distribución recopiladas anteriormente. De lo contrario, la API descartará las estadísticas de distribución recopiladas anteriormente. Si la API db2Runstats está recogiendo solamente estadísticas para columnas XML, se conservan las estadísticas básicas de columna y las estadísticas de distribución recopiladas anteriormente. En el caso en el que se han recopilado anteriormente estadísticas para algunas columnas XML, esas estadísticas se eliminan si la llamada actual a la API db2Runstats no recopila ninguna estadística para esa columna XML, o las estadísticas se sustituyen si la llamada actual a la API db2Runstats recopila estadísticas para la columna XML. De lo contrario, la API descartará las estadísticas de distribución recopiladas anteriormente.

Si el parámetro iRunstatsFlags se establece en el valor DB2RUNSTATS_EXCLUDING_XML, no se recopilan estadísticas para columnas XML. Este valor tiene prioridad sobre todos los demás métodos donde se especifiquen columnas XML.

Después de invocar esta API, la aplicación debe emitir una sentencia COMMIT para liberar los bloqueos.

Para permitir que se generen nuevos planes de acceso, los paquetes que hacen referencia a la tabla de destino se deben volver a vincular después de llamar a esta

API. Los paquetes que contienen consultas que pueden sacar provecho de las vistas estadísticas también se deben volver a vincular después de actualizar estadísticas para esas vistas.

Cuando se recopilan estadísticas para vistas estadísticas, se ejecuta una consulta SQL internamente. Se puede utilizar el programa EXPLAIN para examinar el plan de acceso seleccionado para la consulta e investigar los problemas de rendimiento que pueda haber en la recopilación de estadísticas. Para guardar el plan de acceso de la consulta en las tablas de EXPLAIN, establezca el registro especial CURRENT EXPLAIN MODE en YES.

La ejecución de esta API solamente para la tabla puede producir una situación en la que las estadísticas a nivel de tabla son incoherentes con las estadísticas a nivel de índice ya existentes. Por ejemplo, si se recopilan estadísticas a nivel de índice para una tabla determinada y más tarde se suprime un número significativo de filas de la tabla, la ejecución de esta API para la tabla solamente puede hacer que la cardinalidad de la tabla sea menor que FIRSTKEYCARD (FIRSTKEYCARD es un campo estadístico de catálogo contenido en las vistas de catálogo SYSCAT.INDEXES y SYSSTAT.INDEXES) lo cual es un estado no coherente. De la misma manera, ejecutar esta API para índices solamente puede dejar en un estado no coherente a las estadísticas existentes a nivel de tabla. Por ejemplo, si se recopilan estadísticas a nivel de tabla para una tabla determinada y más tarde se suprime un número importante de filas de la tabla, la ejecución de la API db2Runstats solamente para los índices puede hacer que algunas columnas tengan un campo COLCARD (COLCARD es un campo estadístico de catálogo contenido en las vistas de catálogo SYSCAT.COLUMNS y SYSSTAT.COLUMNS) mayor que la cardinalidad de la tabla. Si se detecta esta incoherencia, se emite un aviso.

db2SelectDB2Copy - Seleccionar la copia de DB2 que la aplicación utiliza

Establece el entorno necesario para que la aplicación utilice una copia de DB2 determinada o la ubicación especificada. Si el entorno ya está definido para la copia de DB2 que desea utilizar, no es necesario que invoque esta API. Sin embargo, si necesita utilizar una copia de DB2 diferente, debe invocar esta API. Invóquela antes de cargar los archivos dll de DB2 en el proceso. Esta llamada sólo se puede realizar una vez por proceso.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiInstall.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2SelectDB2Copy (
    db2Uint32 versionNumber,
    void *pDB2SelectDB2CopyStruct);
```

```
typedef enum DB2CopyParmType
{
   DB2CopyInvalid=0,
   DB2CopyName,
   DB2CopyPath
} db2CopyParmType;

typedef struct DB2SelectDB2CopyStruct
{
   DB2CopyParmType  Type;
   char *psziDB2Copy;
} db2SelectDB2CopyStruct
```

Parámetros de la API db2SelectDB2Copy

versionNumber
Entrada. Especifica el número de versión y el nivel de release de la variable transferida como segundo parámetro, pDB2SelectInstallationStruct.

pDB2SelectDB2CopyStruct
Entrada. Puntero a la estructura DB2SelectDB2CopyStruct.

Parámetros de la estructura de datos DB2SelectDB2CopyStruct

Tipo Entrada. Puede ser DB2CopyName o DB2CopyPath.

psziDB2Copy
Entrada. Si se especifica Type como DB2CopyName, psziDB2Copy es el nombre de la copia de DB2. Si se especifica Type como db2CopyPath, psziDB2Copy es la vía de acceso de instalación de DB2. No puede tener un valor NULL.

Notas de uso

Para utilizar la API, deberá incluir db2ApiInstall.h, que obligará a la aplicación a enlazarse estáticamente en db2ApiInstall.lib.

Además, esta API se debe invocar antes de cargar las bibliotecas de DB2 y una aplicación sólo puede invocarla una vez. Puede evitar cargar bibliotecas de DB2 haciendo uso de la opción /delayload al enlazar las bibliotecas de DB2 o bien puede cargar estas bibliotecas dinámicamente mediante LoadLibraryEx y especificando LOAD_WITH_ALTERED_SEA.

db2SetSyncSession - Establecer sesión de sincronización de satélites

Establece la sesión de sincronización para un satélite. La sesión de sincronización se asocia con la versión de la aplicación de usuario que se ejecuta en el satélite. Cada versión de una aplicación trabaja con una determinada configuración de base de datos y maneja determinados archivos, cada uno de los cuales se puede sincronizar con una ubicación central.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2SetSyncSession (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef struct db2SetSyncSessionStruct
{
   char *piSyncSessionID;
} db2SetSyncSessionStruct;
```

Parámetros de la API db2SetSyncSession

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2SetSyncSessionStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2SetSyncSessionStruct

piSyncSessionID
Entrada. Especifica un identificador para la sesión de sincronización utilizada actualmente por un satélite. El valor especificado debe coincidir con la versión de aplicación apropiada correspondiente al grupo del satélite, tal como está definido en el servidor de control de satélites.

db2SetWriteForDB - Suspender o reanudar las escrituras de E/S para la base de datos

Suspende o reanuda las escrituras de E/S en disco para la base de datos. Es necesario suspender las escrituras de E/S para una base de datos antes de realizar una copia instantánea. Para evitar posibles problemas, mantenga la misma conexión para realizar la suspensión y reanudación de la escritura.

Ámbito

Esta API solamente afecta a la partición de base de datos en la que se ejecuta.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2SetWriteForDB (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef struct db2SetWriteDbStruct
{
   db2int32 iOption;
   char *piTablespaceNames;
} db2SetWriteDbStruct;
```

Parámetros de la API db2SetWriteForDB

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2SetWriteDbStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2SetWriteDbStruct

iOption
Entrada. Especifica la acción. Los valores válidos son:

- **DB2_DB_SUSPEND_WRITE**
  Suspende la escritura de E/S en disco.
- **DB2_DB_RESUME_WRITE**
  Reanuda la escritura de E/S en disco.

piTablespaceNames
Entrada. Reservado para una utilización futura.

db2SpmListIndTrans - Listar transacciones dudosas SPM

Proporciona una lista de las transacciones que son dudosas en el Gestor de punto de sincronismo.

Ámbito

Esta API sólo afecta a la partición de base de datos en la que se emite.

Autorización

Ninguna

Conexión necesaria

Conexión con el Gestor de punto de sincronismo

Archivo de inclusión de la API

sqlxa.h

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
db2SpmListIndTrans (
  db2Uint32 versionNumber,
  void * pParmStruct,
  struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2SpmListIndTransStruct
{
db2SpmRecoverStruct * piIndoubtData;
db2Uint32            iIndoubtDataLen;
db2Uint32            oNumIndoubtsReturned;
db2Uint32            oNumIndoubtsTotal;
db2Uint32            oReqBufferLen;
} db2XaListIndTransStruct;

typedef SQL_STRUCTURE db2SpmRecoverStruct
{
   SQLXA_XID    xid;
   char         luwid[SQLCSPQY_LUWID_SZ+1];
   char         corrtok[SQLCSPQY_APPLID_SZ+1];
   char         partner[SQLCSPQY_LUNAME_SZ+1];
   char         dbname[SQLCSPQY_DBNAME_SZ+1];
   char         dbalias[SQLCSPQY_DBNAME_SZ+1];
   char         role;
   char         uow_status;
   char         partner_status;
} db2SpmRecoverStruct;
```

Parámetros de la API db2SpmListIndTrans

versionNumber
Entrada. Especifica la versión y el nivel de release.

pParmStruct
Entrada. Puntero a la estructura db2SpmListIndTransStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2SpmListIndTransStruct

piIndoubtData
Entrada. Puntero al almacenamiento intermedio suministrado por la aplicación al que se devolverán los datos dudosos. Los datos dudosos se encuentran en el formato db2SpmRecoverStruct. La aplicación puede recorrer la lista de transacciones dudosas mediante el tamaño de la estructura db2SpmRecoverStruct, empezando por la dirección suministrada por este parámetro. Si el valor es NULL, se calcula el tamaño del almacenamiento intermedio necesario y se devuelve en oReqBufferLen. oNumIndoubtsTotal contendrá el número total de transacciones dudosas. La aplicación puede asignar el tamaño de almacenamiento intermedio necesario y emitir de nuevo la API.

oNumIndoubtsReturned
Salida. El número de registros de transacciones dudosas devuelto en el almacenamiento intermedio especificado por pIndoubtData.

oNumIndoubtsTotal
Salida. El número total de registros de transacciones dudosas disponible en

el momento de la invocación. Si el almacenamiento intermedio `piIndoubtData` es demasiado pequeño para incluir todos los registros, `oNumIndoubtsTotal` será mayor que el total de `oNumIndoubtsReturned`. La aplicación puede volver a emitir la API para poder obtener todos los registros.

Este número puede cambiar entre invocaciones de la API como resultado de la resincronización de transacciones dudosas automáticas o heurísticas, o bien como resultado de que otras transacciones acceden al estado dudoso.

oReqBufferLen
Salida. Longitud del almacenamiento intermedio necesario para contener todos los registros de transacciones dudosas en el momento de la invocación de la API. La aplicación puede utilizar este valor para determinar el tamaño del almacenamiento intermedio necesario invocando la API con `pIndoubtData` establecido en `NULL`. A continuación, este valor se puede utilizar para asignar el almacenamiento intermedio y la API se puede emitir con `pIndoubtData` establecido en la dirección del almacenamiento intermedio asignado.

El tamaño del almacenamiento intermedio necesario puede cambiar entre invocaciones de la API como resultado de la resincronización de transacciones dudosas automáticas o heurísticas o como resultado de que otras transacciones acceden al estado dudoso. La aplicación puede asignar un almacenamiento mayor para tener esto en cuenta.

Parámetros de estructura de datos de db2SpmRecoverStruct

xid Salida. Especifica el identificador XA asignado por el gestor de transacciones para identificar de forma exclusiva una transacción global.

luwid Salida. Especifica el ID de la unidad de trabajo lógica (LUWID) asignada por el Gestor de punto de sincronismo para identificar el identificador XA (XID) en el sistema asociado.

corrtok
Salida. Especifica el identificador de aplicación asignado por el gestor de punto de sincronismo para esta transacción.

partner
Salida. Especifica el nombre del sistema asociado.

dbname
Salida. Base de datos del sistema asociado

dbalias
Salida. Especifica el alias de la base de datos en la que se encuentra la transacción dudosa.

role Salida. Rol del gestor de punto de sincronismo.

SQLCSPQY_AR
El gestor de punto de sincronismo es un peticionario de aplicaciones

SQLCSPQY_AS
El gestor de punto de sincronismo es un servidor de aplicaciones

uow_status
Salida. Indica el estado de esta transacción dudosa en el gestor de punto de sincronismo. Los valores válidos son:

SQLCSPQY_STATUS_COM
La transacción está en estado de confirmación en el gestor de punto de sincronismo. La transacción está esperando resincronizarse con el sistema asociado durante el próximo intervalo de resincronización.

SQLCSPQY_STATUS_RBK
La transacción está en estado de retrotracción en el gestor de punto de sincronismo. Esperando a que el sistema asociado inicie la resincronización y resuelva las dudas.

SQLCSPQY_STATUS_IDB
La transacción está en estado de preparación en el gestor de punto de sincronismo. El parámetro conectado se puede utilizar para determinar si la transacción está esperando la segunda fase del proceso de confirmación normal o si se ha producido un error y es necesaria la resincronización con el gestor de transacciones.

SQLCSPQY_STATUS_HCM
La transacción se ha confirmado heurísticamente.

SQLCSPQY_STATUS_HRB
La transacción se ha retrotraído heurísticamente.

Notas de uso

Una aplicación típica realizará los pasos siguientes después de establecer la conexión actual con el Gestor de punto de sincronismo*:

1. Invoque la API `db2SpmListIndTrans` con `piIndoubtData` establecido en el valor `NULL`. Se devolverán valores en `oReqBufferLen` y en `oNumIndoubtsTotal`.
2. Utilice el valor devuelto en `oReqBufferLen` para asignar un almacenamiento intermedio. Es posible que este almacenamiento intermedio sea suficientemente grande si hay más transacciones dudosas adicionales debido a la invocación inicial de esta API para obtener `oReqBufferLen`. La aplicación puede proporciona un almacenamiento intermedio mayor que `oReqBufferLen`.
3. Determine si se han obtenido todos los registros de transacciones dudosas. Esto se puede conseguir comparando `oNumIndoubtsReturned` con `oNumIndoubtsTotal`. Si `oNumIndoubtsTotal` es mayor que `oNumIndoubtsReturned`, la aplicación puede repetir los pasos anteriores.

* Para conectar con el Gestor de punto de sincronismo, determine el nombre del Gestor de punto de sincronismo que se esté utilizando en el servidor de DB2 Connect. Se puede determinar consultando el parámetro de configuración de la base de datos, `spm_name`, en el servidor de DB2 Connect. Emita una conexión especificando `spm_name` como alias de la base de datos en la API de conexión (connect).

db2SyncSatellite - Iniciar sincronización de satélites

Sincroniza un satélite. La sincronización de satélites supone poner un satélite en un estado que sea coherente con los demás satélites de su grupo.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2SyncSatellite (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);
```

Parámetros de la API db2SyncSatellite

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Su valor se establece en NULL.

pSqlca
Salida. Puntero a la estructura sqlca.

db2SyncSatelliteStop - Pausar sincronización de satélites

Detiene la sesión de sincronización actualmente activa del satélite. La sesión se detiene de tal forma que se puede invocar db2SyncSatellite para reiniciar la sincronización del satélite desde donde se detuvo.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2SyncSatelliteStop (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);
```

Parámetros de la API db2SyncSatelliteStop

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Su valor se establece en NULL.

pSqlca
Salida. Puntero a la estructura sqlca.

db2SyncSatelliteTest - Probar si se puede sincronizar un satélite

Prueba la capacidad de un satélite para sincronizarse, es decir, prueba si el satélite se puede colocar en un estado que sea coherente con los demás satélites de su grupo.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2SyncSatelliteTest (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);
```

Parámetros de la API db2SyncSatelliteTest

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Su valor se establece en NULL.

pSqlca
Salida. Puntero a la estructura sqlca.

db2UpdateAlertCfg - Actualizar los valores de configuración de alertas para los indicadores de salud

Actualiza los valores de configuración de alertas para los indicadores de salud.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint

Conexión necesaria

Instancia. Si no existe ninguna conexión de instancia, se creará una conexión de instancia por omisión.

Archivo de inclusión de la API

db2ApiDf.h

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2UpdateAlertCfg (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2UpdateAlertCfgData
{
   db2Uint32 iObjType;
   char *piObjName;
   char *piDbName;
   db2Uint32 iIndicatorID;
   db2Uint32 iNumIndAttribUpdates;
   struct db2AlertAttrib *piIndAttribUpdates;
   db2Uint32 iNumActionUpdates;
   struct db2AlertActionUpdate *piActionUpdates;
   db2Uint32 iNumActionDeletes;
   struct db2AlertActionDelete *piActionDeletes;
   db2Uint32 iNumNewActions;
   struct db2AlertActionNew *piNewActions;
} db2UpdateAlertCfgData;

typedef SQL_STRUCTURE db2AlertAttrib
{
   db2Uint32 iAttribID;
   char *piAttribValue;
} db2AlertAttrib;

typedef SQL_STRUCTURE db2AlertActionUpdate
{
   db2Uint32 iActionType;
   char *piActionName;
   db2Uint32 iCondition;
   db2Uint32 iNumParmUpdates;
   struct db2AlertAttrib *piParmUpdates;
} db2AlertActionUpdate;

typedef SQL_STRUCTURE db2AlertActionDelete
{
   db2Uint32 iActionType;
   char *piName;
   db2Uint32 iCondition;
} db2AlertActionDelete;

typedef SQL_STRUCTURE db2AlertActionNew
{
   db2Uint32 iActionType;
   struct db2AlertScriptAction *piScriptAttribs;
   struct db2AlertTaskAction *piTaskAttribs;
} db2AlertActionNew;

typedef SQL_STRUCTURE db2AlertScriptAction
{
   db2Uint32 scriptType;
   db2Uint32 condition;
   char *pPathName;
   char *pWorkingDir;
   char *pCmdLineParms;
   char stmtTermChar;
   char *pUserID;
   char *pPassword;
   char *pHostName;
```

```
} db2AlertScriptAction;

typedef SQL_STRUCTURE db2AlertTaskAction
{
   char *pTaskName;
   db2Uint32 condition;
   char *pUserID;
   char *pPassword;
   char *pHostName;
} db2AlertTaskAction;
```

Parámetros de la API db2UpdateAlertCfg

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2UpdateAlertCfgData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2UpdateAlertCfgData

iObjType
Entrada. Especifica el tipo de objeto para el que se solicita la configuración. Los valores válidos son:

- DB2ALERTCFG_OBJTYPE_DBM
- DB2ALERTCFG_OBJTYPE_DATABASES
- DB2ALERTCFG_OBJTYPE_TABLESPACES
- DB2ALERTCFG_OBJTYPE_TS_CONTAINERS
- DB2ALERTCFG_OBJTYPE_DATABASE
- DB2ALERTCFG_OBJTYPE_TABLESPACE
- DB2ALERTCFG_OBJTYPE_TS_CONTAINER

piObjName
Entrada. Nombre del espacio de tablas o contenedor de espacio de tablas cuando el tipo de objeto, iObjType, es DB2ALERTCFG_OBJTYPE_TABLESPACE o DB2ALERTCFG_OBJTYPE_TS_CONTAINER. De lo contrario, su valor es NULL.

piDbName
Entrada. Alias de la base de datos para la que se solicitan la configuración cuando el tipo de objeto, iObjType, es DB2ALERTCFG_OBJTYPE_TS_CONTAINER, DB2ALERTCFG_OBJTYPE_TABLESPACE o DB2ALERTCFG_OBJTYPE_DATABASE. En caso contrario, su valor es NULL.

iIndicatorID
Entrada. Indicador de salud para el que se deben aplicar las actualizaciones de la configuración.

iNumIndAttribUpdates
Entrada. Número de atributos de alerta que se deben actualizar para el indicador de salud iIndicatorID.

piIndAttribUpdates
Entrada. Puntero a la estructura db2AlertAttrib.

iNumActionUpdates
Entrada. Número de acciones de alerta que se debe actualizar para el indicador de salud iIndicatorID.

piActionUpdates
Entrada. Puntero a la estructura db2AlertActionUpdate.

iNumActionDeletes
Entrada. Número de acciones de alerta que se debe suprimir del indicador de salud iIndicatorID.

piActionDeletes
Entrada. Puntero a la estructura db2AlertActionDelete.

iNumNewActions
Entrada. Número de nuevas acciones de alerta que se deben añadir al indicador de salud iIndicatorID.

piNewActions
Entrada. Puntero a la estructura db2AlertActionNew.

Parámetros de la estructura de datos db2AlertAttrib

iAttribID
Entrada. Especifica el atributo de alerta que se actualizará. Los valores válidos son:

- DB2ALERTCFG_ALARM
- DB2ALERTCFG_WARNING
- DB2ALERTCFG_SENSITIVITY
- DB2ALERTCFG_ACTIONS_ENABLED
- DB2ALERTCFG_THRESHOLD_CHECK

piAttribValue
Entrada. El nuevo valor del atributo de alerta. Los valores válidos son:

- DB2ALERTCFG_ALARM
- DB2ALERTCFG_WARNING
- DB2ALERTCFG_SENSITIVITY
- DB2ALERTCFG_ACTIONS_ENABLED
- DB2ALERTCFG_THRESHOLD_CHECK

Parámetros de la estructura de datos db2AlertActionUpdate

iActionType
Entrada. Especifica la acción de alerta. Los valores válidos son:

- DB2ALERTCFG_ACTIONTYPE_SCRIPT
- DB2ALERTCFG_ACTIONTYPE_TASK

piActionName
Entrada. Nombre de la acción de alerta. El nombre de una acción de script es el nombre de vía de acceso absoluta del script. El nombre de una acción de tarea es una serie en el formato: <ID-numérico-tarea>.<sufijo-numérico-tarea>.

iCondition
Condición en la que se ejecuta la acción. Los valores válidos para los indicadores de salud basados en valores umbrales son:

- DB2ALERTCFG_CONDITION_ALL
- DB2ALERTCFG_CONDITION_WARNING
- DB2ALERTCFG_CONDITION_ALARM

Para los indicadores de salud basados en estados, utilice el valor numérico definido en sqlmon.

iNumParmUpdates
Entrada. Número de atributos de acción que se deben actualizar en la matriz piParmUpdates.

piParmUpdates
Entrada. Puntero a la estructura db2AlertAttrib.

Parámetros de la estructura de datos db2AlertActionDelete

iActionType
Entrada. Especifica la acción de alerta. Los valores válidos son:

- DB2ALERTCFG_ACTIONTYPE_SCRIPT
- DB2ALERTCFG_ACTIONTYPE_TASK

piName
Entrada. Nombre de la acción de alerta o la acción de script. El nombre de la acción de script es el nombre de vía de acceso absoluta, siempre que el nombre de la acción de tarea sea una serie en formato: <ID-numérico-tarea>.<sufijo-numérico-tarea>.

iCondition
Condición en la que se ejecuta la acción. Los valores válidos para los indicadores de salud basados en valores umbrales son:

- DB2ALERTCFG_CONDITION_ALL
- DB2ALERTCFG_CONDITION_WARNING
- DB2ALERTCFG_CONDITION_ALARM

Para los indicadores de salud basados en estados, utilice el valor numérico definido en sqlmon.

Parámetros de la estructura de datos db2AlertActionNew

iActionType
Entrada. Especifica la acción de alerta. Los valores válidos son:

- DB2ALERTCFG_ACTIONTYPE_SCRIPT
- DB2ALERTCFG_ACTIONTYPE_TASK

piScriptAttribs
Entrada. Puntero a la estructura db2AlertScriptAction.

piTaskAttribs
Entrada. Puntero a la estructura db2AlertTaskAction.

Parámetros de la estructura de datos db2AlertScriptAction

scriptType

Especifica el tipo de script. Los valores válidos son:

- DB2ALERTCFG_SCRIPTTYPE_DB2CMD

- DB2ALERTCFG_SCRIPTTYPE_OS

condition
Condición en la que se ejecuta la acción. Los valores válidos para los indicadores de salud basados en valores umbrales son:

- DB2ALERTCFG_CONDITION_ALL
- DB2ALERTCFG_CONDITION_WARNING
- DB2ALERTCFG_CONDITION_ALARM

Para los indicadores de salud basados en estados, utilice el valor numérico definido en sqlmon.

pPathname
Vía de acceso absoluta del script.

pWorkingDir
Vía de acceso absoluta del directorio donde se debe ejecutar el script.

pCmdLineParms
Parámetros de línea de mandatos que se deben pasar al script cuando éste se invoque. Aplicable opcionalmente para DB2ALERTCFG_SCRIPTTYPE_OS solamente.

stmtTermChar
Carácter utilizado en el script para finalizar sentencias. Aplicable opcionalmente para DB2ALERTCFG_SCRIPTTYPE_DB2CMD solamente.

pUserID
Cuenta de usuario utilizada para ejecutar el script.

pPassword
Contraseña de la cuenta de usuario pUserId.

pHostName
Nombre del sistema principal en el que ejecutar el script. Esto es aplicable a la tarea y al script.

Script Nombre del sistema principal donde reside el script y desde donde se ejecutará.

Task Nombre del sistema principal donde reside el planificador de tareas.

Parámetros de la estructura de datos db2AlertTaskAction

pTaskname
Nombre de la tarea.

condition
Condición para la que se ejecuta la acción.

pUserID
Cuenta de usuario utilizada para ejecutar el script.

pPassword
Contraseña de la cuenta de usuario pUserId.

pHostName
Nombre del sistema principal en el que ejecutar el script. Esto es aplicable a la tarea y al script.

Script Nombre del sistema principal donde reside el script y desde donde se ejecutará.

Task Nombre del sistema principal donde reside el planificador de tareas.

db2UpdateAlternateServerForDB - Actualizar el servidor alternativo para un alias de base de datos en el directorio de bases de datos del sistema

Actualiza el servidor alternativo para un alias de base de datos en el directorio de bases de datos del sistema.

Ámbito

Esta API afecta al directorio de bases de datos del sistema.

Autorización

Una de las siguientes:

- sysadm
- sysctrl

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2UpdateAlternateServerForDB (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2UpdateAltServerStruct
{
   char *piDbAlias;
   char *piHostName;
   char *piPort;
} db2UpdateAltServerStruct;

SQL_API_RC SQL_API_FN
  db2gUpdateAlternateServerForDB (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gUpdateAltServerStruct
{
   db2Uint32 iDbAlias_len;
   char *piDbAlias;
   db2Uint32 iHostName_len;
   char *piHostName;
   db2Uint32 iPort_len;
   char *piPort;
} db2gUpdateAltServerStruct;
```

Parámetros de la API db2UpdateAlternateServerForDB

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2UpdateAltServerStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2UpdateAltServerStruct

piDbAlias
Entrada. Serie que contiene un alias para la base de datos.

piHostName
Entrada. Serie que contiene el nombre de sistema principal totalmente calificado o la dirección IP del nodo donde reside el servidor alternativo para la base de datos. El nombre de sistema principal es el nombre del nodo que se conoce en la red TCP/IP. La longitud máxima del nombre de sistema principal es de 255 caracteres. La dirección IP puede ser una dirección IPv4 o IPv6.

piPort Entrada. Número de puerto de la instancia del gestor de bases de datos del servidor alternativo. La longitud máxima del número de puerto es de 14 caracteres.

Parámetros específicos de la estructura de datos db2gUpdateAltServerStruct

iDbAlias_len
Entrada. Longitud en bytes de piDbAlias.

iHostName_len
Entrada. Longitud en bytes de piHostName.

iPort_len
Entrada. Longitud en bytes de piPort.

Notas de uso

La API solamente se aplica al directorio de bases de datos del sistema.

La API solamente se debe utilizar en un servidor. Si se emite en un cliente, no será efectiva y se emitirá el aviso SQL1889W.

Si se habilita el soporte de LDAP (Lightweight Directory Access Protocol) en la máquina actual, el servidor alternativo de la base de datos se actualizará automáticamente en el directorio LDAP.

db2UpdateContact - Actualizar los atributos de un contacto

Actualiza los atributos de un contacto. Los contactos son usuarios a los que se pueden enviar mensajes de notificación. Los contactos se pueden definir localmente en el sistema o en una lista global. El valor del parámetro de configuración contact_host del Servidor de administración de DB2 (DAS) determina si la lista es local o global.

Autorización

Ninguna

Conexión necesaria

Instancia. Si no existe ninguna conexión de instancia, se creará una conexión de instancia por omisión.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2UpdateContact (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2UpdateContactData
{
   char *piUserid;
   char *piPassword;
   char *piContactName;
   db2Uint32 iNumAttribsUpdated;
   struct db2ContactAttrib *piAttribs;
} db2UpdateContactData;

typedef SQL_STRUCTURE db2ContactAttrib
{
   db2Uint32 iAttribID;
   char *piAttribValue;
} db2ContactAttrib;
```

Parámetros de la API db2UpdateContact

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2UpdateContactData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2UpdateContactData

piContactName
Entrada. Especifica el nombre del contacto que se va a actualizar.

iNumAttribsUpdated
Entrada. Número de atributos que se deben actualizar.

piAttribs
Entrada. Puntero a la estructura db2ContactAttrib.

Parámetros de la estructura de datos db2ContactAttrib

iAttribID
Entrada. Especifica el atributo de contacto. Los valores válidos son:

- DB2CONTACT_ADDRESS
- DB2CONTACT_TYPE
- DB2CONTACT_MAXPAGELEN
- DB2CONTACT_DESCRIPTION

piAttribValue
Entrada. El nuevo valor del atributo del contacto.

Notas de uso

Esta API no está soportada en UNIX y Linux. No obstante, podrá acceder a la misma función utilizando la interfaz de SQL.

db2UpdateContactGroup - Actualizar los atributos de un grupo de contactos

Actualiza los atributos de un grupo de contactos. Un grupo de contactos contiene una lista de usuarios a los que se pueden enviar mensajes de notificación. Los grupos de contactos se pueden definir localmente en el sistema o en una lista global. El valor del parámetro de configuración contact_host del Servidor de administración de DB2 (DAS) determina si la lista es local o global.

Autorización

Ninguna.

Conexión necesaria

Ninguna.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2UpdateContactGroup (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2UpdateContactGroupData
{
   char *piUserid;
   char *piPassword;
   char *piGroupName;
   db2Uint32 iNumNewContacts;
   struct db2ContactTypeData *piNewContacts;
   db2Uint32 iNumDroppedContacts;
   struct db2ContactTypeData *piDroppedContacts;
   char *piNewDescription;
} db2UpdateContactGroupData;

typedef SQL_STRUCTURE db2ContactTypeData
{
   db2Uint32 contactType;
   char *pName;
} db2ContactTypeData;
```

Parámetros de la API db2UpdateContactGroup

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2ResetMonitorData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2UpdateContactGroupData

piUserid
Entrada. Nombre del usuario.

piPassword
Entrada. Contraseña de piUserid.

piGroupName
Entrada. Nombre del grupo de contactos que se debe actualizar.

iNumNewContacts
Entrada. Número de nuevos contactos que se deben añadir al grupo.

piNewContacts
Entrada. Puntero a la estructura db2ContactTypeData.

iNumDroppedContacts
Entrada. Número de contactos en el grupo que se debe descartar.

piDroppedContacts
Entrada. Puntero a la estructura db2ContactTypeData.

piNewDescription
Entrada. Nueva descripción del grupo. Establezca este parámetro en el valor NULL en caso de que la descripción antigua no deba cambiarse.

Parámetros de la estructura de datos db2ContactTypeData

contactType
Especifica el tipo de contacto. Los valores válidos son:

- DB2CONTACT_SINGLE
- DB2CONTACT_GROUP

pName
Nombre del grupo de contactos, o nombre del contacto si contactType es DB2CONTACT_SINGLE.

Notas de uso

Esta API no está soportada en UNIX y Linux. No obstante, podrá acceder a la misma función utilizando la interfaz de SQL.

db2UpdateHealthNotificationList - Actualizar la lista de contactos a los que se puedan enviar notificaciones de alerta de salud

Actualiza la lista de contactos de notificación para las alertas de estado emitidas por una instancia.

Autorización

Una de las siguientes:

- sysadm

- sysctrl
- sysmaint

Conexión necesaria

Instancia. Si no existe ninguna conexión de instancia, se creará una conexión de instancia por omisión.

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2UpdateHealthNotificationList (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2UpdateHealthNotificationListData
{
   db2Uint32 iNumUpdates;
   struct db2HealthNotificationListUpdate *piUpdates;
} db2UpdateHealthNotificationListData;

typedef SQL_STRUCTURE db2HealthNotificationListUpdate
{
   db2Uint32 iUpdateType;
   struct db2ContactTypeData *piContact;
} db2HealthNotificationListUpdate;

typedef SQL_STRUCTURE db2ContactTypeData
{
   db2Uint32 contactType;
   char *pName;
} db2ContactTypeData;
```

Parámetros de la API db2UpdateHealthNotificationList

versionNumber
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2UpdateHealthNotificationListData.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2UpdateHealthNotificationListData

iNumUpdates
Entrada. Número de actualizaciones.

piUpdates
Entrada. Puntero a la estructura db2HealthNotificationListUpdate.

Parámetros de la estructura de datos db2HealthNotificationListUpdate

iUpdateType
Entrada. Especifica el tipo de actualización. Los valores válidos son:

- DB2HEALTHNOTIFICATIONLIST_ADD
- DB2HEALTHNOTIFICATIONLIST_DROP

piContact
Entrada. Puntero a la estructura db2ContactTypeData.

Parámetros de la estructura de datos db2ContactTypeData

contactType
Especifica el tipo de contacto. Los valores válidos son:
- DB2CONTACT_SINGLE
- DB2CONTACT_GROUP

pName
Nombre del grupo de contactos, o nombre del contacto si contactType es DB2CONTACT_SINGLE.

db2UtilityControl - Establecer el nivel de prioridad de los programas de utilidad en ejecución

Controla el nivel de prioridad de los programas de utilidad en ejecución. Se puede utilizar para regular y desregular las invocaciones de programas de utilidad.

Autorización

Una de las siguientes:
- sysadm
- sysctrl
- sysmaint

Conexión necesaria

Instancia

Archivo de inclusión de la API

```
db2ApiDf.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2UtilityControl (
   db2Uint32 version,
   void * pUtilityControlStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2UtilityControlStruct
{
   db2Uint32 iID;
   db2Uint32 iAttribute;
   void *pioValue;
} db2UtilityControlStruct;

SQL_API_RC SQL_API_FN
  db2gUtilityControl (
   db2Uint32 version,
   void * pgUtilityControlStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gUtilityControlStruct
```

```
{
   db2Uint32 iID;
   db2Uint32 iAttribute;
   void *pioValue;
} db2gUtilityControlStruct;
```

Parámetros de la API db2UtilityControl

version
Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro, pUtilityControlStruct.

pUtilityControlStruct
Entrada. Puntero a la estructura db2UtilityControlStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2UtilityControlStruct

iId Entrada. Especifica el ID del programa de utilidad que se debe modificar.

iAttribute
Entrada. Especifica el atributo que se debe modificar. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio de inclusión) son:

DB2UTILCTRL_PRIORITY_ATTRIB
Modifica la prioridad de regulación del programa de utilidad.

pioValue
Entrada. Especifica el nuevo valor de atributo asociado con el parámetro iAttribute.

Nota: Si el parámetro iAttribute se establece en DB2UTILCTRL_PRIORITY_ATTRIB, el parámetro pioValue debe apuntar a un db2Uint32 que contenga la prioridad.

Notas de uso

Se devuelve el código SQL1153N si no existe ningún programa de utilidad que tenga el iId especificado. Esto puede indicar que la función se invocó con argumentos no válidos o que el programa de utilidad ha terminado.

Se devuelve el código SQL1154N si el programa de utilidad no es compatible con la función de regulación.

sqlabndx - Programa de aplicación de vinculación para crear un paquete

Invoca el programa de utilidad de vinculación, que prepara las sentencias de SQL almacenadas en el archivo de vinculación generado por el precompilador y crea un paquete que se almacena en la base de datos.

Ámbito

Esta API se puede invocar desde cualquier servidor de particiones de base de datos en db2nodes.cfg. Actualiza los catálogos de base de datos de la partición de

catálogo. Sus efectos son visibles en todos los servidores de particiones de base de datos.

Autorización

Una de las siguientes:

- Autorización sysadm o dbadm
- Privilegio BINDADD si no existe un paquete y una de las opciones siguientes:
- Autorización IMPLICIT_SCHEMA para la base de datos si el nombre de esquema del paquete no existe
- Privilegio CREATEIN en el esquema si existe el nombre de esquema del paquete
- Privilegio ALTERIN en el esquema si existe el paquete
- Privilegio BIND en el paquete si éste existe.

El usuario también necesita todos los privilegios necesarios para compilar las sentencias de SQL estáticas en la aplicación. Los privilegios otorgados a grupos no se utilizan para la comprobación de autorización de las sentencias estáticas. Si el usuario tiene autorización sysadm, pero carece de privilegios explícitos para realizar la vinculación, el gestor de bases de datos otorga automáticamente la autorización dbadm.

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sql.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlabndx (
        _SQLOLDCHAR * pBindFileName,
        _SQLOLDCHAR * pMsgFileName,
        struct sqlopt * pBindOptions,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgbndx (
        unsigned short MsgFileNameLen,
        unsigned short BindFileNameLen,
        struct sqlca * pSqlca,
        struct sqlopt * pBindOptions,
        _SQLOLDCHAR * pMsgFileName,
        _SQLOLDCHAR * pBindFileName);
```

Parámetros de la API sqlabndx

pBindFileName

Entrada. serie que contiene el nombre del archivo de vinculación, o el nombre de un archivo que contiene una lista de nombres de archivo de vinculación. Los nombres de archivo de vinculación deben contener la extensión .bnd. Puede especificarse una vía de acceso para estos archivos.

Anteponga el signo de arroba (@) al nombre de los archivos de lista de vinculación. Por ejemplo, el nombre totalmente calificado de un archivo de lista de vinculación podría ser:

```
/u/user1/bnd/@all.lst
```

El archivo de lista de vinculación debe contener uno o más nombres de archivos de vinculación, y debe tener la extensión .lst.

Anteponga un signo más (+) a todos los nombres de archivos de vinculación, excepto el primero. Los nombres de archivos de vinculación pueden estar en uno o más líneas. Por ejemplo, el archivo de lista de vinculación `all.lst` puede contener:

```
mybind1.bnd+mybind2.bnd+
mybind3.bnd+
mybind4.bnd
```

La lista de archivos de vinculación puede contener especificaciones de la vía de acceso. Si no se especifica ninguna vía de acceso, el gestor de bases de datos obtiene la información sobre la vía de acceso a partir del archivo de lista de vinculación.

pMsgFileName
Entrada. Serie que contiene el destino de los mensajes de error, de aviso e informativos devueltos. Puede ser la vía de acceso y el nombre de un archivo del sistema operativo o un dispositivo estándar. Si ya existe un archivo, se sobregrabará. Si no existe, el archivo se creará.

pBindOptions
Entrada. Estructura que se utiliza para pasar opciones de vinculación a la API. Para obtener más información sobre esta estructura, consulte SQLOPT.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlgbndx

pMsgFileName
Entrada. Serie que contiene el destino de los mensajes de error, de aviso e informativos devueltos. Puede ser la vía de acceso y el nombre de un archivo del sistema operativo o un dispositivo estándar. Si ya existe un archivo, se sobregrabará. Si no existe, el archivo se creará.

BindFileNameLen
Entrada. Longitud, en bytes, del parámetro de datos.

Notas de uso

La vinculación puede realizarse como parte del proceso de precompilación para un archivo fuente de programa de aplicación o posteriormente como un paso independiente. Utilice BIND cuando la vinculación se realice como un proceso independiente.

El nombre utilizado para crear el paquete se almacena en el archivo de vinculación y se basa en el nombre del archivo fuente a partir del cual se ha generado (se eliminan las vías de acceso o las extensiones existentes). Por ejemplo, un archivo fuente precompilado llamado miapl.sqc genera un archivo de vinculación por omisión llamado miapl.bnd y un nombre de paquete por omisión MIAPL. (Sin embargo, el nombre de archivo de vinculación y el nombre de paquete pueden alterarse temporalmente durante la precompilación, utilizando las opciones SQL_BIND_OPT y SQL_PKG_OPT de sqlaprep.

BIND se ejecuta bajo la transacción que el usuario ha iniciado. Después de efectuar la vinculación, BIND emite una operación COMMIT (si la vinculación es satisfactoria) o ROLLBACK (si la vinculación no es satisfactoria) para terminar la transacción actual e iniciar otra.

La vinculación se detiene si se produce un error muy grave o se producen más de 100 errores. Si se produce un error muy grave durante la vinculación, BIND detiene la vinculación, intenta cerrar todos los archivos y descarta el paquete.

Los programas de aplicación de vinculación tienen prerrequisitos y restricciones que quedan fuera del ámbito del presente manual. Por ejemplo, una aplicación no se puede vincular desde un cliente de la Versión 8 a un servidor de la Versión 8 y luego ejecutarse en un servidor de la Versión 7.

Los tipos y valores de las opciones de vinculación se definen en sql.h.

Sintaxis de la API de REXX

Esta API se puede invocar desde REXX mediante la interfaz SQLDB2.

sqlaintp - Obtener mensaje de error

Obtiene el mensaje asociado a una condición de error especificada por el campo sqlcode de la estructura sqlca.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sql.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlaintp (
        char * pBuffer,
        short BufferSize,
        short LineWidth,
        const char * pMsgFileName,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgintp (
        short BufferSize,
        short LineWidth,
        struct sqlca * pSqlca,
        _SQLOLDCHAR * pBuffer);
```

Parámetros de la API sqlaintp

pBuffer
Salida. Puntero a un almacenamiento intermedio de series donde se coloca el texto del mensaje. Si se debe truncar el mensaje para que quepa en el almacenamiento intermedio, el truncamiento tiene en cuenta el carácter de terminación de serie nula.

BufferSize
Entrada. Tamaño, en bytes, de un almacenamiento intermedio de texto que deberá contener el texto del mensaje recuperado.

LineWidth
Entrada. Ancho máxima de línea correspondiente a cada línea del texto de mensaje. Las líneas se dividen a nivel de palabras. Un valor cero indica que el texto del mensaje se devuelve sin divisiones de línea.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Se devuelve un solo mensaje para cada llamada a la API.

Al final de cada mensaje se coloca una secuencia de línea nueva (salto de línea (LF), o retorno de carro/salto de línea (CR/LF)).

Si se especifica un ancho de línea positivo, se insertan secuencias de línea nueva entre palabras para que las líneas no sobrepasen el ancho de línea.

Si una palabra es más larga que un ancho de línea, se colocan en la línea tantos caracteres como quepan, se inserta una línea nueva y los caracteres restantes se colocan en la línea siguiente.

En una aplicación multihebra, sqlaintp debe estar asociado a un contexto válido; de lo contrario, no se puede obtener el texto de mensaje para SQLCODE - 1445.

Códigos de retorno

Código	Mensaje
+i	Número entero positivo que indica el número de bytes del mensaje formateado. Si este valor es mayor que el tamaño de almacenamiento intermedio proporcionado por la aplicación solicitante, se trunca el mensaje.
-1	No hay memoria suficiente disponible para el funcionamiento de los servicios de formateo de mensajes. No se devuelve el mensaje solicitado.
-2	Ausencia de error. La sqlca no contenía un código de error (SQLCODE = 0).
-3	El archivo de mensajes es inaccesible o incorrecto.
-4	El ancho de línea es menor que cero.
-5	sqlca no válida, dirección de almacenamiento intermedio incorrecta o longitud de almacenamiento intermedio incorrecta.

Si el código de retorno es -1 o -3, el almacenamiento intermedio de mensajes contendrá información adicional sobre el problema.

Sintaxis de la API de REXX

```
GET MESSAGE INTO :msg [LINEWIDTH width]
```

Parámetros de la API de REXX

msg Variable de REXX en la que se coloca el texto del mensaje.

width Ancho máximo de las líneas del mensaje de texto. Las líneas se dividen a nivel de palabras. Si no se proporciona el ancho o se establece en 0, se devuelve el texto del mensaje sin divisiones de línea.

sqlaprep - Precompilar programa de aplicación

Procesa un archivo fuente de programa de aplicación que contiene sentencias de SQL incorporadas. Se produce un archivo fuente modificado, que contiene llamadas de lenguaje principal para las sentencias de SQL y, por omisión, se crea un paquete en la base de datos.

Ámbito

Esta API se puede invocar desde cualquier servidor de particiones de base de datos en db2nodes.cfg. Actualiza los catálogos de base de datos de la partición de catálogo. Sus efectos son visibles en todos los servidores de particiones de base de datos.

Autorización

Una de las siguientes:

- Autorización sysadm o dbadm
- Privilegio BINDADD si no existe un paquete y una de las opciones siguientes:
- Autorización IMPLICIT_SCHEMA para la base de datos si el nombre de esquema del paquete no existe
- Privilegio CREATEIN en el esquema si existe el nombre de esquema del paquete
- Privilegio ALTERIN en el esquema si existe el paquete
- Privilegio BIND en el paquete si éste existe.

El usuario también necesita todos los privilegios necesarios para compilar las sentencias de SQL estáticas en la aplicación. Los privilegios otorgados a grupos no se utilizan para la comprobación de autorización de las sentencias estáticas. Si el usuario tiene autorización sysadm, pero carece de privilegios explícitos para realizar la vinculación, el gestor de bases de datos otorga automáticamente la autorización dbadm.

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sql.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlaprep (
        _SQLOLDCHAR * pProgramName,
        _SQLOLDCHAR * pMsgFileName,
        struct sqlopt * pPrepOptions,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgprep (
        unsigned short MsgFileNameLen,
        unsigned short ProgramNameLen,
        struct sqlca * pSqlca,
        struct sqlopt * pPrepOptions,
        _SQLOLDCHAR * pMsgFileName,
        _SQLOLDCHAR * pProgramName);
```

Parámetros de la API sqlaprep

pProgramName
Entrada. Serie que contiene el nombre de la aplicación que se debe precompilar. Utilice las extensiones siguientes:

- .sqb : para aplicaciones COBOL
- .sqc : para aplicaciones C
- .sqC: para aplicaciones C++ de UNIX
- .sqf : para aplicaciones FORTRAN
- .sqx : para aplicaciones C++

Cuando se utiliza la opción TARGET, la extensión del nombre del archivo de entrada no es necesario que sea una de las listadas anteriormente.

La extensión preferida para las aplicaciones C++ que contienen SQL incorporado en los sistemas basados en UNIX es sqC; sin embargo, los sistemas basados en UNIX aceptan la utilización del convenio sqx, que se creó para sistemas que no distinguen entre mayúsculas y minúsculas.

pMsgFileName
Entrada. Serie que contiene el destino de los mensajes de error, de aviso e informativos devueltos. Puede ser la vía de acceso y el nombre de un archivo del sistema operativo o un dispositivo estándar. Si ya existe un archivo, se sobregrabará. Si no existe, el archivo se creará.

pPrepOptions
Entrada. Estructura que se utiliza para pasar opciones de precompilación a la API. Para obtener más información sobre esta estructura, consulte SQLOPT.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlgprep

MsgFileNameLen
Entrada. Longitud, en bytes, del parámetro pMsgFileName.

ProgramNameLen
Entrada. Longitud, en bytes, del parámetro pProgramName.

Notas de uso

Se produce un archivo fuente modificador, que contiene equivalentes de lenguaje de sistema principal a las sentencias de SQL. Por omisión, se crea un paquete en la base de datos con la que se ha establecido una conexión. El nombre del paquete es el mismo que el nombre de archivo de programa (menos la extensión y convertido a mayúsculas), hasta un máximo de 8 caracteres.

Después de establecer conexión con una base de datos, sqlaprep se ejecuta bajo la transacción que se inició. A continuación, PRECOMPILE PROGRAM emite una operación COMMIT o ROLLBACK para finalizar la transacción actual e iniciar otra.

La precompilación se detiene si se produce un error muy grave o se producen más de 100 errores. Si se produce un error muy grave, PRECOMPILE PROGRAM detiene la precompilación, intenta cerrar todos los archivos y descarta el paquete.

Los tipos y valores de las opciones de precompilación se definen en sql.h.

Al utilizar el mandato PRECOMPILE o la API sqlaprep, el nombre del paquete se puede especificar con la opción PACKAGE USING. Al utilizar esta opción, se pueden especificar un máximo de 128 bytes para el nombre del paquete. Si no se utiliza esta opción, el nombre del paquete lo genera el precompilador. El nombre del archivo fuente del programa de aplicación (menos la extensión y convertido a mayúsculas) utiliza un máximo de 8 caracteres. El nombre generado seguirá teniendo un máximo de 8 bytes para ser compatible con versiones anteriores de DB2.

Sintaxis de la API de REXX

```
Esta API se puede invocar desde REXX mediante la interfaz SQLDB2.
```

sqlarbnd - Volver a vincular paquete

Permite al usuario volver a crear un paquete almacenado en la base de datos sin necesidad de un archivo de vinculación.

Autorización

Una de las siguientes:

- Autorización sysadm o dbadm
- Privilegio ALTERIN en el esquema
- Privilegio BIND en el paquete.

El ID de autorización conectado a la columna BOUNDBY de la tabla de catálogos del sistema SYSCAT.PACKAGES, que es el ID del vinculador más reciente del paquete, se utiliza como ID de autorización de vinculador para la revinculación y para el esquema por omisión de las referencias de tabla del paquete. Tenga en cuenta que este calificador por omisión puede ser diferente del ID de autorización del usuario que ejecuta la petición de revinculación. REBIND utilizará las mismas opciones de vinculación que se han especificado al crear el paquete.

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sql.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlarbnd (
        char * pPackageName,
        struct sqlca * pSqlca,
        struct sqlopt * pRebindOptions);

SQL_API_RC SQL_API_FN
  sqlgrbnd (
        unsigned short PackageNameLen,
        char * pPackageName,
        struct sqlca * pSqlca,
        struct sqlopt * pRebindOptions);
```

Parámetros de la API sqlarbnd

pPackageName
Entrada. Serie que contiene el nombre calificado o no calificado que indica el paquete que se debe volver a vincular. El nombre de paquete no calificado está calificado implícitamente por el ID de autorización actual. Este nombre no incluye la versión del paquete. Cuando se especifica un paquete que tiene una versión que no es una serie vacía, el ID de versión se debe especificar mediante la opción de revinculación SQL_VERSION_OPT.

pSqlca
Salida. Puntero a la estructura sqlca.

pRebindOptions
Entrada. Puntero a la estructura SQLOPT, que se utiliza para pasar opciones de revinculación a la API. Para obtener más información sobre esta estructura, consulte SQLOPT.

Parámetros específicos de la API sqlgrbnd

PackageNameLen
Entrada. Longitud, en bytes, del parámetro pPackageName.

Notas de uso

REBIND no confirma automáticamente la transacción a continuación de una revinculación satisfactoria. El usuario debe confirmar la transacción explícitamente. Esto permite realizar un análisis "que ocurriría si", en el que el usuario actualiza determinadas estadísticas y entonces intenta volver a vincular el paquete para ver qué cambios se han producido. También permite múltiples revinculaciones dentro de una unidad de trabajo.

Esta API:

- Proporciona un modo rápido para volver a crear un paquete. Esto permite al usuario aprovechar un cambio en el sistema sin necesidad del archivo de vinculación original. Por ejemplo, si es probable que una sentencia de SQL determinada pueda aprovechar un índice recién creado, se puede utilizar el mandato REBIND para volver a crear el paquete. REBIND también se puede utilizar para volver a crear paquetes después de haber ejecutado db2Runstats, aprovechando así las nuevas estadísticas.
- Proporciona un método para volver a crear paquetes no operativos. Los paquetes no operativos deben volverse a vincular explícitamente invocando el programa de utilidad de vinculación o el programa de utilidad de revinculación. Un paquete se marcará como no operativo (la columna VALID del catálogo de sistema SYSCAT.PACKAGES se establecerá en X) si se descarta una instancia de función de la que depende el paquete. La opción de conservación de revinculación no está soportada para paquetes no operativos.
- Proporciona a los usuarios control sobre la revinculación de paquetes no válidos. El gestor de bases de datos volverá a vincular automáticamente (o implícitamente) los paquetes no válidos cuando éstos se ejecuten. Esto podría producir un retardo considerable en la ejecución de la primera petición SQL del paquete no válido. Puede ser deseable volver a vincular explícitamente los paquetes no válidos, en lugar de dejar que el sistema los vuelva a vincular automáticamente, a fin de eliminar el retardo inicial y de evitar mensajes de error de SQL inesperados que pueden devolverse en el caso de que falle la revinculación implícita. Por ejemplo, después de la migración de la base de

datos, todos los paquetes almacenados en la base de datos quedarán invalidados por el mandato MIGRATE DATABASE. Dado que esto puede incluir un gran número de paquetes, puede ser deseable volver a vincular explícitamente todos los paquetes no válidos a la vez. Esta revinculación explícita puede llevarse a cabo utilizando BIND, REBIND o la herramienta db2rbind.

La elección de utilizar BIND o REBIND para volver a vincular explícitamente un paquete dependerá de las circunstancias. Se recomienda utilizar REBIND siempre que la situación no requiera específicamente el uso de BIND, porque el rendimiento de REBIND es significativamente mejor que el de BIND. Sin embargo, debe utilizar BIND en estos casos:

- Cuando se hayan producido modificaciones en el programa (por ejemplo, cuando se hayan añadido o suprimido sentencias de SQL o cuando el paquete no coincida con el ejecutable para el programa).
- Cuando el usuario desee modificar cualquiera de las opciones de vinculación como parte de la revinculación. REBIND no soporta ninguna opción de vinculación. Por ejemplo, si el usuario desea que se le otorguen privilegios en el paquete como parte del proceso de vinculación, se deberá utilizar BIND, dado que tiene una opción SQL_GRANT_OPT.
- Cuando el paquete no existe actualmente en la base de datos.
- Cuando se desee la detección de todos los errores de vinculación. REBIND sólo devuelve el primer error que detecta y luego finaliza, mientras que el mandato BIND devuelve los 100 primeros errores que se producen durante la vinculación.

DB2 Connect soporta REBIND.

Si se ejecuta REBIND en un paquete que está utilizando otro usuario, la revinculación no se producirá hasta que finalice la unidad de trabajo lógica del otro usuario, ya que durante la revinculación se mantiene un bloqueo exclusivo en el registro del paquete de la tabla de catálogos del sistema SYSCAT.PACKAGES.

Cuando se ejecuta REBIND, el gestor de bases de datos recrea el paquete a partir de las sentencias de SQL almacenadas en la tabla de catálogos del sistema SYSCAT.STATEMENTS. Si existen muchas versiones con el mismo número de paquete y creador, solamente se puede vincular una sola versión cada vez. Si no se especifica la versión mediante la opción de revinculación SQL_VERSION_OPT, VERSION se establece en ″″ por omisión. Aunque exista un solo paquete cuyo nombre y creador coincida con los especificados en la petición de revinculación, no se revinculará el paquete a menos que su versión coincida con la versión especificada explícita o implícitamente.

Si REBIND encuentra un error, el proceso se detiene y se devuelve un mensaje de error.

Las tablas de Explain se llenan durante la revinculación si SQL_EXPLSNAP_OPT o SQL_EXPLAIN_OPT se han establecido en YES o ALL (examine las columnas EXPLAIN_SNAPSHOT y EXPLAIN_MODE del catálogo). Las tablas de explicación utilizadas son las del solicitante de REBIND, no del vinculador original. Los tipos y valores de las opciones de revinculación se definen en sql.h.

Sintaxis de la API de REXX

Esta API se puede invocar desde REXX mediante la interfaz SQLDB2.

sqlbctcq - Cerrar una consulta de contenedor de espacio de tablas

Finaliza una petición de consulta de un contenedor de espacio de tablas y libera los recursos asociados.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint
- dbadm

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlbctcq (
   struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgctcq (
   struct sqlca * pSqlca);
```

Parámetros de la API sqlbctcq

pSqlca
Salida. Puntero a la estructura sqlca.

sqlbctsq - Cerrar una consulta de espacio de tablas

Finaliza una petición de una consulta de espacio de tablas y libera los recursos asociados.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint
- dbadm
- load

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlbctsq (
   struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgctsq (
   struct sqlca * pSqlca);
```

Parámetros de la API sqlbctsq

pSqlca
Salida. Puntero a la estructura sqlca.

---

sqlbftcq - Captar los datos de la consulta para filas de un contenedor de espacio de tablas

Capta un número especificado de filas de datos de la consulta para un contenedor de espacio de tablas, donde cada fila consta de datos de un contenedor.

Ámbito

En un entorno de bases de datos particionadas, solamente se listan los espacios de tablas pertenecientes a la partición de base de datos actual.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint
- dbadm

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlbftcq (
   struct sqlca * pSqlca,
   sqluint32 MaxContainers,
   struct SQLB_TBSCONTQRY_DATA * pContainerData,
   sqluint32 * pNumContainers);

SQL_API_RC SQL_API_FN
  sqlgftcq (
   struct sqlca * pSqlca,
   sqluint32 MaxContainers,
   struct SQLB_TBSCONTQRY_DATA * pContainerData,
   sqluint32 * pNumContainers);
```

Parámetros de la API sqlbftcq

pSqlca
Salida. Puntero a la estructura sqlca.

MaxContainers
Entrada. Número máximo de filas de datos que puede contener el área de salida asignada por el usuario (apuntada por pContainerData).

pContainerData
Salida. Puntero al área de salida, estructura para contener los datos de la consulta. Para obtener más información sobre esta estructura, consulte SQLB-TBSCONTQRY-DATA. El llamador de esta API debe asignar espacio para el número de filas de estas estructuras indicado por MaxContainers, y establecer pContainerData para que apunte a este espacio. La API utilizará este espacio para devolver los datos del contenedor de espacio de tablas.

pNumContainers
Salida. Número de filas devueltas en los datos de salida.

Notas de uso

El usuario debe asignar y liberar la memoria indicada por el parámetro pContainerData. Esta API solamente se puede ejecutar después de invocar sqlbotcq satisfactoriamente. Esta API se puede invocar repetidamente para recuperar la lista generada por sqlbotcq.

sqlbftpq - Captar los datos de la consulta para filas de un espacio de tablas

Capta un número especificado de filas de datos de la consulta para un espacio de tablas, donde cada fila consta de datos de un espacio de tablas.

Ámbito

En un entorno de bases de datos particionadas, solamente se listan los espacios de tablas pertenecientes a la partición de base de datos actual.

Autorización

Una de las siguientes:
- sysadm
- sysctrl
- sysmaint
- dbadm
- load

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlbftpq (
   struct sqlca * pSqlca,
   sqluint32 MaxTablespaces,
   struct SQLB_TBSPQRY_DATA * pTablespaceData,
   sqluint32 * pNumTablespaces);

SQL_API_RC SQL_API_FN
  sqlgftpq (
   struct sqlca * pSqlca,
   sqluint32 MaxTablespaces,
   struct SQLB_TBSPQRY_DATA * pTablespaceData,
   sqluint32 * pNumTablespaces);
```

Parámetros de la API sqlbftpq

pSqlca
Salida. Puntero a la estructura sqlca.

MaxTablespaces
Entrada. Número máximo de filas de datos que puede contener el área de salida asignada por el usuario (apuntada por pTablespaceData).

pTablespaceData
Entrada y salida. Puntero al área de salida, estructura para contener los datos de la consulta. Para obtener más información sobre esta estructura, consulte SQLB-TBSPQRY-DATA. El que llama a esta API deberá:

- Asignar espacio para el número de filas de datos de estas estructuras especificado por MaxTablespaces
- Inicializar las estructuras
- Establecer el parámetro TBSPQVER de la primera estructura en SQLB_TBSPQRY_DATA_ID
- Establecer pTablespaceData para que apunte a este espacio. La API utilizará este espacio para devolver los datos del espacio de tablas.

pNumTablespaces
Salida. Número de filas devueltas en los datos de salida.

Notas de uso

El usuario debe asignar y liberar la memoria indicada por el parámetro pTablespaceData. Esta API solamente se puede ejecutar después de invocar sqlbotsq satisfactoriamente. Esta API se puede invocar repetidamente para recuperar la lista generada por sqlbotsq.

sqlbgtss - Obtener estadísticas de utilización del espacio de tablas

Proporciona información sobre la utilización del espacio para un espacio de tablas.

Ámbito

En un entorno de bases de datos particionadas, solamente se listan los espacios de tablas pertenecientes a la partición de base de datos actual.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint
- dbadm
- load

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlbgtss (
   struct sqlca * pSqlca,
   sqluint32 TablespaceId,
   struct SQLB_TBS_STATS * pTablespaceStats);

SQL_API_RC SQL_API_FN
  sqlggtss (
   struct sqlca * pSqlca,
   sqluint32 TablespaceId,
   struct SQLB_TBS_STATS * pTablespaceStats);
```

Parámetros de la API sqlbgtss

pSqlca
Salida. Puntero a la estructura sqlca.

TablespaceId
Entrada. ID del espacio de tablas individual que se debe consultar.

pTablespaceStats
Salida. Puntero a la estructura SQLB_TBS_STATS asignada por el usuario. La información sobre el espacio de tablas se devuelve en esta estructura.

Notas de uso

Consulte SQLB-TBS-STATS para obtener información sobre los campos devueltos y su significado.

sqlbmtsq - Obtener los datos de la consulta para todos los espacios de tablas

Proporciona una interfaz de una sola llamada para los datos de la consulta de espacio de tablas. Los datos de la consulta para todos los espacios de tablas de la base de datos se devuelven en una matriz.

Ámbito

En un entorno de bases de datos particionadas, solamente se listan los espacios de tablas pertenecientes a la partición de base de datos actual.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint
- dbadm
- load

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlbmtsq (
   struct sqlca * pSqlca,
   sqluint32 * pNumTablespaces,
   struct SQLB_TBSPQRY_DATA *** pppTablespaceData,
   sqluint32 reserved1,
   sqluint32 reserved2);

SQL_API_RC SQL_API_FN
  sqlgmtsq (
   struct sqlca * pSqlca,
   sqluint32 * pNumTablespaces,
   struct SQLB_TBSPQRY_DATA *** pppTablespaceData,
   sqluint32 reserved1,
   sqluint32 reserved2);
```

Parámetros de la API sqlbmtsq

pSqlca
Salida. Puntero a la estructura sqlca.

pNumTablespaces
Salida. Número total de espacios de tablas contenidos en la base de datos conectada.

pppTablespaceData
Salida. El que llama proporciona a la API la dirección de un puntero. La API asigna el espacio para los datos de consulta del espacio de tablas y se devuelve al que realiza la llamada un puntero a ese espacio. Cuando finaliza la llamada, el puntero apunta a una matriz de punteros SQLB_TBSPQRY_DATA que apunta al conjunto completo de datos de consulta del espacio de tablas.

reserved1
Entrada. Siempre es SQLB_RESERVED1.

reserved2
Entrada. Siempre es SQLB_RESERVED2.

Notas de uso

Esta API utiliza los servicios de nivel inferior, a saber:

- sqlbotsq
- sqlbftpq
- sqlbctsq

para obtener a la vez todos los datos de la consulta de espacio de tablas.

Si existe suficiente memoria disponible, esta función devuelve el número de espacios de tablas y un puntero a la ubicación en la memoria de los datos de consulta del espacio de tablas. El usuario debe liberar esta memoria mediante una llamada a sqlefmem.

Si no existe suficiente memoria disponible, esta función devuelve simplemente el número de espacios de tablas y no se asigna memoria. Si ocurre esto, utilice sqlbotsq, sqlbftpq y sqlbctsq para recuperar cada vez algo menos que la lista completa.

sqlbotcq - Abrir una consulta de contenedor de espacio de tablas

Prepara una operación de consulta de contenedor de espacio de tablas y devuelve el número de contenedores existentes actualmente en el espacio de tablas.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint
- dbadm

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlbotcq (
   struct sqlca * pSqlca,
   sqluint32 TablespaceId,
   sqluint32 * pNumContainers);

SQL_API_RC SQL_API_FN
  sqlgotcq (
   struct sqlca * pSqlca,
   sqluint32 TablespaceId,
   sqluint32 * pNumContainers);
```

Parámetros de la API sqlbotcq

pSqlca
Salida. Puntero a la estructura sqlca.

TablespaceId
Entrada. ID del espacio de tablas para el que se desean datos del contenedor. Si se especifica el identificador especial SQLB_ALL_TABLESPACES (en sqlutil.h), se genera una lista completa de contenedores para toda la base de datos.

pNumContainers
Salida. Número de contenedores existentes en el espacio de tablas especificado.

Notas de uso

Esta API está normalmente seguida por una o más llamadas a sqlbftcq y luego por una llamada a sqlbctcq.

Una aplicación puede utilizar las API siguientes para recuperar información sobre los contenedores utilizados por espacios de tabla:

- sqlbtcq

  Recupera una lista completa de información de contenedor. La API asigna el espacio necesario para contener la información sobre todos los contenedores, y devuelve un puntero a esa información. Utilice esta API para examinar la lista de contenedores cuando busque una información determinada. La utilización de esta API equivale a invocar las tres API siguientes (sqlbotcq, sqlbftcq, sqlbctcq), salvo que esta API asigna automáticamente la memoria para la información de salida. Las llamadas a esta API deben ir seguidas por una llamada a sqlefmem para liberar la memoria.
- sqlbotcq
- sqlbftcq
- sqlbctcq

  Estas tres API actúan como un cursor de SQL, en cuanto que utilizan el modelo OPEN/FETCH/CLOSE. El llamador debe proporcionar el área de salida para la recuperación de información. A diferencia de un cursor de SQL, solamente puede estar activa una única consulta de contenedor de espacio de tablas en cada momento. Utilice estas API para explorar la lista de contenedores de espacios de tablas cuando busque una información determinada. Estas API permiten al usuario controlar las necesidades de memoria de una aplicación (comparado con sqlbtcq).

Cuando se invoca sqlbotcq, se crea una instantánea de la información sobre el contenedor actual en el agente que presta servicio a la aplicación. Si la aplicación emite una segunda llamada de consulta de contenedor de espacio de tablas (sqlbtcq o sqlbotcq), esta instantánea se sustituye con información renovada.

No se realiza ningún bloqueo, por lo que la información del almacenamiento intermedio puede que no refleje los cambios hechos por otra aplicación después de la creación de la instantánea. La información no forma parte de una transacción.

Existe un almacenamiento intermedio de instantáneas para las consultas de espacio de tablas y otro para las consultas de contenedor de espacio de tablas. Estos almacenamientos intermedios son independientes entre sí.

sqlbotsq - Abrir una consulta de espacio de tablas

Prepara una operación de consulta de espacio de tablas y devuelve el número de espacios de tablas existentes actualmente en la base de datos.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint
- dbadm
- load

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlbotsq (
   struct sqlca * pSqlca,
   sqluint32 TablespaceQueryOptions,
   sqluint32 * pNumTablespaces);

SQL_API_RC SQL_API_FN
  sqlgotsq (
   struct sqlca * pSqlca,
   sqluint32 TablespaceQueryOptions,
   sqluint32 * pNumTablespaces);
```

Parámetros de la API sqlbotsq

pSqlca
Salida. Puntero a la estructura sqlca.

TablespaceQueryOptions
Entrada. Indica qué espacios de tablas hay que procesar. Los valores válidos (definidos en sqlutil) son:

SQLB_OPEN_TBS_ALL
Procesa todos los espacios de tablas de la base de datos.

SQLB_OPEN_TBS_RESTORE
Procesa solamente los espacios de tablas que el agente del usuario está restaurando.

pNumTablespaces
Salida. Número de espacios de tablas contenidos en la base de datos conectada.

Notas de uso

Esta API va normalmente seguida por una o más llamadas a sqlbftpq y luego por una llamada a sqlbctsq.

Una aplicación puede utilizar las API siguientes para recuperar información sobre los espacios de tablas definidos actualmente:

- sqlbstpq

  Recupera información sobre un espacio de tablas determinado. Se devuelve una sola entrada de espacio de tablas (que se coloca en un espacio proporcionado por el llamador). Utilice esta API cuando se conozca el identificador del espacio de tablas y se desee información solamente sobre esa tabla.
- sqlbmtsq

  Recupera información sobre todos los espacios de tablas. La API asigna el espacio necesario para contener la información sobre todos los espacios de tablas, y devuelve un puntero a esa información. Utilice esta API para examinar la lista de espacios de tablas cuando busque una información determinada. La utilización de esta API equivale a invocar las tres API siguientes, salvo que esta API asigna automáticamente la memoria para la información de salida. Las llamadas a esta API deben ir seguidas por una llamada a sqlefmem para liberar la memoria.
- sqlbotsq
- sqlbftpq
- sqlbctsq

  Estas tres API actúan como un cursor de SQL, en cuanto que utilizan el modelo OPEN/FETCH/CLOSE. El llamador debe proporcionar el área de salida para la recuperación de información. A diferencia de un cursor de SQL, solamente puede estar activa una única consulta de espacio de tablas en cada momento. Utilice estas API para explorar la lista de espacios de tablas cuando busque una información determinada. Estas API permiten al usuario controlar las necesidades de memoria de una aplicación (comparado con sqlbmtsq).

Cuando se invoca sqlbotsq, se coloca en almacenamiento intermedio una instantánea de la información del espacio de tablas actual en el agente que presta servicio a la aplicación. Si la aplicación emite una segunda llamada de consulta del espacio de tablas (sqlbmtsq o sqlbotsq), esta instantánea se sustituye con información renovada.

No se realiza ningún bloqueo, por lo que la información del almacenamiento intermedio puede que no refleje cambios más recientes hechos por otra aplicación. La información no forma parte de una transacción.

Existe un almacenamiento intermedio de instantáneas para las consultas de espacio de tablas y otro para las consultas de contenedor de espacio de tablas. Estos almacenamientos intermedios son independientes entre sí.

sqlbstpq - Obtener información sobre un espacio de tablas individual

Recupera información sobre un espacio de tablas individual definido actualmente.

Ámbito

En un entorno de bases de datos particionadas, solamente se listan los espacios de tablas pertenecientes a la partición de base de datos actual.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint
- dbadm
- load

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlbstpq (
   struct sqlca * pSqlca,
   sqluint32 TablespaceId,
   struct SQLB_TBSPQRY_DATA * pTablespaceData,
   sqluint32 reserved);

SQL_API_RC SQL_API_FN
  sqlgstpq (
   struct sqlca * pSqlca,
   sqluint32 TablespaceId,
   struct SQLB_TBSPQRY_DATA * pTablespaceData,
   sqluint32 reserved);
```

Parámetros de la API sqlbstpq

pSqlca
Salida. Puntero a la estructura sqlca.

TablespaceId
Entrada. Identificador del espacio de tablas que se debe consultar.

pTablespaceData
Entrada y salida. Puntero a una estructura SQLB_TBSPQRY_DATA suministrada por el usuario donde se colocará la información sobre el espacio de tablas cuando vuelva. El llamador de esta API debe inicializar la estructura y establecer TBSPQVER en SQLB_TBSPQRY_DATA_ID (en sqlutil).

reserved
Entrada. Siempre es SQLB_RESERVED1.

Notas de uso

Esta API recupera información sobre un espacio de tablas individual si se conoce el identificador del espacio de tablas que se debe consultar. Esta API proporciona una alternativa a la utilización conjunta de las API OPEN TABLESPACE QUERY, FETCH y CLOSE, las cuales requieren más recursos y se deben utilizar para buscar el espacio de tablas deseado cuando no se conoce de antemano el identificador del espacio de tablas. Los ID de espacio de tablas se pueden encontrar en los catálogos del sistema. No se toman instantáneas de agente; puesto que solamente existe una

única entrada para devolver, se devuelve directamente.

sqlbstsc - Definir contenedores de espacios de tablas

Esta API proporciona un mecanismo de restauración redirigida, en la que el usuario restaura una base de datos y se desea o es necesario un conjunto diferente de contenedores de almacenamiento del sistema operativo. Utilice esta API cuando el estado del espacio de tablas sea el de definición de almacenamiento pendiente o definición de almacenamiento permitido. Estos estados son posibles durante una operación de restauración, inmediatamente antes de la restauración de las páginas de base de datos.

Autorización

Una de las siguientes:

- sysadm
- sysctrl

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlbstsc (
   struct sqlca * pSqlca,
   sqluint32 SetContainerOptions,
   sqluint32 TablespaceId,
   sqluint32 NumContainers,
   struct SQLB_TBSCONTQRY_DATA * pContainerData);

SQL_API_RC SQL_API_FN
  sqlgstsc (
   struct sqlca * pSqlca,
   sqluint32 SetContainerOptions,
   sqluint32 TablespaceId,
   sqluint32 NumContainers,
   struct SQLB_TBSCONTQRY_DATA * pContainerData);
```

Parámetros de la API sqlbstsc

pSqlca
Salida. Puntero a la estructura sqlca.

SetContainerOptions
Entrada. Utilice este campo para especificar más opciones. Los valores válidos (definidos en sqlutil) son:

SQLB_SET_CONT_INIT_STATE
Rehace operaciones de alteración de espacio de tablas al realizar un avance.

SQLB_SET_CONT_FINAL_STATE
Pasa por alto las operaciones de alteración de espacio de tablas contenidas en el archivo de anotaciones al realizar un avance.

TablespaceId
Entrada. Identificador del espacio de tablas que se debe modificar.

NumContainers
Entrada. Número de filas contenidas en la estructura apuntada por pContainerData.

pContainerData
Entrada. Especificaciones del contenedor. Aunque se utiliza la estructura SQLB_TBSCONTQRY_DATA, solamente se utilizan los campos contType, totalPages, nombre y nameLen (para lenguajes que no sean C); los demás campos no se tienen en cuenta.

Notas de uso

Esta API se utiliza en combinación con db2Restore.

Una copia de seguridad de una base de datos, o uno o más espacios de tabla, conserva un registro de todos los contenedores de espacios de tablas que están siendo utilizados por los espacios de tablas de los que se está realizando una copia de seguridad. Durante una restauración, se comprueban todos los contenedores listados en la copia de seguridad para ver si existen y son accesibles actualmente. Si uno o más de los contenedores no es accesible por cualquier razón, la restauración fallará. Para permitir una restauración en un caso de este tipo, se soporta la redirección de los contenedores de espacios de tablas durante la restauración. Este soporte incluye la adición, el cambio o la eliminación de contenedores de espacios de tabla. Es esta API la que permite al usuario añadir, cambiar o eliminar dichos contenedores.

La utilización de esta API incluye normalmente esta secuencia de acciones:

1. Invocar db2Restore con CallerAction establecido en DB2RESTORE_RESTORE_STORDEF. El programa de utilidad de restauración devuelve un código de SQL que indica que algunos de los contenedores son inaccesibles.
2. Invocar sqlbstsc para crear las definiciones de contenedores de espacios de tablas con el parámetro SetContainerOptions establecido en SQLB_SET_CONT_FINAL_STATE.
3. Invocar db2Restore por segunda vez con CallerAction establecido en DB2RESTORE_CONTINUE.

La secuencia de acciones anterior permite que la restauración utilice las nuevas definiciones de contenedores de espacios de tablas y pasa por alto las operaciones de adición de contenedor de espacio de tablas contenidas en los archivos de anotaciones cuando se realice un avance (db2Rollforward) una vez finalizada la restauración.

Al definir la lista de contenedores, el usuario de esta API debe tener en cuenta que debe existir suficiente espacio de disco para que la operación de restauración o avance sustituya todos los datos originales en los nuevos contenedores. Si no existe espacio suficiente, esos espacios de tablas quedarán en estado de recuperación pendiente hasta que se habilite espacio de disco suficiente. Es conveniente que el administrador de bases de datos mantenga regularmente registros de la utilización del disco. Luego, cuando sea necesaria una operación de restauración o recuperación hacia adelante, se conocerá el espacio de disco necesario.

sqlbtcq - Obtener los datos de la consulta para todos los contenedores de espacios de tablas

Proporciona una interfaz de una sola llamada para los datos de la consulta de contenedor de espacio de tablas. Los datos de la consulta para todos los contenedores de un espacio de tablas o para todos los contenedores de todos los espacios de tablas se devuelven en una matriz.

Ámbito

En un entorno de bases de datos particionadas, solamente se listan los espacios de tablas pertenecientes a la partición de base de datos actual.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint
- dbadm

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlbtcq (
   struct sqlca * pSqlca,
   sqluint32 TablespaceId,
   sqluint32 * pNumContainers,
   struct SQLB_TBSCONTQRY_DATA ** ppContainerData);

SQL_API_RC SQL_API_FN
  sqlgtcq (
   struct sqlca * pSqlca,
   sqluint32 TablespaceId,
   sqluint32 * pNumContainers,
   struct SQLB_TBSCONTQRY_DATA ** ppContainerData);
```

Parámetros de la API sqlbtcq

pSqlca
Salida. Puntero a la estructura sqlca.

TablespaceId
Entrada. ID del espacio de tablas para el que se desea recuperar datos de contenedor, o un ID especial, SQLB_ALL_TABLESPACES (definido en sqlutil), que genera una lista de todos los contenedores de la base de datos completa.

pNumContainers
Salida. Número de contenedores existentes en el espacio de tablas.

ppContainerData
Salida. El que llama proporciona a la API la dirección de un puntero a una estructura SQLB_TBSCONTQRY_DATA. La API asigna el espacio para los datos de consulta del contenedor del espacio de tablas y se devuelve al que realiza la llamada un puntero a ese espacio. Cuando finaliza la llamada, el puntero a la estructura SQLB_TBSCONTQRY_DATA apunta al conjunto completo de datos de consulta del contenedor del espacio de tablas.

Notas de uso

Esta API utiliza los servicios de nivel inferior, a saber:

- sqlbotcq
- sqlbftcq
- sqlbctcq

para obtener a la vez todos los datos de la consulta de contenedor de espacio de tabla.

Si existe suficiente memoria disponible, esta función devuelve el número de contenedores y un puntero a la ubicación en la memoria de los datos de la consulta de contenedor de espacio de tablas. El usuario debe liberar esta memoria mediante una llamada a sqlefmem. Si no existe suficiente memoria disponible, esta función devuelve simplemente el número de contenedores y no se asigna memoria. Si ocurre esto, utilice sqlbotcq, sqlbftcq y sqlbctcq para recuperar cada vez algo menos que la lista completa.

sqlcspqy - Listar transacciones dudosas DRDA

Proporciona una lista de transacciones que son dudosas entre las conexiones asociadas del gestor de punto de sincronismo. Esta API está en desuso. Consulte 'API db2SpmListIndTrans - Listar transacciones dudosas SPM'.

Autorización

Ninguna

Conexión necesaria

Instancia

Archivo de inclusión de la API

```
sqlxa.h
```

Sintaxis de la API y de las estructuras de datos

```
extern int SQL_API_FN sqlcspqy(SQLCSPQY_INDOUBT     **indoubt_data,
                               sqlint32 *indoubt_count,
                               struct sqlca * sqlca);
```

Parámetros de la API sqlcspqy

indoubt_data
Salida. Puntero a la matriz devuelta.

indoubt_count
Salida. Número de elementos de la matriz devuelta.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Las transacciones dudosas DRDA se producen cuando se pierde la comunicación entre los coordinadores y los participantes en unidades de trabajo distribuidas.

Una unidad de trabajo distribuida permite a un usuario o una aplicación leer y actualizar datos en múltiples ubicaciones dentro de una sola unidad de trabajo. Dicho trabajo requiere una confirmación de dos fases.

La primera fase solicita a todos los participantes que se preparen para una confirmación. La segunda fase confirma o retrotrae las transacciones. Si un coordinador o un participante no está disponible después de la primera fase, las transacciones distribuidas son dudosas.

Antes de emitir el mandato LIST DRDA INDOUBT TRANSACTIONS, se debe conectar el proceso de aplicación a la instancia de SPM (Gestor de puntos de sincronismo). Utilice el parámetro de configuración del gestor de bases de datos spm_name como el alias de base de datos en la sentencia CONNECT.

sqle_activate_db - Activar base de datos

Activa la base de datos especificada y arranca todos los servicios de base de datos necesarios, de modo que la base de datos esté disponible para que se conecte y la utilice cualquier aplicación.

Ámbito

Esta API activa la base de datos especificada en todos los servidores de particiones de base de datos. Si se produce un error durante la activación de la base de datos en uno o más de los servidores de particiones, se emite un aviso. La base de datos permanece activada en todos los servidores de particiones en los que la API se ejecutado satisfactoriamente.

Nota: Si el error se produce en la partición del coordinador o partición del catálogo, la API devuelve un sqlcode negativo y la base de datos no se activará en ningún servidor de particiones de base de datos.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint

Conexión necesaria

Ninguna. Las aplicaciones que llaman a ACTIVATE DATABASE no pueden tener ninguna conexión de base de datos existente.

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqle_activate_db (
        char * pDbAlias,
        char * pUserName,
        char * pPassword,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlg_activate_db (
        unsigned short DbAliasLen,
        unsigned short UserNameLen,
        unsigned short PasswordLen,
        char * pDbAlias,
        char * pUserName,
        char * pPassword,
        void * pReserved,
        struct sqlca * pSqlca);
```

Parámetros de la API sqle_activate_db

pDbAlias
Entrada. Puntero al nombre de alias de base de datos.

pUserName
Entrada. Puntero al ID de usuario que empieza la base de datos. Puede ser NULL.

pPassword
Entrada. Puntero a la contraseña para el nombre de usuario. Puede ser NULL, pero debe especificarse si se especifica un nombre de usuario.

pReserved
Reservado para una utilización futura.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlg_activate_db

DbAliasLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud del nombre de alias de base de datos, expresada en bytes.

UserNameLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud del nombre del usuario, expresada en bytes. El valor se establece en cero si no se proporciona ningún nombre de usuario.

PasswordLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud de la contraseña, expresada en bytes. El valor se establece en cero si no se proporciona ninguna contraseña.

Notas de uso

Si una base de datos no se ha iniciado y se encuentra un mandato de DB2 CONNECT TO (o una conexión implícita) en una aplicación, la aplicación debe esperar a que el gestor de bases de datos inicie la base de datos necesaria. En estos casos, la primera aplicación dedica tiempo a inicializar la base de datos antes de

que pueda realizar cualquier trabajo. Pero una vez que la primera aplicación ha iniciado una base de datos, las demás aplicaciones pueden simplemente conectarse y utilizarla.

Los administradores de bases de datos pueden utilizar ACTIVATE DATABASE para arrancar bases de datos seleccionadas. Esto elimina el tiempo que la aplicación emplea en la inicialización de la base de datos.

Las bases de datos inicializadas por ACTIVATE DATABASE solamente se pueden concluir mediante sqle_deactivate_db o db2InstanceStop. Para obtener una lista de las bases de datos activadas, invoque db2GetSnapshot.

Si se ha iniciado una base de datos mediante DB2 CONNECT TO (o una conexión implícita) y posteriormente se emite ACTIVATE DATABASE para esa misma base de datos, se debe utilizar DEACTIVATE DATABASE para concluir esa base de datos.

ACTIVATE DATABASE se comporta de un modo similar a DB2 CONNECT TO (o una conexión implícita) cuando se utiliza con una base de datos que requiere un reinicio (por ejemplo, una base de datos en un estado incoherente). La base de datos se reiniciará antes de que ACTIVATE DATABASE pueda inicializarla.

Sintaxis de la API de REXX

Esta API se puede invocar desde REXX mediante la interfaz SQLDB2.

sqle_deactivate_db - Desactivar base de datos

Detiene la base de datos especificada.

Ámbito

En un entorno de bases de datos particionadas, esta API desactiva la base de datos especificada en todos los servidores de particiones de base de datos. Si uno o varios de estos servidores de particiones de base de datos encuentra un error, se devuelve un aviso. La base de datos se desactivará satisfactoriamente en algunos servidores de particiones de base de datos, pero permanecerá activada en los servidores de particiones de base de datos donde se produjo el error.

Nota: Si el error se produce en la partición del coordinador o partición del catálogo, la API devuelve un sqlcode negativo y la base de datos no se reactivará en ningún servidor de particiones de base de datos en los que se desactivó la base de datos.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint

Conexión necesaria

Ninguna. Las aplicaciones que llaman a DEACTIVATE DATABASE no pueden tener ninguna conexión de base de datos existente.

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqle_deactivate_db (
        char * pDbAlias,
        char * pUserName,
        char * pPassword,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlg_deactivate_db (
        unsigned short DbAliasLen,
        unsigned short UserNameLen,
        unsigned short PasswordLen,
        char * pDbAlias,
        char * pUserName,
        char * pPassword,
        void * pReserved,
        struct sqlca * pSqlca);
```

Parámetros de la API sqle_deactivate_db

pDbAlias
Entrada. Puntero al nombre de alias de base de datos.

pUserName
Entrada. Puntero al ID de usuario que detiene la base de datos. Puede ser NULL.

pPassword
Entrada. Puntero a la contraseña para el nombre de usuario. Puede ser NULL, pero debe especificarse si se especifica un nombre de usuario.

pReserved
Reservado para una utilización futura.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlg_deactivate_db

DbAliasLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud del nombre de alias de base de datos, expresada en bytes.

UserNameLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud del nombre del usuario, expresada en bytes. El valor se establece en cero si no se proporciona ningún nombre de usuario.

PasswordLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud de la contraseña, expresada en bytes. El valor se establece en cero si no se proporciona ninguna contraseña.

Notas de uso

Las bases de datos inicializadas por ACTIVATE DATABASE sólo se pueden concluir mediante DEACTIVATE DATABASE. db2InstanceStop detiene

automáticamente todas las bases de datos activadas antes de detener el gestor de bases de datos. Si ACTIVATE DATABASE ha inicializado una base de datos, la última sentencia de DB2 CONNECT RESET (contador igual a 0) no concluirá la base de datos; se debe utilizar DEACTIVATE DATABASE.

Sintaxis de la API de REXX

Esta API se puede invocar desde REXX mediante la interfaz SQLDB2.

sqleaddn - Añadir un servidor de particiones de base de datos al entorno de bases de datos particionadas

Añade un nuevo servidor de particiones de base de datos al entorno de bases de datos particionadas. Esta API crea particiones de base de datos para todas las bases de datos definidas actualmente en la instancia en el nuevo servidor de particiones de base de datos. El usuario puede especificar el servidor de particiones de base de datos fuente para cualquier espacio de tablas temporales del sistema que debe crearse con las bases de datos, o especificar que no debe crearse ningún espacio de tablas temporales del sistema. La API se deje ejecutar desde el servidor de particiones de base de datos que se está añadiendo, y solamente se puede ejecutar en un servidor de particiones de base de datos.

Ámbito

Esta API sólo afecta al servidor de particiones de base de datos en el que se ejecuta la API.

Autorización

Una de las siguientes:

- sysadm
- sysctrl

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqleaddn (
        void * pAddNodeOptions,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgaddn (
        unsigned short addnOptionsLen,
        struct sqlca * pSqlca,
        void * pAddNodeOptions);
```

Parámetros de la API sqleaddn

pAddNodeOptions
Entrada. Puntero a la estructura opcional sqle_addn_options. Esta

estructura se utiliza para especificar el servidor de particiones de base de datos fuente, si existe, de las definiciones de espacio de tablas temporales del sistema para todas las particiones de base de datos creadas durante la adición del nodo. Si no se especifica (es decir, se especifica un puntero nulo), las definiciones de espacio de tablas temporal del sistema serán las mismas que para la partición del catálogo.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlgaddn

addnOptionsLen
Entrada. Número entero sin signo de 2 bytes que representa la longitud de la estructura opcional sqle_addn_options en bytes.

Notas de uso

Antes de añadir un nuevo servidor de particiones de base de datos, asegúrese de que haya espacio de almacenamiento suficiente para los contenedores que se deben crear para todas las bases de datos existentes del sistema.

La operación de añadir nodo crea una partición de base de datos vacía en el nuevo servidor de particiones de base de datos para cada base de datos existente en la instancia. Los parámetros de configuración para las nuevas particiones de base de datos se establecen en el valor por omisión.

Si una operación de adición de nodo falla mientras se crea localmente una partición de base de datos, entra en fase de borrado, en la que descarta localmente todas las bases de datos que se han creado. Esto significa que las particiones de base de datos se eliminan solamente del servidor de particiones de base de datos que se está añadiendo (es decir, el servidor de particiones de base de datos local). Las particiones de base de datos existentes permanecen en todos los demás servidores de particiones de base de datos sin quedar afectadas. Si esta acción falla, no se realiza ningún borrado adicional y se devuelve un error.

Las particiones de base de datos del nuevo servidor de particiones de base de datos no se pueden utilizar para contener datos de usuario hasta que se haya emitido la sentencia ALTER DATABASE PARTITION GROUP para añadir el servidor de particiones de base de datos a un grupo de particiones de base de datos.

Esta API falla si está en curso una operación de creación o descarte de una base de datos. La API se puede invocar de nuevo cuando haya finalizado la operación.

Esta API falla si en cualquier momento, en una base de datos del sistema, se ha creado, satisfactoriamente o no, una tabla de usuario con una columna XML, o se ha registrado, satisfactoriamente o no, un objeto XSR.

Para determinar si la base de datos está habilitada para el almacenamiento automático, la API sqleaddn debe comunicarse con la partición de catálogo de cada una de las bases de datos de la instancia. Si el almacenamiento automático está habilitado, las definiciones de vía de acceso de almacenamiento se recuperan como parte de esa comunicación. Del mismo modo, si se deben crear espacios de tablas temporales del sistema con las particiones de base de datos, puede que la API sqleaddn tenga que comunicarse con otro servidor de particiones de base de datos del entorno de bases de datos particionadas a fin de recuperar las definiciones de

espacios de tablas. El parámetro de configuración start_stop_time del gestor de bases de datos se utiliza para especificar el tiempo, en minutos, antes del cual el otro servidor de particiones de base de datos debe responder con las definiciones de almacenamiento automático y espacios de tablas. Si se excede este tiempo, la API fallará. Aumente el valor de start_stop_time y llame de nuevo a la API.

Sintaxis de la API de REXX

Esta API se puede invocar desde REXX mediante la interfaz SQLDB2.

sqleatcp - Conectar a instancia y cambiar contraseña

Permite a una aplicación especificar el nodo en el que deben ejecutarse las funciones a nivel de instancia (por ejemplo, CREATE DATABASE y FORCE APPLICATION. Este nodo puede ser la instancia actual (tal como está definida por el valor de la variable de entorno DB2INSTANCE), otra instancia situada en la misma estación de trabajo o una instancia situada en una estación de trabajo remota. Esta API establece una conexión de instancia lógica con el nodo especificado e inicia una conexión de comunicaciones física con el nodo si no existe ya una.

Nota: Esta API amplía la función de la API sqleatin al permitir el cambio opcional de la contraseña de usuario para la instancia con la que se establece conexión. El sistema de bases de datos DB2 proporciona soporte para el cambio de contraseñas en sistemas operativos AIX, Linux y Windows.

Autorización

Ninguna

Conexión necesaria

Esta API establece una conexión de instancia.

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqleatcp (
        char * pNodeName,
        char * pUserName,
        char * pPassword,
        char * pNewPassword,
        struct sqlca * pSqlca);
```

Parámetros de la API sqleatcp

pNodeName
Entrada. Serie que contiene el alias de la instancia a la que desea conectarse el usuario. Esta instancia puede tener una entrada que coincide en el directorio de nodo local. La única excepción a esto es la instancia local (especificada por la variable de entorno DB2INSTANCE), que puede especificarse como objeto de una conexión, pero que no se puede utilizar como nombre de nodo en el directorio de nodos. Puede ser NULL.

pUserName
Entrada. Serie que contiene el nombre de usuario que se utiliza para autentificar la conexión. Puede ser NULL.

pPassword
Entrada. Serie que contiene la contraseña del nombre de usuario especificado. Puede ser NULL.

pNewPassword
Entrada. Serie que contiene la contraseña nueva del nombre de usuario especificado. Su valor se establece en NULL si no es necesario un cambio de contraseña.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Un nombre de nodo contenido en el directorio de nodos se puede considerar como un alias para una instancia.

Si una petición ATTACH se ejecuta satisfactoriamente, el campo sqlerrmc de la sqlca contendrá 9 símbolos separados por el hexadecimal FF (similar a los símbolos devueltos cuando una petición CONNECT se ejecuta satisfactoriamente):

1. Código de país/región del servidor de aplicaciones
2. Página de códigos del servidor de aplicaciones
3. ID de autorización
4. Nombre del nodo (tal como está especificado en la API)
5. Identidad y tipo de plataforma del servidor
6. ID del agente que se ha iniciado en el servidor
7. Índice de agente
8. Número de nodo del servidor
9. Número de particiones de base de datos si se trata de un servidor de bases de datos particionadas.

Si el nombre de nodo es una serie de longitud cero (serie nula), se devuelve información sobre el estado actual de la conexión. Si no existe ninguna conexión, se devuelve el código de SQL 1427. En otro caso, se devuelve información sobre la conexión en el campo sqlerrmc de la sqlca (tal como se describió anteriormente).

Si no se ha creado una conexión, las API a nivel de instancia se ejecutan para la instancia actual, especificada por la variable de entorno DB2INSTANCE.

Ciertas funciones (por ejemplo, db2start, db2stop y todos los servicios de directorio) no se ejecutan nunca de forma remota. Es decir, afectan solamente al entorno de la instancia local, tal como se define por el valor de la variable de entorno DB2INSTANCE.

Si existe una conexión y la API se ejecuta con un nombre de nodo, se descarta la conexión actual y se intenta establecer una conexión con el nodo nuevo.

El lugar donde se autentican el nombre de usuario y la contraseña, y el lugar sonde se cambia la contraseña dependen del tipo de autentificación de la instancia seleccionada.

El nodo con el que se debe establecer una conexión también se puede especificar mediante una llamada a la API sqlesetc.

Sintaxis de la API de REXX

Desde no se puede llamar directamente a esta API. No obstante, el programador de REXX puede utilizar esta función llamando al procesador de línea de mandatos de DB2 para ejecutar el mandato ATTACH.

sqleatin - Conectar a instancia

Permite a una aplicación especificar el nodo en el que deben ejecutarse las funciones a nivel de instancia (por ejemplo CREATE DATABASE y FORCE APPLICATION). Este nodo puede ser la instancia actual (tal como está definida por el valor de la variable de entorno DB2INSTANCE), otra instancia situada en la misma estación de trabajo o una instancia situada en una estación de trabajo remota. Esta API establece una conexión de instancia lógica con el nodo especificado e inicia una conexión de comunicaciones física con el nodo si no existe ya una.

Nota: Si es necesario cambiar la contraseña, utilice la API sqleatcp en lugar de la API sqleatin.

Autorización

Ninguna

Conexión necesaria

Esta API establece una conexión de instancia.

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqleatin (
        char * pNodeName,
        char * pUserName,
        char * pPassword,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgatin (
        unsigned short PasswordLen,
        unsigned short UserNameLen,
        unsigned short NodeNameLen,
        struct sqlca * pSqlca,
        char * pPassword,
        char * pUserName,
        char * pNodeName);
```

Parámetros de la API sqleatin

pNodeName
Entrada. Serie que contiene el alias de la instancia a la que desea conectarse el usuario. Esta instancia puede tener una entrada que coincide en el directorio de nodo local. La única excepción a esto es la instancia

local (especificada por la variable de entorno DB2INSTANCE), que puede especificarse como objeto de una conexión, pero que no se puede utilizar como nombre de nodo en el directorio de nodos. Puede ser NULL.

pUserName
Entrada. Serie que contiene el nombre de usuario que se utiliza para autentificar la conexión. Puede ser NULL.

pPassword
Entrada. Serie que contiene la contraseña del nombre de usuario especificado. Puede ser NULL.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlgatin

PasswordLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud de la contraseña, expresada en bytes. El valor se establece en cero si no se proporciona ninguna contraseña.

UserNameLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud del nombre del usuario, expresada en bytes. El valor se establece en cero si no se proporciona ningún nombre de usuario.

NodeNameLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud del nombre de nodo, expresada en bytes. El valor se establece en cero si no se proporciona ningún nombre de nodo.

Notas de uso

Nota: Un nombre de nodo contenido en el directorio de nodos se puede considerar como un alias para una instancia.

Si una petición ATTACH se ejecuta satisfactoriamente, el campo sqlerrmc de la sqlca contendrá 9 símbolos separados por el hexadecimal FF (similar a los símbolos devueltos cuando una petición CONNECT se ejecuta satisfactoriamente):

1. Código de país/región del servidor de aplicaciones
2. Página de códigos del servidor de aplicaciones
3. ID de autorización
4. Nombre del nodo (tal como está especificado en la API)
5. Identidad y tipo de plataforma del servidor
6. ID del agente que se ha iniciado en el servidor
7. Índice de agente
8. Número de nodo del servidor
9. Número de particiones de base de datos si se trata de un servidor de bases de datos particionadas.

Si el nombre de nodo es una serie de longitud cero (serie nula), se devuelve información sobre el estado actual de la conexión. Si no existe ninguna conexión, se devuelve el código de SQL 1427. En otro caso, se devuelve información sobre la conexión en el campo sqlerrmc de la sqlca (tal como se describió anteriormente).

Si no se ha creado una conexión, las API a nivel de instancia se ejecutan para la instancia actual, especificada por la variable de entorno DB2INSTANCE.

Determinadas funciones (db2start, db2stop y todos los servicios de directorio por ejemplo) nunca se ejecutan de forma remota. Es decir, afectan solamente al entorno de la instancia local, tal como se define por el valor de la variable de entorno DB2INSTANCE.

Si existe una conexión y la API se ejecuta con un nombre de nodo, se descarta la conexión actual y se intenta establecer una conexión con el nodo nuevo.

El lugar donde se autentica el nombre de usuario y la contraseña depende del tipo de autentificación de la instancia de destino.

El nodo con el que se debe establecer una conexión también se puede especificar mediante una llamada a la API sqlesetc.

Sintaxis de la API de REXX

```
ATTACH [TO nodename [USER username USING password]]
```

Parámetros de la API de REXX

nodename
Alias de la instancia a la que desea conectarse el usuario. Esta instancia puede tener una entrada que coincide en el directorio de nodo local. La única excepción a esto es la instancia local (especificada por la variable de entorno DB2INSTANCE), que puede especificarse como objeto de una conexión, pero que no se puede utilizar como nombre de nodo en el directorio de nodos.

username
Nombre utilizado por el usuario para conectar con la instancia.

password
Contraseña utilizada para autentificar el nombre de usuario.

sqlecadb - Catalogar una base de datos del directorio de bases de datos del sistema

Almacena información sobre la ubicación de la base de datos en el directorio de bases de datos del sistema. La base de datos puede estar ubicada en la estación de trabajo local o en un servidor de particiones de base de datos remoto.

Ámbito

Esta API afecta al directorio de bases de datos del sistema. En un entorno de bases de datos particionadas, cuando se cataloga una base de datos local en el directorio de bases de datos del sistema, debe llamarse a esta API desde un servidor de particiones de base de datos donde resida la base de datos.

Autorización

Una de las siguientes:

- SYSADM
- SYSCTRL

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlecadb (
        _SQLOLDCHAR * pDbName,
        _SQLOLDCHAR * pDbAlias,
        unsigned char Type,
        _SQLOLDCHAR * pNodeName,
        _SQLOLDCHAR * pPath,
        _SQLOLDCHAR * pComment,
        unsigned short Authentication,
        _SQLOLDCHAR * pPrincipal,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgcadb (
        unsigned short PrinLen,
        unsigned short CommentLen,
        unsigned short PathLen,
        unsigned short NodeNameLen,
        unsigned short DbAliasLen,
        unsigned short DbNameLen,
        struct sqlca * pSqlca,
        _SQLOLDCHAR * pPrinName,
        unsigned short Authentication,
        _SQLOLDCHAR * pComment,
        _SQLOLDCHAR * pPath,
        _SQLOLDCHAR * pNodeName,
        unsigned char Type,
        _SQLOLDCHAR * pDbAlias,
        _SQLOLDCHAR * pDbName);
```

Parámetros de la API sqlecadb

pDbName
Entrada. Serie que contiene el nombre de la base de datos.

pDbAlias
Entrada. Serie que contiene un alias para la base de datos.

Type Entrada. Carácter individual que indica si la base de datos es indirecta, remota o se cataloga a través de DCE. Los valores válidos (definido en sqlenv.h) son:

SQL_INDIRECT
Especifica que la base de datos reside en la instancia local.

SQL_REMOTE
Especifica que la base de datos reside en otra instancia.

SQL_DCE
Especifica que la base de datos se cataloga a través de DCE.

pNodeName
Entrada. Serie que contiene el nombre del servidor de partición de base de datos donde la base de datos está ubicada. Puede ser NULL.

Nota: Si no se especifica **pPath** ni **pNodeName**, se considera que la base de datos es local y que su ubicación es la especificada en el parámetro de configuración **dftdbpath**del gestor de bases de datos.

pPath

Entrada. Serie que, en sistemas basados en Linux y UNIX, especifica el nombre de la vía de acceso en la que reside la base de datos que se está catalogando. La longitud máxima es de 215 caracteres.

En el sistema operativo Windows, esta serie especifica la letra de la unidad en la que reside la base de datos que se está catalogando.

Si se proporciona un puntero NULL, se considera que la vía de acceso por omisión de la base de datos es la especificada en el parámetro de configuración del gestor de bases de datos **dftdbpath**.

pComment

Entrada. Serie que contiene una descripción opcional de la base de datos. Una serie nula indica que no hay comentarios. La longitud máxima de una serie de comentario es de 30 caracteres.

Authentication

Entrada. Contiene el tipo de autentificación especificado para la base de datos. La autentificación es un proceso que verifica que el usuario sea quien dice ser. El acceso a objetos de base de datos depende de la autentificación del usuario. Los valores válidos (definidos en sqlenv.h) son:

SQL_AUTHENTICATION_SERVER
Especifica que la autentificación se realiza en el servidor de particiones de base de datos que contiene la base de datos de destino.

SQL_AUTHENTICATION_CLIENT
Especifica que la autentificación se realiza en el servidor de particiones de base de datos donde se invoca la aplicación.

SQL_AUTHENTICATION_KERBEROS
Especifica que la autentificación se realiza utilizando el mecanismo de seguridad de Kerberos.

SQL_AUTHENTICATION_NOT_SPECIFIED
No se especifica ninguna autentificación.

SQL_AUTHENTICATION_SVR_ENCRYPT
Especifica que la autentificación se realiza en el servidor de particiones de base de datos donde reside la base de datos de destino, y que la contraseña de autentificación se debe cifrar.

SQL_AUTHENTICATION_DATAENC
Especifica que la autentificación tiene lugar en el servidor de particiones de base de datos que contiene la base de datos de destino y que las conexiones deben utilizar el cifrado de datos.

SQL_AUTHENTICATION_GSSPLUGIN
Especifica que la autentificación se realiza utilizando un mecanismo de seguridad externo basado en un plugin de la API de GSS.

SQL_AUTHENTICATION_SVRENC_AESO
Especifica que la autenticación se realiza en el servidor de particiones de base de datos que contiene la base de datos de

destino, y que el ID de usuario y la contraseña de autenticación se cifran utilizando el algoritmo de cifrado AES (Advanced Encryption Standard). Este tipo de autentificación está disponible como de DB2 Versión 9.5 Fixpack 3.

Este parámetro se puede establecer en SQL_AUTHENTICATION_NOT_SPECIFIED, excepto cuando se cataloga una base de datos que reside en un servidor DB2 Versión 1.

La especificación del tipo de autentificación en el catálogo de la base de datos produce una mejora del rendimiento durante una conexión.

pPrincipal
Entrada. Serie que contiene el nombre principal del servidor DB2 en el que reside la base de datos. Este valor sólo debe especificarse cuando **authentication** es SQL_AUTHENTICATION_KERBEROS.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlgcadb

PrinLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud del nombre principal, expresada en bytes. El valor se establece en cero si no se proporciona ningún principal. Este valor debe ser distinto de cero solamente si **authentication** se ha especificado como SQL_AUTHENTICATION_KERBEROS.

CommentLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud en bytes del comentario. El valor se establece en 0 si no se proporciona ningún comentario.

PathLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud de la vía de acceso del directorio de bases de datos locales, expresada en bytes. El valor se establece en 0 si no se proporciona ninguna vía de acceso.

NodeNameLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud en bytes del nodo. El valor se establece en 0 si no se proporciona ningún nombre de nodo.

DbAliasLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud, expresada en bytes, del alias de base de datos.

DbNameLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud en bytes del nombre de base de datos.

pPrinName
Entrada. Serie que contiene el nombre principal del servidor DB2 en el que reside la base de datos. Este valor sólo debe especificarse cuando **authentication** es SQL_AUTHENTICATION_KERBEROS.

Notas de uso

Utilice CATALOG DATABASE para catalogar bases de datos ubicadas en nodos locales o remotos, volver a catalogar bases de datos que se han descatalogado anteriormente o mantener múltiples alias para una base de datos (independientemente de la ubicación de la base de datos).

DB2 cataloga las bases de datos automáticamente cuando éstas se crean. Cataloga una entrada para la base de datos en el directorio de bases de datos locales y otra entrada en el directorio de bases de datos del sistema. Si la base de datos se crea desde un cliente remoto (o un cliente que se está ejecutando desde una instancia diferente en la misma máquina), también se realiza una entrada en el directorio de bases de datos del sistema en la instancia del cliente.

Las bases de datos creadas en la instancia actual (tal como está definida por el valor de la variable de entorno **DB2INSTANCE**) están catalogadas como indirectas. Las bases de datos creadas en otras instancias se catalogan como remotas (aunque residan físicamente en la misma máquina).

CATALOG DATABASE automáticamente crea un directorio de bases de datos del sistema si no existe ninguno. Este directorio se almacena en la vía de acceso que contiene la instancia del gestor de bases de datos que se está utilizando. El directorio de bases de datos del sistema se mantiene fuera de la base de datos. Cada entrada del directorio contiene:

- Alias
- Tipo de autentificación
- Comentario
- Base de datos
- Tipo de entrada
- Directorio de bases de datos locales (cuando se cataloga una base de datos local)
- Nombre del nodo (cuando se cataloga una base de datos remota)
- Información de release

Si una base de datos se cataloga con el parámetro de **tipo** establecido en SQL_INDIRECT, no se tendrá en cuenta el valor del parámetro de **autenticación** proporcionado, y la autentificación se establecerá en SQL_AUTHENTICATION_NOT_SPECIFIED en el directorio.

Si la colocación en antememoria de directorios está habilitada, los archivos de directorios de bases de datos, nodos y DCS se almacenarán en la antememoria. La antememoria de directorio de una aplicación se crea durante la primera búsqueda de directorio. Dado que la antememoria sólo se renueva cuando la aplicación modifica alguno de los archivos de directorio, puede que los cambios de directorio efectuados por otras aplicaciones no entren en vigor hasta que se haya reiniciado la aplicación. Para renovar la memoria caché de DB2 (servidor solamente), detenga (db2stop) y luego reinicie (db2start) el gestor de bases de datos. Para renovar la antememoria de directorios para otra aplicación, detenga dicha aplicación y, a continuación, reiníciela.

Sintaxis de la API para REXX

```
CATALOG DATABASE dbname [AS alias] [ON path|AT NODE nodename]
[AUTHENTICATION authentication] [WITH "comment"]
CATALOG GLOBAL DATABASE db_global_name AS alias
USING DIRECTORY {DCE} [WITH "comment"]
```

Parámetros de la API para REXX

dbname
Nombre de la base de datos que se debe catalogar.

alias Nombre alternativo de la base de datos. Si no se especifica un alias, se utiliza el nombre de base de datos como alias.

vía-acceso
Vía de acceso donde reside la base de datos que se está catalogando.

nodename
Nombre de la estación de trabajo remota donde reside la base de datos que se está catalogando.

Nota: Si no se especifica **path** ni **nodename**, se considera que la base de datos es local y que su ubicación es la especificada en el parámetro de configuración **dftdbpath**del gestor de bases de datos.

authentication
Lugar donde se debe realizar la autentificación. Los valores válidos son:

SERVER
La autentificación se realiza en el servidor de particiones de base de datos donde reside la base de datos de destino. Es el valor por omisión.

CLIENT
La autentificación se realiza en el servidor de particiones de base de datos donde se invoca la aplicación.

KERBEROS
Especifica que la autentificación se realiza utilizando el mecanismo de seguridad de Kerberos.

NOT_SPECIFIED
No se especifica ninguna autentificación.

SVR_ENCRYPT
Especifica que la autentificación se realiza en el servidor de particiones de base de datos donde reside la base de datos de destino, y que el ID de usuario y la contraseña de autentificación deben cifrarse.

DATAENC
Especifica que la autentificación tiene lugar en el servidor de particiones de base de datos que contiene la base de datos de destino y que las conexiones deben utilizar el cifrado de datos.

GSSPLUGIN
Especifica que la autentificación se realiza utilizando un mecanismo de seguridad externo basado en un plugin de la API de GSS.

SQL_AUTHENTICATION_SVRENC_AESO
Especifica que la autenticación se realiza en el servidor de particiones de base de datos que contiene la base de datos de destino, y que el ID de usuario y la contraseña de autenticación se cifran utilizando el algoritmo de cifrado AES.

comment
Describe la base de datos o la entrada de base de datos del directorio de bases de datos del sistema. La longitud máxima de una serie de

comentario es de 30 caracteres. No se permite ningún retorno de carro o carácter de salto de línea. El texto de comentario debe escribirse entre comillas dobles.

db_global_name
Nombre totalmente calificado que identifica de forma exclusiva a la base de datos en el espacio de nombres de DCE.

DCE Servicio global de directorio que se está utilizando.

Ejemplos de REXX

```
call SQLDBS 'CATALOG GLOBAL DATABASE /.../cell1/subsys/database/DB3
AS dbtest USING DIRECTORY DCE WITH "Base de datos de ejemplo"'
```

sqlecran - Crear una base de datos en un servidor de particiones de base de datos

Crea una base de datos solamente en el servidor de particiones de base de datos que ha llamado a la API. Esta API no está destinada al uso general. Por ejemplo, debe utilizarse con db2Restore si la partición de base de datos de un servidor de particiones de base de datos estaba dañada y se tiene que volver a crear. Un uso incorrecto de esta API puede producir incoherencias en el sistema, por lo que se debe utilizar siempre con precaución.

Nota: Si esta API se utiliza para volver a crear una partición de base de datos que se había descartado (porque estaba dañada), la base de datos que está en este servidor de particiones de base de datos estará en el estado de pendiente de restauración. Después de volver a crear la partición de base de datos, la base de datos debe restaurarse inmediatamente en este servidor de particiones de base de datos.

Ámbito

Esta API sólo afecta al servidor de particiones de base de datos en el que se invoca la API.

Autorización

Una de las siguientes:

- sysadm
- sysctrl

Conexión necesaria

Instancia. Para crear una base de datos en otro servidor de particiones de base de datos, es necesario conectarse primero a dicho servidor de particiones de base de datos. Esta API establece temporalmente una conexión de base de datos durante el proceso.

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlecran (
        char * pDbName,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgcran (
        unsigned short reservedLen,
        unsigned short dbNameLen,
        struct sqlca * pSqlca,
        void * pReserved,
        char * pDbName);
```

Parámetros de la API sqlecran

pDbName
Entrada. Serie que contiene el nombre de la base de datos que se debe crear. No debe ser NULL.

pReserved
Entrada. Puntero de reserva que se establece en un valor nulo o que apunta a cero. Reservado para una utilización futura.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlgcran

reservedLen
Entrada. Reservado para la longitud de pReserved.

dbNameLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud del nombre de la base de datos, expresada en bytes.

Notas de uso

Cuando la base de datos se crea satisfactoriamente, se coloca en el estado de restauración pendiente. Para poder ser utilizada, la base de datos se debe restaurar en el servidor de particiones de base de datos.

Sintaxis de la API de REXX

Esta API se puede invocar desde REXX mediante la interfaz SQLDB2.

sqlecrea - Crear una base de datos

Inicializa una base de datos nueva con una secuencia de clasificación opcional definida por el usuario, crea los tres espacios de tablas iniciales, crea las tablas del sistema y asigna la anotación cronológica de recuperación.

Ámbito

En un entorno de bases de datos particionadas, esta API afecta a todos los servidores de particiones de base de datos que están listados en el archivo `db2nodes.cfg`.

El servidor de particiones de base de datos desde el que se llama a esta API se convierte en la partición de catálogo para la nueva base de datos.

Autorización

Una de las siguientes:

- sysadm
- sysctrl

Conexión necesaria

Instancia. Para crear una base de datos en otro nodo (remoto), es necesario conectarse primero a dicho nodo. Esta API establece temporalmente una conexión de base de datos durante el proceso.

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlecrea (
        char * pDbName,
        char * pLocalDbAlias,
        char * pPath,
        struct sqledbdesc * pDbDescriptor,
        SQLEDBTERRITORYINFO * pTerritoryInfo,
        char Reserved2,
        void * pDbDescriptorExt,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgcrea (
        unsigned short PathLen,
        unsigned short LocalDbAliasLen,
        unsigned short DbNameLen,
        struct sqlca * pSqlca,
        void * pReserved1,
        unsigned short Reserved2,
        SQLEDBTERRITORYINFO * pTerritoryInfo,
        struct sqledbdesc * pDbDescriptor,
        char * pPath,
        char * pLocalDbAlias,
        char * pDbName);
```

Parámetros de la API sqlecrea

pDbName
Entrada. Serie que contiene el nombre de la base de datos. Es el nombre de base de datos que se catalogará en el directorio de bases de datos del sistema. Una vez que la base de datos se ha creado satisfactoriamente en el directorio de bases de datos del sistema del servidor, se cataloga automáticamente en el directorio de bases de datos del sistema con un alias de base de datos idéntico al nombre de la base de datos. No debe ser NULL.

pLocalDbAlias
Entrada. Serie que contiene el alias que debe colocarse en el directorio de bases de del sistema del cliente. Puede ser NULL. Si no se especifica ningún alias local, el valor por omisión es el nombre de la base de datos.

pPath Entrada. En sistemas Linux y UNIX, especifica la vía de acceso en la que se debe crear la base de datos. Si no se especifica ninguna vía de acceso, la base de datos se crea en la vía de acceso de base de datos por omisión que se especifica en el archivo de configuración del gestor de bases de datos (parámetro dftdbpath). En el sistema operativo Windows, especifica la letra de la unidad en la que se debe crear la base de datos. Puede ser NULL.

Nota: Para entornos de bases de datos particionadas, no se deberá crear una base de datos en un directorio montado en NFS. Si no se especifica ninguna vía de acceso, asegúrese de que el parámetro de configuración dftdbpath del gestor de bases de datos no se establezca en una vía de acceso montada con NFS (por ejemplo, en sistemas basados en UNIX, no deberá especificar el directorio $HOME del propietario de la instancia). La vía de acceso especificada para esta API en un entorno de bases de datos particionadas no puede ser una vía de acceso relativa.

pDbDescriptor
Entrada. Un puntero al bloque de descripción de base de datos utilizado al crear la base de datos. El bloque de descripción de base de datos puede utilizarlo el usuario para suministrar valores que se almacenan permanentemente en el archivo de configuración de la base de datos.

Los valores suministrados son el orden de clasificación, un comentario de base de datos o una definición de espacio de tablas. El valor suministrado puede ser NULL si no desea suministrar ningún valor. Para obtener información acerca de los valores que pueden suministrarse mediante este parámetro, consulte el tema relativo a la estructura de datos SQLEDBDESC.

pTerritoryInfo
Entrada. Un puntero a la estructura sqledbterritoryinfo, que contiene el entorno local y el conjunto de códigos de la base de datos. Puede ser NULL. El conjunto de códigos por omisión para una base de datos es UTF-8 (Unicode). Si se necesita un conjunto de códigos y un territorio determinados para una base de datos, el conjunto de códigos y el territorio deseados deben especificarse mediante la estructura sqledbterritoryinfo. Si este campo es NULL, se permite uno de los siguientes valores como valor de clasificación para la base de datos (sqlcode 1083): NULL, SQL_CS_SYSTEM, SQL_CS_IDENTITY_16BIT, SQL_CS_UCA400_NO, SQL_CS_UCA400_LTH, SQL_CS_UCA400_LSK o SQL_CS_UNICODE.

Reserved2
Entrada. Reservado para una utilización futura.

pDbDescriptorExt
Entrada. Este parámetro hace referencia a un bloque de descripción de base de datos ampliado (sqledbdescext) utilizado al crear la base de datos. El bloque de descripción de base de datos ampliado habilita el almacenamiento automático para una base de datos, elige un tamaño de página por omisión para la base de datos y especifica valores para los nuevos atributos de espacio de tablas que se han especificado. Si se establece en un valor nulo o en cero, se elige un tamaño de página por omisión de 4096 bytes para la base de datos y se habilita el almacenamiento automático.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlgcrea

PathLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud de la vía de acceso, expresada en bytes. El valor se establece en cero si no se proporciona ninguna vía de acceso.

LocalDbALiasLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud del alias de base de datos local, expresada en bytes. El valor se establece en cero si no se proporciona ningún alias local.

DbNameLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud del nombre de la base de datos, expresada en bytes.

Notas de uso

CREATE DATABASE:

- Crea una base de datos en el subdirectorio especificado. En un entorno de bases de datos particionadas, crea la base de datos en todos los servidores de particiones de base de datos listados en `db2nodes.cfg` y crea en cada servidor de partición de base de datos un directorio $DB2INSTANCE/NODExxxx bajo el subdirectorio especificado, donde xxxx representa el número del servidor de particiones de base de datos locales. En un entorno de una sola partición, crea un directorio $DB2INSTANCE/NODE0000 bajo el subdirectorio especificado.
- Crea las tablas de catálogos del sistema y la anotación cronológica de recuperación.
- Cataloga la base de datos en los directorios de bases de datos siguientes:
  - directorio de bases de datos locales del servidor en la vía de acceso indicada por pPath o, si no se especifica la vía de acceso, la vía de acceso de base de datos por omisión definida en el archivo de configuración del sistema gestor de bases de datos. En cada sistema de archivos que contiene una base de datos reside un directorio de bases de datos local.
  - directorio de bases de datos del sistema del servidor para la instancia conectada. La entrada de directorio resultante contendrá el nombre de base de datos y un alias de base de datos.

    Si se ha llamado a la API desde un cliente remoto, el directorio de bases de datos del sistema del cliente también se actualiza con el nombre de base de datos y un alias.
- Crea un sistema o un directorio de bases de datos local si no existe ninguno de los dos. Si se han especificado, los valores de comentario y de conjunto de códigos se colocan en ambos directorios.
- Almacena el conjunto de códigos, el territorio y la secuencia de clasificación que se han especificado. Se establece un distintivo en el archivo de configuración de base de datos si la secuencia de clasificación consta de valores exclusivos o si es la secuencia de identidad.
- Crea los esquemas denominados SYSCAT, SYSFUN, SYSIBM y SYSSTAT con SYSIBM como propietario. El servidor de particiones de base de datos desde el que se llama a esta API se convierte en la partición de catálogo para la nueva base de datos. Se crean automáticamente dos grupos de particiones de base de datos: IBMDEFAULTGROUP e IBMCATGROUP.
- Vincula los archivos de vinculación del gestor de bases de datos definidos anteriormente a la base de datos (éstos se listan en db2ubind.lst). Si uno o varios de estos archivos no se vinculan satisfactoriamente, sqlecrea devuelve un aviso

en SQLCA y proporciona información acerca de las vinculaciones que han fallado. Si falla una vinculación, el usuario puede realizar la acción correctiva y vincular manualmente el archivo anómalo. De todos modos se crea la base de datos. Un esquema denominado NULLID se crea implícitamente al efectuar las vinculaciones con el privilegio CREATEIN otorgado a PUBLIC, en el caso de que no se haya seleccionado la opción RESTRICTIVE.

- Crea los espacios de tablas SYSCATSPACE, TEMPSPACE1 y USERSPACE1. El espacio de tablas SYSCATSPACE sólo se crea en la partición de catálogo. Todas las particiones de base de datos tienen las mismas definiciones de espacio de tablas.
- Otorga lo siguiente:
  - Autorización DBADM, CONNECT, CREATETAB, BINDADD, CREATE_NOT_FENCED, IMPLICIT_SCHEMA y LOAD al creador de la base de datos
  - Autorización CONNECT, CREATETAB, BINDADD e IMPLICIT_SCHEMA a PUBLIC
  - Privilegio USE en el espacio de tablas USERSPACE1 a PUBLIC
  - Privilegio SELECT en cada catálogo de sistema a PUBLIC
  - Privilegio BIND y EXECUTE a PUBLIC para cada programa de utilidad vinculado satisfactoriamente
  - Privilegio EXECUTE WITH GRANT a PUBLIC sobre todas las funciones del esquema SYSFUN.
  - Privilegio EXECUTE a PUBLIC sobre todos los procedimientos del esquema SYSIBM.

  Nota: Si la opción RESTRICTIVE está presente hará que el parámetro de configuración de la base de datos RESTRICT_ACCESS se establezca en YES y que no se otorguen automáticamente privilegios ni autorizaciones a PUBLIC. Para obtener información más detallada, consulte la opción RESTRICTIVE del mandato CREATE DATABASEopción .

Con la autorización dbadm, se pueden otorgar estos privilegios (y revocarlos) a otros usuarios o a PUBLIC. Si otro administrador con autorización sysadm o dbadm sobre la base de datos revoca estos privilegios, el creador de la base de datos los retiene a pesar de todo.

En un entorno de bases de datos particionadas, el gestor de bases de datos crea un subdirectorio, $DB2INSTANCE/NODExxxx, bajo la vía de acceso especificada o por omisión en todos los servidores de particiones de base de datos. xxxx es el número de nodo tal como está definido en el archivo `db2nodes.cfg` (es decir, el nodo 0 se convierte en NODE0000). Los subdirectorios SQL00001 a SQLnnnnn residirán en esta vía de acceso. Esto asegura que los objetos de base de datos asociados a servidores de particiones de base de datos diferentes se almacenen en directorios diferentes (incluso si todos los servidores de particiones de base de datos comparten el subdirectorio $DB2INSTANCE bajo la vía de acceso especificada o por omisión).

En sistemas operativos Windows y AIX, la longitud del nombre del juego de códigos está limitada a un máximo de 9 caracteres. Por ejemplo, especifique un nombre de conjunto de códigos como ISO885915 en lugar de ISO8859-15.

La API sqlecrea acepta una estructura de datos llamada Bloque de descriptor de base de datos (SQLEDBDESC). Puede definir su propia secuencia de clasificación dentro de la estructura.

Nota: Sólo puede definir su propia secuencia de clasificación para una base de datos de un solo byte.

Para especificar una secuencia de clasificación para una base de datos:

- Pase la estructura SQLEDBDESC deseada, o
- Pase un puntero NULL. Se utiliza la secuencia de clasificación del sistema operativo (basada en el código de entorno local actual y en la página de códigos). Es igual que especifica un SQLDBCSS igual a SQL_CS_SYSTEM (0).

La ejecución del mandato CREATE DATABASE fallará si la aplicación ya está conectada a una base de datos.

Si la estructura del bloque de descripción de base de datos no está establecida correctamente, se devolverá un mensaje de error.

El valor más relevante del bloque de descripción de base de datos debe establecerse en el valor simbólico SQLE_DBDESC_2 (definido en sqlenv). Los siguientes órdenes de clasificación definidos por usuario de ejemplo están disponibles en los archivos de inclusión del lenguaje principal.

sqle819a
Si la página de códigos de la base de datos es 819 (ISO Latin/1), esta secuencia provocará la clasificación según el CCSID 500 (EBCDIC internacional) del sistema principal.

sqle819b
Si la página de códigos de la base de datos es 819 (ISO Latin/1), esta secuencia provocará la clasificación según el CCSID 037 (EBCDIC inglés de Estados Unidos) del sistema principal.

sqle850a
Si la página de códigos de la base de datos es 850 (ASCII Latin/1), esta secuencia provocará la clasificación según el CCSID 500 (EBCDIC internacional) del sistema principal.

sqle850b
Si la página de códigos de la base de datos es 850 (ASCII Latin/1), esta secuencia provocará la clasificación según el CCSID 037 (EBCDIC inglés de Estados Unidos) del sistema principal.

sqle932a
Si la página de códigos de la base de datos es 932 (ASCII Japonés), esta secuencia provocará la clasificación según el CCSID 5035 (EBCDIC japonés) del sistema principal.

sqle932b
Si la página de códigos de la base de datos es 932 (ASCII Japonés), esta secuencia provocará la clasificación según el CCSID 5026 (EBCDIC japonés) del sistema principal.

La secuencia de clasificación especificada durante la creación de la base de datos no se puede modificar después. Determina cómo se comparan las series de caracteres. Esto afecta a la estructura de los índices y a los resultados de las consultas. En una base de datos Unicode, las tablas y vistas del catálogo siempre se crean con la clasificación IDENTITY, sin tener en cuenta la clasificación especificada en la llamada a la creación de base de datos. En una base de datos no Unicode, las tablas y vistas del catálogo se crean con la clasificación de base de datos.

Utilice sqlecadb para definir diferentes nombres de alias para la nueva base de datos.

El Asesor de configuración se invoca por omisión durante el proceso de creación de bases de datos, a menos que se haya indicado específicamente no hacerlo.

Sintaxis de la API de REXX

```
CREATE DATABASE dbname [ON path] [ALIAS dbalias]
[USING CODESET codeset TERRITORY territory]
[COLLATE USING {SYSTEM | IDENTITY | USER :udcs}]
[NUMSEGS numsegs] [DFT_EXTENT_SZ dft_extentsize]
[CATALOG TABLESPACE <tablespace_definition>]
[USER TABLESPACE <tablespace_definition>]
[TEMPORARY TABLESPACE <tablespace_definition>]
[WITH comment]


Donde <tablespace_definition> significa:
MANAGED BY {
SYSTEM USING :SMS_string |
DATABASE USING :DMS_string }
[ EXTENTSIZE number_of_pages ]
[ PREFETCHSIZE number_of_pages ]
[ OVERHEAD number_of_milliseconds ]
[ TRANSFERRATE number_of_milliseconds ]
```

Parámetros de la API de REXX

dbname
Nombre de la base de datos.

dbalias
Alias de la base de datos.

path Vía de acceso donde debe crearse la base de datos. Si no se especifica ninguna vía de acceso, la base de datos se crea en la vía de acceso de base de datos por omisión que se especifica en el archivo de configuración del gestor de bases de datos (parámetro de configuración dftdbpath).

Nota: Para entornos de bases de datos particionadas, no se deberá crear una base de datos en un directorio montado en NFS. Si no se especifica ninguna vía de acceso, asegúrese de que el parámetro de configuración dftdbpath del gestor de bases de datos no se establezca en una vía de acceso montada con NFS (por ejemplo, en sistemas basados en UNIX, no deberá especificar el directorio $HOME del propietario de la instancia). La vía de acceso especificada para esta API en un entorno de bases de datos particionadas no puede ser una vía de acceso relativa.

codeset
Conjunto de códigos que se debe utilizar para los datos entrados en la base de datos.

territory
Código de territorio (entorno local) que se debe utilizar para los datos entrados en la base de datos.

SYSTEM
Para bases de datos no Unicode, es la opción por omisión, con la secuencia de clasificación basada en el territorio de la base de datos. Para bases de datos Unicode, esta opción es equivalente a la opción IDENTITY.

IDENTITY
Secuencia de clasificación de identidad, en la que se comparan las series byte por byte. Es el valor por omisión para bases de datos Unicode.

USER udcs
La aplicación que llama especifica el orden de clasificación en una variable de lenguaje principal que contiene una serie de 256 bytes que define el orden de clasificación.

numsegs
Número de directorios (contenedores de espacios de tablas) que se crearán y utilizarán para almacenar los archivos de tabla de base de datos para cualquier espacio de tablas SMS por omisión.

dft_extentsize
Especifica el tamaño de extensión por omisión de los espacios de tablas de la base de datos.

SMS_string
Variable de lenguaje principal REXX compuesta que identifica uno o más contenedores que pertenecerán al espacio de tablas y donde se almacenarán los datos del espacio de tablas. En las expresiones que siguen, XXX representa el nombre de la variable de lenguaje principal. Tenga en cuenta que cada uno de los nombres de directorio no puede sobrepasar los 254 bytes de longitud.

XXX.0 Número de directorios especificados

XXX.1 Primer nombre de directorio para el espacio de tablas SMS

XXX.2 Segundo nombre de directorio para el espacio de tablas SMS

XXX.3 y así sucesivamente.

DMS_string
Variable de lenguaje principal REXX compuesta que identifica uno o más contenedores que pertenecerán al espacio de tablas, donde se almacenarán los datos del espacio de tablas, los tamaños de contenedores (especificados en un número de páginas de 4 KB) y los tipos (archivo o dispositivo). Los dispositivos especificados (no archivos) deben existir previamente. En las expresiones que siguen, XXX representa el nombre de la variable de lenguaje principal. Tenga en cuenta que cada uno de los nombres de contenedor no puede sobrepasar los 254 bytes de longitud.

XXX.0 Número de series de la variable de lenguaje principal REXX (número de elementos de primer nivel)

XXX.1.1
Tipo del primer contenedor (archivo o dispositivo)

XXX.1.2
Primer nombre de archivo o de dispositivo

XXX.1.3
Tamaño (en páginas) del primer contenedor

XXX.2.1
Tipo del segundo contenedor (archivo o dispositivo)

XXX.2.2
Segundo nombre de archivo o de dispositivo

XXX.2.3
Tamaño (en páginas) del segundo contenedor

XXX.3.1
y así sucesivamente.

EXTENTSIZE número_de_páginas
Número de páginas de 4 KB que se escribirán en un contenedor antes de pasar al próximo contenedor.

PREFETCHSIZE número_de_páginas
Número de páginas de 4 KB que se leerán del espacio de tablas cuando se realice la captación previa de los datos.

OVERHEAD número_de_milisegundos
Número que especifica la actividad general del controlador de E/S, búsqueda de disco y tiempo de latencia en milisegundos.

TRANSFERRATE número_de_milisegundos
Número que especifica el tiempo, en milisegundos, para la lectura de una página de 4 KB en la memoria.

comment
Descripción de la base de datos o la entrada de base de datos del directorio del sistema. No utilice caracteres de retorno de carro ni de salto de línea en el comentario. Asegúrese de entrecomillar el texto del comentario. El tamaño máximo es de 30 caracteres.

sqlectnd - Catalogar una entrada en el directorio de nodos

Almacena información en el directorio de nodos sobre la ubicación de una instancia de servidor DB2 de acuerdo con el protocolo de comunicaciones utilizado para acceder a esa instancia. La información es necesaria para establecer una conexión de base de datos o asociación entre una aplicación y una instancia de servidor.

Autorización

Una de las siguientes:

- sysadm
- sysctrl

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlectnd (
        struct sqle_node_struct * pNodeInfo,
        void * pProtocolInfo,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgctnd (
        struct sqlca * pSqlca,
        struct sqle_node_struct * pNodeInfo,
        void * pProtocolInfo);
```

Parámetros de la API sqlectnd

pNodeInfo
Entrada. Puntero a una estructura del directorio de nodos.

pProtocolInfo
Entrada. Puntero a la estructura de protocolos:

- SQLE-NODE-LOCAL
- SQLE-NODE-NPIPE
- SQLE-NODE-TCPIP

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

DB2 crea el directorio de nodos en la primera llamada a esta API si el directorio de nodos no existe. En el sistema operativo Windows, el directorio de nodos se almacena en el directorio de la instancia que se esté utilizando. En sistemas basados en UNIX, se almacena en el directorio de instalación de DB2 (sqllib, por ejemplo).

Si la colocación en antememoria de directorios está habilitada, los archivos de directorios de bases de datos, nodos y DCS se almacenarán en la antememoria. La antememoria de directorio de una aplicación se crea durante la primera búsqueda de directorio. Dado que la antememoria sólo se renueva cuando la aplicación modifica alguno de los archivos de directorio, puede que los cambios de directorio efectuados por otras aplicaciones no entren en vigor hasta que se haya reiniciado la aplicación. Para renovar la antememoria compartida de DB2 (servidor solamente), detenga (mandato db2stop) y luego reinicie (mandato db2start) el gestor de bases de datos. Para renovar la antememoria de directorios para otra aplicación, detenga dicha aplicación y, a continuación, reiníciela.

Sintaxis de la API de REXX, opción 1

```
CATALOG LOCAL NODE nodename INSTANCE instance_name [WITH comment]
```

Parámetros de la API de REXX, opción 1

nodename
Alias del nodo que se debe catalogar.

instance_name
Nombre de la instancia que se debe catalogar.

comment
Descripción opcional asociada con esta entrada del directorio de nodos. No incluya un carácter CR/LF en un comentario. La longitud máxima es de 30 caracteres. El texto de comentario debe escribirse entre comillas dobles.

Sintaxis de la API de REXX, opción 2

```
CATALOG NPIPE NODE nodename REMOTE computer_name INSTANCE instance_name
```

Parámetros de la API de REXX, opción 2

nodename
Alias del nodo que se debe catalogar.

computer_name
Nombre de sistema del nodo donde reside la base de datos de destino.

instance_name
Nombre de la instancia que se debe catalogar.

Sintaxis de la API de REXX, opción 3

```
CATALOG TCPIP NODE nodename REMOTE hostname SERVER servicename
[WITH comment]
```

Parámetros de la API de REXX, opción 3

nodename
Alias del nodo que se debe catalogar.

hostname
Nombre de sistema principal o dirección IPv4 o dirección IPv6 del nodo donde reside la base de datos de destino.

servicename
Nombre de servicio de la instancia del gestor de bases de datos del nodo remoto, o número de puerto asociado a ese nombre de servicio.

comment
Descripción opcional asociada con esta entrada del directorio de nodos. No incluya un carácter CR/LF en un comentario. La longitud máxima es de 30 caracteres. El texto de comentario debe escribirse entre comillas dobles.

Sintaxis de la API de REXX, opción 4

```
CATALOG TCPIP4 NODE nodename REMOTE hostname SERVER servicename
[WITH comment]
```

Parámetros de la API de REXX, opción 4

nodename
Alias del nodo que se debe catalogar.

hostname
Nombre de sistema principal o dirección IPv4 o dirección IPv6 del nodo donde reside la base de datos de destino.

servicename
Nombre de servicio de la instancia del gestor de bases de datos del nodo remoto, o número de puerto asociado a ese nombre de servicio.

comment
Descripción opcional asociada con esta entrada del directorio de nodos. No incluya un carácter CR/LF en un comentario. La longitud máxima es de 30 caracteres. El texto de comentario debe escribirse entre comillas dobles.

Sintaxis de la API de REXX, opción 5

```
CATALOG TCPIP6 NODE nodename REMOTE hostname SERVER servicename
[WITH comment]
```

Parámetros de la API de REXX, opción 5

nodename
Alias del nodo que se debe catalogar.

hostname
Nombre de sistema principal o dirección IPv4 o dirección IPv6 del nodo donde reside la base de datos de destino.

servicename
Nombre de servicio de la instancia del gestor de bases de datos del nodo remoto, o número de puerto asociado a ese nombre de servicio.

comment
Descripción opcional asociada con esta entrada del directorio de nodos. No incluya un carácter CR/LF en un comentario. La longitud máxima es de 30 caracteres. El texto de comentario debe escribirse entre comillas dobles.

sqledcgd - Cambiar un comentario de base de datos en el directorio de bases de datos locales o del sistema

Cambia un comentario de base de datos en el directorio de bases de datos del sistema o en el directorio de bases de datos locales. El texto actualmente asociado con un comentario puede sustituirse por texto de comentario nuevo.

Ámbito

Esta API sólo afecta al servidor de particiones de base de datos en el que se ejecuta la API.

Autorización

Una de las siguientes:

- sysadm
- sysctrl

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqledcgd (
        _SQLOLDCHAR * pDbAlias,
        _SQLOLDCHAR * pPath,
        _SQLOLDCHAR * pComment,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdcgd (
        unsigned short CommentLen,
        unsigned short PathLen,
        unsigned short DbAliasLen,
        struct sqlca * pSqlca,
        _SQLOLDCHAR * pComment,
        _SQLOLDCHAR * pPath,
        _SQLOLDCHAR * pDbAlias);
```

Parámetros de la API sqledcgd

pDbAlias
Entrada. Serie que contiene el alias de la base de datos. Este es el nombre que se cataloga en el directorio de bases de datos locales o del sistema si se especifica la vía de acceso.

pPath Entrada. Serie que contiene la vía de acceso donde reside el directorio de bases de datos locales. Si la vía de acceso especificada es un puntero NULL, se utiliza el directorio de bases de datos del sistema.

El comentario solo se cambia en el directorio de bases de datos locales o del sistema del servidor de particiones de base de datos donde se ejecuta la API. Para cambiar el comentario de base de datos en todos los servidores de particiones de base de datos, ejecute la API en cada servidor de particiones de base de datos.

pComment
Entrada. Serie que contiene una descripción opcional de la base de datos. Una serie nula indica que no hay comentarios. También puede indicar que no hay ningún cambio para un comentario de base de datos existente.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlgdcgd

CommentLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud en bytes del comentario. El valor se establece en cero si no se proporciona ningún comentario.

PathLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud en bytes del parámetro de la vía de acceso. El valor se establece en cero si no se proporciona ninguna vía de acceso.

DbAliasLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud, expresada en bytes, del alias de base de datos.

Notas de uso

El texto de comentario nuevo sustituye al texto existente. Para añadir información, entre el texto de comentario antiguo, seguido del texto nuevo.

Sólo se modifica el comentario para una entrada asociada con el alias de base de datos. Otras entradas con el mismo nombre de base de datos, pero con alias diferentes, no se verán afectadas.

Si se especifica la vía de acceso, el alias de base de datos se debe catalogar en el directorio de bases de datos locales. Si no se especifica la vía de acceso, el alias de base de datos se debe catalogar en el directorio de bases de datos del sistema.

Sintaxis de la API de REXX

```
CHANGE DATABASE database_alias COMMENT [ON path] WITH comment
```

Parámetros de la API de REXX

database_alias
Alias de la base de datos cuyo comentario se debe cambiar.

Para cambiar el comentario en el directorio de bases de datos del sistema, es necesario especificar el alias de base de datos.

Si se especifica la vía de acceso donde reside la base de datos (mediante el parámetro path), escriba el nombre (no el alias) de la base de datos. Utilice este método para cambiar el comentario en el directorio de bases de datos locales.

path Vía de acceso donde reside la base de datos.

comment
Describe la entrada del directorio de bases de datos del sistema o del directorio de bases de datos locales. Se puede entrar cualquier comentario que ayude a describir la base de datos catalogada. La longitud máxima de una serie de comentario es de 30 caracteres. No se permite ningún retorno de carro o carácter de salto de línea. El texto de comentario debe escribirse entre comillas dobles.

sqledpan - Descartar una base de datos de un servidor de particiones de base de datos

Descarta una base de datos de un servidor de particiones de base de datos especificado. Solamente se puede ejecutar en un entorno de bases de datos particionadas.

Ámbito

Esta API sólo afecta al servidor de particiones de base de datos en el que se invoca la API.

Autorización

Una de las siguientes:

- sysadm
- sysctrl

Conexión necesaria

Ninguna. Se establece una conexión de instancia que permanece activa durante la duración de la llamada.

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqledpan (
        char * pDbAlias,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdpan (
        unsigned short Reserved1,
        unsigned short DbAliasLen,
        struct sqlca * pSqlca,
        void * pReserved2,
        char * pDbAlias);
```

Parámetros de la API sqledpan

pDbAlias
Entrada. Serie que contiene el alias de base de datos que debe descartarse. Este nombre se utiliza para hacer referencia al nombre real de la base de datos en el directorio de bases de datos del sistema.

pReserved
Reservado. Debe ser NULL.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlgdpan

Reserved1
Reservado para una utilización futura.

DbAliasLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud, expresada en bytes, del alias de base de datos.

pReserved2
Puntero de reserva que se establece en un valor nulo o que apunta a cero. Reservado para una utilización futura.

Notas de uso

Un uso incorrecto de esta API puede producir incoherencias en el sistema, por lo que se debe utilizar siempre con precaución.

Sintaxis de la API de REXX

Esta API se puede invocar desde REXX mediante la interfaz SQLDB2.

sqledrpd - Descartar base de datos

Suprime el contenido de la base de datos y todos los archivos de anotaciones cronológicas para la base de datos, descataloga la base de datos y suprime el subdirectorio de bases de datos.

Ámbito

Por omisión, esta API afecta a todos los servidores de particiones de base de datos que están listados en el archivo db2nodes.cfg.

Autorización

Una de las siguientes:

- sysadm
- sysctrl

Conexión necesaria

Instancia. No es necesario invocar ATTACH antes de descartar una base de datos remota. Si la base de datos está catalogada como remota, se establece una conexión de instancia al nodo remoto durante el tiempo que dura la llamada.

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqledrpd (
        _SQLOLDCHAR * pDbAlias,
        _SQLOLDCHAR * pReserved2,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdrpd (
        unsigned short Reserved1,
        unsigned short DbAliasLen,
        struct sqlca * pSqlca,
        _SQLOLDCHAR * pReserved2,
        _SQLOLDCHAR * pDbAlias);
```

Parámetros de la API sqledrpd

pDbAlias
Entrada. Serie que contiene el alias de base de datos que debe descartarse. Este nombre se utiliza para hacer referencia al nombre real de la base de datos en el directorio de bases de datos del sistema.

pReserved2
Puntero de reserva que se establece en un valor nulo o que apunta a cero. Reservado para una utilización futura.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlgdrpd

Reserved1
Reservado para una utilización futura.

DbAliasLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud, expresada en bytes, del alias de base de datos.

Notas de uso

La API sqledrpd suprime todos los datos de usuario y archivos de anotaciones. Si los archivos de anotaciones son necesarios para una recuperación en avance después de una operación de restauración, se deben guardar los archivos de anotaciones antes de invocar esta API.

La base de datos no se debe estar utilizando; todos los usuarios deben estar desconectados de la base de datos antes de que ésta se pueda descartar.

Para descartar una base de datos debe estar catalogada en el directorio de bases de datos del sistema. Solamente se elimina del directorio de bases de datos del sistema el alias de base de datos especificado. Si existen otros alias con el mismo nombre de base de datos, sus entradas permanecerán. Si la base de datos que se está descartando es la última entrada del directorio de bases de datos locales, dicho directorio se suprimirá automáticamente.

Si esta API se invoca desde un cliente remoto (o desde una instancia diferente de la misma máquina), el alias especificado se elimina del directorio de bases de datos

del sistema del cliente. El nombre de base de datos correspondiente se elimina del directorio de bases de datos del sistema del servidor.

Sintaxis de la API de REXX

```
DROP DATABASE dbalias
```

Parámetros de la API de REXX

dbalias
Alias de la base de datos que se debe descartar.

sqledrpn - Comprobar si se puede descartar un servidor de particiones de base de datos

Comprueba si una base de datos está utilizando un servidor de particiones de base de datos. Se devuelve un mensaje que indica si se puede descartar el servidor de particiones de base de datos.

Ámbito

Esta API sólo afecta al servidor de particiones de base de datos en el que se ejecuta la API.

Autorización

Una de las siguientes:

- sysadm
- sysctrl

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqledrpn (
        unsigned short Action,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdrpn (
        unsigned short Reserved1,
        struct sqlca * pSqlca,
        void * pReserved2,
        unsigned short Action);
```

Parámetros de la API sqledrpn

Acción
La acción solicitada. El valor válido es: SQL_DROPNODE_VERIFY

pReserved
Reservado. Debe ser NULL.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlgdrpn

Reserved1
Reservado para la longitud de pReserved2.

pReserved2
Puntero de reserva que se establece en un valor nulo o que apunta a 0. Está reservado para una utilización futura.

Notas de uso

Si se devuelve un mensaje que indica que el servidor de particiones de base de datos no está en uso, utilice el mandato db2stop con DROP NODENUM para eliminar la entrada del servidor de particiones de base de datos en el archivo db2nodes.cfg, con lo cual se elimina el servidor del entorno de bases de datos particionadas.

Si se devuelve un mensaje que indica que el servidor de particiones de base de datos está en uso, se deben emprender las acciones siguientes:

1. El servidor de particiones de base de datos que debe descartarse contendrá una partición de base de datos para cada base de datos de la instancia. Si alguna de estas particiones de base de datos contiene datos, redistribuya los grupos de particiones de base de datos que utilicen estas particiones de base de datos. Redistribuya los grupos de particiones de base de datos para trasladar los datos a particiones de base de datos situadas en servidores de particiones de base de datos que no se van a descartar.
2. Una vez redistribuidos los grupos de particiones de base de datos, descarte la partición de base de datos en cada grupo de particiones de base de datos que haga uso de ella. Para ello puede utilizar la opción de la API sqludrdt para descartar nodos, o la sentencia ALTER DATABASE PARTITION GROUP.
3. Descarte cualquier supervisor de sucesos que esté definido en el servidor de particiones de base de datos.
4. Ejecute de nuevo sqledrpn para asegurarse de que la partición de base de datos contenida en el servidor de particiones de base de datos ya no se esté utilizando.

Sintaxis de la API de REXX

Esta API se puede invocar desde REXX mediante la interfaz SQLDB2.

sqledtin - Desconectar de instancia

Elimina la conexión de instancia lógica y termina la conexión de comunicación física si no hay otras conexiones lógicas que utilicen esta capa.

Autorización

Ninguna

Conexión necesaria

Ninguna. Elimina una conexión de instancia existente.

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqledtin (
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdtin (
        struct sqlca * pSqlca);
```

Parámetros de la API sqledtin

pSqlca
Salida. Puntero a la estructura sqlca.

Sintaxis de la API de REXX

```
DETACH
```

sqlefmem - Liberar la memoria asignada por las API sqlbtcq y sqlbmtsq

Libera la memoria asignada por las API de DB2 a petición del llamador. Pensado para su uso con las API sqlbtcq y sqlbmtsq.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlefmem (
        struct sqlca * pSqlca,
        void * pBuffer);

SQL_API_RC SQL_API_FN
  sqlgfmem (
        struct sqlca * pSqlca,
        void * pBuffer);
```

Parámetros de la API sqlefmem

pSqlca
Salida. Puntero a la estructura sqlca.

pBuffer
Entrada. Puntero a la memoria que se debe liberar.

sqlefrce - Desconectar usuarios y aplicaciones del sistema

Fuerza a las aplicaciones o usuarios locales o remotos a salir del sistema para permitir tareas de mantenimiento en un servidor. Atención: si se fuerza una operación que no se puede interrumpir (por ejemplo, una restauración de base de datos), se debe volver a ejecutar satisfactoriamente la operación para que la base de datos pase a estar disponible.

Ámbito

Esta API afecta a todos los servidores de particiones de base de datos que están listados en el archivo db2nodes.cfg.

En un entorno de bases de datos particionadas, no es necesario ejecutar esta API desde la partición coordinadora de la aplicación que se está desconectando. Esta API se puede ejecutar desde cualquier servidor de particiones de base de datos existente en el entorno de bases de datos particionadas.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint

Conexión necesaria

Instancia. Para forzar a los usuarios a salir de un servidor remoto, primero es necesario conectarse a dicho servidor. Si no existe ninguna conexión, esta API se ejecuta localmente.

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlefrce (
        sqlint32 NumAgentIds,
        sqluint32 * pAgentIds,
        unsigned short ForceMode,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgfrce (
        struct sqlca * pSqlca,
        unsigned short ForceMode,
        sqluint32 * pAgentIds,
        sqlint32 NumAgentIds);
```

Parámetros de la API sqlefrce

NumAgentIds
Entrada. Número entero que representa el número de total de usuarios que se deben desconectar. Este número debe ser igual al número de elementos de la matriz de los ID de agente.

Si el valor de este parámetro se establece en SQL_ALL_USERS (definido en sqlenv), se desconectan todas las aplicaciones con conexiones de base de datos o conexiones de instancia. Si su valor se establece en cero, se devuelve un error.

pAgentIds
Entrada. Puntero a una matriz de enteros largos sin signo. Cada entrada describe el ID de agente del usuario de la base de datos correspondiente.

ForceMode
Entrada. Número entero que especifica la modalidad de operación de la API sqlefrce. Sólo se soporta la modalidad asíncrona. Esto significa que la API no espera a que se desconecten todos los usuarios especificados antes de efectuar el retorno. Efectúa el retorno tanto pronto como la API se ha ejecutado satisfactoriamente; en caso contrario, se produce un error. Como resultado, puede existir un corto intervalo entre el momento en el que finaliza la llamada para desconectar la aplicación y el momento en el que se desconectan los usuarios especificados.

El valor de este parámetro se debe establecer en SQL_ASYNCH (definido en sqlenv).

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

El gestor de bases de datos permanece activo para que sus operaciones subsiguientes puedan manejarse sin necesidad de ejecutar db2start.

Para conservar la integridad de la base de datos, sólo se pueden desconectar los usuarios que están desocupados o que están ejecutando operaciones de base de datos que se pueden interrumpir.

Después de emitir un mandato de desconexión, la base de datos seguirá aceptando peticiones de conexión. Pueden ser necesarias operaciones adicionales de desconexión para desconectar completamente todos los usuarios. Se utilizan las funciones del supervisor del sistema de base de datos para recopilar los ID de agente de los usuarios que se deben desconectar.

Cuando la modalidad de desconexión se establece en SQL_ASYNCH (el único valor permitido), la API devuelve inmediatamente el control a la aplicación solicitante.

Se realiza una validación mínima en la matriz de los ID de agente que se deben desconectar. El usuario debe asegurarse de que el puntero especifica una matriz que contiene el número total de elementos especificados. Si el valor de NumAgentIds se establece en SQL_ALL_USERS, no se tiene en cuenta la matriz.

Cuando se desconecta un usuario, se retrotrae una unidad de trabajo para asegurar la coherencia de la base de datos.

Se desconectarán todos los usuarios que se puedan desconectar. Si no se puede encontrar uno o más de los ID de agente, el valor de sqlcode en la estructura sqlca se establece en 1230. Un ID de agente puede no encontrarse, por ejemplo, si el usuario finaliza la sesión entre el momento en el que se obtiene el ID de agente y el momento en el que se invoca sqlefrce. El usuario que realiza la llamada a esta API no se desconecta nunca.

Los ID de agente se reutilizan para desconectar aplicaciones algún tiempo después de que las recopile el supervisor del sistema de base de datos. Por tanto, cuando un usuario finaliza la sesión, otro usuario puede iniciar una y obtener el mismo ID de agente a través de este ciclo de reutilización, con lo que se puede producir la desconexión de un usuario equivocado.

Sintaxis de la API de REXX

```
FORCE APPLICATION {ALL | :agentidarray} [MODE ASYNC]
```

Parámetros de la API de REXX

ALL Se desconectarán todas las aplicaciones. Esto incluye aplicaciones que tienen conexiones de base de datos y aplicaciones que tienen conexiones de instancia.

agentidarray
Variable compuesta de lenguaje principal de REXX que contiene la lista de los ID de agente que se deben desconectar. En la información que sigue a continuación, XXX es el nombre de la variable de lenguaje principal:

- XXX.0
Número de agentes que se deben desconectar

- XXX.1
Primer ID de agente

- XXX.2
Segundo ID de agente

- XXX.3
y así sucesivamente.

ASYNC
Es la única modalidad de operación permitida actualmente. Significa que sqlefrce no espera a que se desconecten todas las aplicaciones especificadas antes de que sqlefrce concluya la ejecución.

sqlegdad - Catalogar una base de datos en el directorio de DCS (Database Connection Services)

Almacena información relativa a bases de datos remotas del directorio de DCS (Database Connection Services). A estas bases de datos se accede mediante un Peticionario de aplicaciones (AR), por ejemplo DB2 Connect. Al tener una entrada de directorio de DCS con un nombre de base de datos que coincide con un nombre de base de datos del directorio de bases de datos del sistema, se invoca el AR especificado para reenviar las peticiones SQL al servidor remoto donde reside la base de datos.

Autorización

Una de las siguientes:

- sysadm
- sysctrl

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlegdad (
        struct sql_dir_entry * pDCSDirEntry,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlggdad (
        struct sqlca * pSqlca,
        struct sql_dir_entry * pDCSDirEntry);
```

Parámetros de la API sqlegdad

pDCSDirEntry
Entrada. Puntero a una estructura sql_dir_entry (directorio de Database Connection Services).

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

El programa DB2 Connect proporciona conexiones a Servidores de aplicaciones DRDA tales como:

- Bases de datos DB2 para OS/390 en sistemas principales de arquitectura System/370 y System/390
- Bases de datos DB2 para VM y VSE en sistemas principales de arquitectura System/370 y System/390
- Bases de datos OS/400 en sistemas principales Application System/400 (AS/400)

El gestor de bases de datos crea un directorio de DCS (Database Connection Services) si no existe uno. Este directorio se almacena en la vía de acceso que contiene la instancia del gestor de bases de datos que se está utilizando. El directorio de DCS se mantiene fuera de la base de datos.

La base de datos también se debe catalogar como base de datos remota en el directorio de bases de datos del sistema.

Nota: Si la colocación en antememoria de directorios está habilitada, los archivos de directorios de bases de datos, nodos y DCS se almacenarán en la antememoria. La antememoria de directorio de una aplicación se crea durante la primera búsqueda de directorio. Dado que la antememoria sólo se renueva cuando la aplicación modifica alguno de los archivos de directorio, puede que los cambios de directorio efectuados por otras aplicaciones no sean efectivos hasta que se haya reiniciado la aplicación. Para renovar la antememoria compartida de DB2 (servidor solamente), detenga (db2stop) y, a continuación, reinicie (db2start) el gestor de bases de datos. Para renovar la antememoria de directorios para otra aplicación, detenga dicha aplicación y, a continuación, reiníciela.

Sintaxis de la API de REXX

```
CATALOG DCS DATABASE dbname [AS target_dbname]
[AR arname] [PARMS parms] [WITH comment]
```

Parámetros de la API de REXX

dbname
Nombre de la base de datos local de la entrada de directorio que se debe añadir.

target_dbname
Nombre de la base de datos de destino.

arname
Nombre del cliente de aplicaciones.

parms Serie de parámetros. Si se especifica, la serie debe entrecomillarse.

comment
Descripción asociada a la entrada. La longitud máxima es de 30 caracteres. Encierre el comentario entre comillas dobles.

sqlegdcl - Finalizar una exploración del directorio de DCS (Database Connection Services)

Libera los recursos asignados por la API sqlegdsc.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlegdcl (
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlggdcl (
        struct sqlca * pSqlca);
```

Parámetros de la API sqlegdcl

pSqlca
Salida. Puntero a la estructura sqlca.

Sintaxis de la API de REXX

```
CLOSE DCS DIRECTORY
```

sqlegdel - Descatalogar una base de datos del directorio de DCS (Database Connection Services)

Suprime una entrada del directorio de Servicios de conexión a bases de datos (DCS).

Autorización

Una de las siguientes:

- sysadm
- sysctrl

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlegdel (
        struct sql_dir_entry * pDCSDirEntry,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlggdel (
        struct sqlca * pSqlca,
        struct sql_dir_entry * pDCSDirEntry);
```

Parámetros de la API sqlegdel

pDCSDirEntry
Entrada/Salida. Un puntero para la estructura de directorios de los servicios de conexión de base de datos. Rellene el campo ldb de la estructura con el nombre local de la base de datos que se debe suprimir. La entrada del directorio de DCS que coincide con un nombre de base de datos local se copia en esta estructura antes de que se suprima.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Las bases de datos de DCS también se catalogan en el directorio de bases de datos del sistema como bases de datos remotas que se pueden descatalogar mediante la API sqleuncd.

Para volver a catalogar una base de datos en el directorio de DCS, utilice la API sqlegdad.

Para listar las bases de datos de DCS que están catalogadas en un nodo, utilice las API sqlegdsc, sqlegdgt y sqlegdcl.

Si se habilita la puesta de directorios en antememoria (mediante el parámetro de configuración dir_cache), los archivos de directorios de bases de datos, nodos y DCS se almacenan en la antememoria. La antememoria de directorio de una aplicación se crea durante la primera búsqueda de directorio. Dado que la antememoria sólo se renueva cuando la aplicación modifica alguno de los archivos de directorio, puede que los cambios de directorio efectuados por otras aplicaciones no entren en vigor hasta que se haya reiniciado la aplicación. Para renovar la antememoria compartida de DB2 (servidor solamente), detenga (db2stop) y, a continuación, reinicie (db2start) el gestor de bases de datos. Para

renovar la antememoria de directorios para otra aplicación, detenga dicha aplicación y, a continuación, reiníciela.

Sintaxis de la API de REXX

```
UNCATALOG DCS DATABASE dbname [USING :value]
```

Parámetros de la API de REXX

dbname
Nombre de la base de datos local de la entrada de directorio que se debe suprimir.

value Variable compuesta de lenguaje principal de REXX en la que se devuelve información sobre entradas del directorio. En las expresiones que siguen, XXX representa el nombre de la variable de lenguaje principal. Si no se suministra ningún nombre, se utiliza el nombre SQLGWINF.

XXX.0 Número de elementos de la variable (siempre 7)

XXX.1 RELEASE

XXX.2 LDB

XXX.3 TDB

XXX.4 AR

XXX.5 PARMS

XXX.6 COMMENT

XXX.7 RESERVED

---

sqlegdge - Obtener una entrada específica del directorio de DCS (Database Connection Services)

Devuelve información para una entrada específica del directorio de DCS (Database Connection Services).

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlegdge (
        struct sql_dir_entry * pDCSDirEntry,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlggdge (
        struct sqlca * pSqlca,
        struct sql_dir_entry * pDCSDirEntry);
```

Parámetros de la API sqlegdge

pDCSDirEntry
Entrada/Salida. Puntero para la estructura de directorios de DCS (servicios de conexión de base de datos). Llene el campo ldb de esta estructura con el nombre local de la base de datos cuya entrada del directorio de DCS se debe recuperar.

Los campos restantes de la estructura se llenan cuando la API concluye su ejecución.

pSqlca
Salida. Puntero a la estructura sqlca.

Sintaxis de la API de REXX

```
GET DCS DIRECTORY ENTRY FOR DATABASE dbname [USING :value]
```

Parámetros de la API de REXX

dbname
Especifica el nombre de la base de datos local de la entrada de directorio que debe obtenerse.

value Variable compuesta de lenguaje principal de REXX en la que se devuelve información sobre entradas del directorio. En las expresiones que siguen, XXX representa el nombre de la variable de lenguaje principal. Si no se suministra ningún nombre, se utiliza el nombre SQLGWINF.

XXX.0 Número de elementos de la variable (siempre 7)

XXX.1 RELEASE

XXX.2 LDB

XXX.3 TDB

XXX.4 AR

XXX.5 PARMS

XXX.6 COMMENT

XXX.7 RESERVED.

sqlegdgt - Obtener entradas del directorio de DCS (servicios de conexión de base de datos)

Transfiere una copia de entradas del directorio de DCS (Database Connection Services) a un almacenamiento intermedio proporcionado por la aplicación.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlegdgt (
        short * pNumEntries,
        struct sql_dir_entry * pDCSDirEntries,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlggdgt (
        struct sqlca * pSqlca,
        short * pNumEntries,
        struct sql_dir_entry * pDCSDirEntries);
```

Parámetros de la API sqlegdgt

pNumEntries
Entrada/Salida. Puntero para un entero corto que representa el número de entradas que se deben copiar en el almacenamiento intermedio del programa solicitante. Se devuelve el número de entradas que se han copiado realmente.

pDCSDirEntries
Salida. Puntero a un almacenamiento intermedio donde se almacenarán las entradas recopiladas del directorio de DCS cuando finalice la llamada a la API. El almacenamiento intermedio debe ser suficientemente grande para contener el número de entradas especificado en el parámetro pNumEntries.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Se debe invocar la API sqlegdsc, que devuelve el número de entradas, antes de emitir GET DCS DIRECTORY ENTRIES.

Si se copian todas las entradas en el programa solicitante, se cierra automáticamente la exploración del directorio de DCS (Database Connection Services) y se liberan todos los recursos.

Si quedan entradas de directorio, se deben hacer más llamadas a esta API o invocar CLOSE DCS DIRECTORY SCAN para liberar los recursos del sistema.

Sintaxis de la API de REXX

```
GET DCS DIRECTORY ENTRY [USING :value]
```

Parámetros de la API de REXX

value Variable compuesta de lenguaje principal de REXX en la que se devuelve información sobre entradas del directorio. En las expresiones que siguen, XXX representa el nombre de la variable de lenguaje principal. Si no se suministra ningún nombre, se utiliza el nombre SQLGWINF.

XXX.0 Número de elementos de la variable (siempre 7)

XXX.1 RELEASE

XXX.2 LDB

XXX.3 TDB

XXX.4 AR

XXX.5 PARMS

XXX.6 COMMENT

XXX.7 RESERVED

sqlegdcl - Iniciar una exploración del directorio de DCS (Database Connection Services)

Almacena una copia en la memoria de las entradas del directorio de DCS (Database Connection Services) y devuelve el número de entradas. Esta instantánea del directorio en el momento en el que se abre el directorio.

La copia no se actualiza si el directorio cambia después de una llamada a esta API. Utilice la API sqlegdgt y la API sqlegdcl para liberar los recursos asociados a la invocación de esta API.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlegdsc (
        short * pNumEntries,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlggdsc (
        struct sqlca * pSqlca,
        short * pNumEntries);
```

Parámetros de la API sqlegdsc

pNumEntries
Salida. Dirección de un área de 2 bytes donde se devuelve el número de entradas de directorio.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

El que solicita la exploración utiliza el valor devuelto pNumEntries para asignar suficiente memoria para recibir las entradas de directorio. Si se recibe una llamada de exploración mientras ya se tiene una copia, se libera la copia anterior y se recopila una nueva copia.

Sintaxis de la API de REXX

```
OPEN DCS DIRECTORY
```

sqlegins - Obtener instancia actual

Devuelve el valor de la variable de entorno DB2INSTANCE.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlegins (
        _SQLOLDCHAR * pInstance,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlggins (
        struct sqlca * pSqlca,
        _SQLOLDCHAR * pInstance);
```

Parámetros de la API sqlegins

pInstance
Salida. Puntero a un almacenamiento intermedio de series donde se coloca el nombre de la instancia del gestor de bases de datos. Este almacenamiento intermedio debe tener como mínimo 9 bytes de longitud, incluido 1 byte para el carácter de terminación nula.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

El valor de la variable de entorno DB2INSTANCE no es necesariamente la instancia a la que está conectado el usuario.

Para identificar la instancia a la que está conectado actualmente el usuario, invoque `sqleatin - Conectar`, con argumentos nulos excepto para la estructura sqlca.

Sintaxis de la API de REXX

```
GET INSTANCE INTO :instance
```

Parámetros de la API de REXX

instance
Variable de lenguaje principal de REXX en la que se debe colocar el nombre de la instancia del gestor de bases de datos.

sqleintr - Interrumpir peticiones de aplicaciones

Detiene una petición. Se llama a esta API desde un gestor de señales Control-Inter de una aplicación. El gestor de señales Control-Inter puede ser el gestor de señales por omisión, instalado mediante sqleisig - Instalar manejador de señales, o una rutina proporcionada por el programador e instalada mediante una llamada de sistema operativo apropiada.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_INTR
        sqleintr ( void );

SQL_API_RC SQL_API_FN
  sqlgintr (
        void);
```

Parámetros de la API sqleintr

Ninguna

Notas de uso

No se debe invocar ninguna API del gestor de bases de datos desde una gestor de interrupciones excepto sqleintr. Sin embargo, el sistema no impedirá que se realice esa acción.

Las transacciones de base de datos que estén en estado de confirmación o retrotracción no se pueden interrumpir.

Si se interrumpe una petición del gestor de bases de datos, se devuelve un código que indica que se interrumpió la petición.

La tabla siguiente resume el efecto de una operación de interrupción sobre otras API:

Tabla 9. Acciones INTERRUPT

Actividad de base de datos	**Acción**
BACKUP	Programa de utilidad cancelado. Los datos del soporte de almacenamiento pueden estar incompletos.
BIND	Vinculación cancelada. La creación del paquete se ha retrotraído.
COMMIT	Ninguna. COMMIT se completa.

Tabla 9. Acciones INTERRUPT (continuación)

Actividad de base de datos	Acción
CREATE DATABASE/CREATE DATABASE AT NODE/ADD NODE/DROP NODE VERIFY	Después de un punto determinado, estas API no son interrumpibles. Si la llamada de interrupción se recibe antes de este punto, la base de datos no se crea. Si la llamada de interrupción se recibe después de este punto, se pasa por alto.
DROP DATABASE/DROP DATABASE AT NODE	Ninguna. Las API se completan.
EXPORT/IMPORT/ RUNSTATS	Programa de utilidad cancelado. Las actualizaciones de base de datos se retrotraen.
FORCE APPLICATION	Ninguna. FORCE APPLICATION se completa.
LOAD	Programa de utilidad cancelado. Los datos de tabla pueden estar incompletos.
PREP	Precompilación cancelada. La creación del paquete se ha retrotraído.
REORGANIZE TABLE	La interrupción se diferirá hasta que la copia haya finalizado. La recreación de los índices se reanudará en el próximo intento de acceder a la tabla.
RESTORE	Programa de utilidad cancelado. DROP DATABASE se ha realizado. No aplicable a restauraciones a nivel de espacio de tablas.
ROLLBACK	Ninguna. ROLLBACK se completa.
Servicios de directorio	El directorio ha quedado en estado coherente. La función de utilidad puede o no haberse realizado.
Sentencias de definición de datos SQL	Las transacciones de base de datos están establecidas en el estado anterior a la invocación de la sentencia de SQL.
Otras sentencias de SQL	Las transacciones de base de datos están establecidas en el estado anterior a la invocación de la sentencia de SQL.

Sintaxis de la API de REXX

```
INTERRUPT
```

Ejemplos

```
call SQLDBS 'INTERRUPT'
```

sqleisig - Instalar manejador de señales

Instala el manejador de señales de interrupción por omisión (normalmente Control-C y/o Control-Inter. Cuando este gestor por omisión detecta una señal de interrupción, inicializa la señal e invoca sqleintr.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqleisig (
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgisig (
        struct sqlca * pSqlca);
```

Parámetros de la API sqleisig

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Si una aplicación carece de manejador de señales y se recibe una interrupción, se concluye la aplicación. Esta API proporciona un mecanismo simple de gestión de señales y se puede utilizar si una aplicación no tiene unos requisitos importantes para la gestión de interrupciones.

Se debe invocar la API para que el manejador de señales de interrupción funcione debidamente.

Si una aplicación necesita un sistema de gestión de interrupciones más complejo, se puede desarrollar una rutina de manejo de señales para invocar la API sqleintr. Utilice la llamada de sistema operativo o la función de señales de la biblioteca específica del lenguaje. La API sqleintr debe ser la única operación del gestor de bases de datos realizada por un manejador de señales personalizado. Siga todas las técnicas y prácticas de programación del sistema operativo para asegurarse de que los gestores de señales instalados previamente funcionan debidamente.

Sintaxis de la API de REXX

```
INSTALL SIGNAL HANDLER
```

sqlemgdb - Migrar la versión anterior de la base de datos DB2 a la versión actual

Convierte versiones anteriores (Versión 8.x o superior) de bases de datos DB2 a versiones actuales.

Autorización

sysadm

Conexión necesaria

Esta API establece una conexión de base de datos.

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlemgdb (
        _SQLOLDCHAR * pDbAlias,
        _SQLOLDCHAR * pUserName,
        _SQLOLDCHAR * pPassword,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgmgdb (
        unsigned short PasswordLen,
        unsigned short UserNameLen,
        unsigned short DbAliasLen,
        struct sqlca * pSqlca,
        _SQLOLDCHAR * pPassword,
        _SQLOLDCHAR * pUserName,
        _SQLOLDCHAR * pDbAlias);
```

Parámetros de la API sqlemgdb

pDbAlias
Entrada. Serie que contiene el alias de la base de datos que está catalogada en el directorio de bases de datos del sistema.

pUserName
Entrada. Serie que contiene el nombre de usuario de la aplicación. Puede ser NULL.

pPassword
Entrada. Serie que contiene la contraseña del nombre de usuario especificado (si lo hay). Puede ser NULL.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlgmgdb

PasswordLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud en bytes del comentario. El valor se establece en cero si no se proporciona ninguna contraseña.

UserNameLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud en bytes del nombre de usuario. El valor se establece en cero si no se proporciona ningún nombre de usuario.

DbAliasLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud, expresada en bytes, del alias de base de datos.

Notas de uso

Esta API sólo migrará una base de datos a una versión más nueva y no se puede utilizar para convertir una base de datos migrada a su versión anterior.

La base de datos debe catalogarse antes de la migración.

Sintaxis de la API de REXX

```
MIGRATE DATABASE dbalias [USER username USING password]
```

Parámetros de la API de REXX

dbalias
Alias de la base de datos que se debe migrar.

username
Nombre de usuario utilizado para reiniciar la base de datos.

password
Contraseña utilizada para autentificar el nombre de usuario.

sqlencl - Finalizar una exploración del directorio de nodos

Libera los recursos asignados por la API sqlenops.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlencls (
        unsigned short Handle,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgncls (
        unsigned short Handle,
        struct sqlca * pSqlca);
```

Parámetros de la API sqlencls

Handle
Entrada. Identificador devuelto desde la API OPEN NODE DIRECTORY SCAN asociada.

pSqlca
Salida. Puntero a la estructura sqlca.

Sintaxis de la API de REXX

```
CLOSE NODE DIRECTORY :scanid
```

Parámetros de la API de REXX

scanid Variable de lenguaje principal que contiene el identificador de exploración devuelto por la API OPEN NODE DIRECTORY SCAN.

sqlengne - Obtener la entrada siguiente del directorio de nodos

Devuelve la entrada siguiente del nodo de directorios después de invocar sqlenops - Abrir exploración de directorio de nodos. Las llamadas subsiguientes a esta API devuelven entradas adicionales.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlengne (
        unsigned short Handle,
        struct sqleninfo ** ppNodeDirEntry,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgngne (
        unsigned short Handle,
        struct sqleninfo ** ppNodeDirEntry,
        struct sqlca * pSqlca);
```

Parámetros de la API sqlengne

Handle
Entrada. Identificador devuelto por sqlenops - Abrir exploración de directorio de nodos.

ppNodeDirEntry
Salida. Dirección de un puntero a una estructura sqleninfo. El que llama a esta API no tiene que proporcionar memoria para la estructura, tan sólo el puntero. Cuando finaliza la ejecución de la API, el puntero apunta a la entrada siguiente del directorio de nodos en la copia del directorio de nodos asignada por sqlenops - Abrir exploración de directorio de nodos.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Todos los campos del almacenamiento intermedio donde reside la información sobre las entradas del directorio de nodos están rellenados con blancos por la derecha.

El valor de sqlcode en sqlca se establece en 1014 si no hay más entradas por explorar cuando se invoca esta API.

Se puede explorar el directorio completo invocando esta API el número de veces indicado por pNumEntries.

Sintaxis de la API de REXX

```
GET NODE DIRECTORY ENTRY :scanid [USING :value]
```

Parámetros de la API de REXX

scanid Variable de lenguaje principal de REXX que contiene el identificador devuelto por la API Abrir exploración de directorio de nodos.

value Variable compuesta de lenguaje principal de REXX en la que se devuelve información sobre entradas del directorio. Si no se suministra ningún nombre, se utiliza el nombre SQLENINFO. En la información que sigue a continuación, XXX representa el nombre de la variable de lenguaje principal (los nombres de campo correspondientes proceden de la estructura devuelta por la API):

XXX.0 Número de elementos de la variable (siempre 16)

XXX.1 NODENAME

XXX.2 LOCALLU

XXX.3 PARTNERLU

XXX.4 MODE

XXX.5 COMMENT

XXX.6 RESERVED

XXX.7 PROTOCOL (tipo de protocolo)

XXX.9 RESERVED

XXX.10
SYMDESTNAME (nombre de destino simbólico)

XXX.11
SECURITY (tipo de seguridad)

XXX.12
HOSTNAME

XXX.13
SERVICENAME

XXX.14
FILESERVER

XXX.15
OBJECTNAME

XXX.16
INSTANCE (nombre de instancia local).

sqlenops - Iniciar una exploración del directorio de nodos

Almacena una copia en la memoria del directorio de nodos y devuelve el número de entradas. Esta instantánea del directorio en el momento en el que se abre el directorio. Esta copia no se actualiza, aunque el propio directorio se modifique más tarde.

Invoque la API sqlengne para avanzar por el directorio de nodos y examinar información sobre las entradas del nodo. Cierre la exploración invocando la API sqlencls. Esta acción elimina la copia del directorio de la memoria.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlenops (
        unsigned short * pHandle,
        unsigned short * pNumEntries,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgnops (
        unsigned short * pHandle,
        unsigned short * pNumEntries,
        struct sqlca * pSqlca);
```

Parámetros de la API sqlenops

pHandle
Salida. Identificador devuelto por esta API. Este identificador debe pasarse a la API sqlengne y la API sqlencls.

pNumEntries
Salida. Dirección de un área de 2 bytes donde se devuelve el número de entradas de directorio.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

El espacio de almacenamiento asignado por esta API se libera invocando sqlencls - Cerrar exploración de directorio de nodos.

Se pueden ejecutar varias exploraciones para un mismo directorio de nodos, pero los resultados pueden no ser los mismos. El directorio puede cambiar entre una apertura y otra.

Puede haber un máximo de ocho exploraciones de directorio de nodos por proceso.

Sintaxis de la API de REXX

```
OPEN NODE DIRECTORY USING :value
```

Parámetros de la API de REXX

value Variable compuesta de REXX en la que se devuelve información sobre el directorio de nodos. En las expresiones que siguen, XXX representa el nombre de la variable de lenguaje principal.

XXX.0 Número de elementos de la variable (siempre 2)

XXX.1 Especifica una variable de lenguaje principal de REXX que contiene un número para el ID de exploración

XXX.2 Número de entradas contenidas en el directorio.

sqleqryc - Consultar valores de conexión del cliente

Devuelve valores de conexión actuales para un proceso de aplicación. La estructura de datos sqle_conn_setting se llena con los tipos y valores de la conexión.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqleqryc (
        struct sqle_conn_setting * pConnectionSettings,
        unsigned short NumSettings,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgqryc (
        struct sqle_conn_setting * pConnectionSettings,
        unsigned short NumSettings,
        struct sqlca * pSqlca);
```

Parámetros de la API sqleqryc

pConnectionSettings
Entrada/Salida. Un puntero para una estructura sqle_conn_setting, que especifica los tipos y valores de conexión. El usuario define una matriz de estructuras de valores de conexión NumSettings y establece el campo de tipo de cada elemento de la matriz para indicar una de las cinco opciones posibles de valores de conexión. Cuando vuelve, el campo de valor de cada elemento contiene el valor actual de la opción especificada.

NumSettings
Entrada. Cualquier valor entero (comprendido entre 0 y 7) que representa el número de valores que se deben definir para las opciones de conexión.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Los valores de conexión para un proceso de aplicación pueden consultarse en cualquier momento durante la ejecución.

Si QUERY CLIENT se ejecuta satisfactoriamente, los campos de la estructura sqle_conn_setting contendrán los valores de conexión actuales del proceso de aplicación. Si no se ha invocado nunca SET CLIENT, los campos contendrán los

valores de las opciones de precompilación solamente si ya se ha procesado una sentencia de SQL; de lo contrario, los campos contendrán los valores por omisión de las opciones de precompilación.

Sintaxis de la API de REXX

```
QUERY CLIENT INTO :output
```

Parámetros de la API de REXX

output

Variable compuesta de lenguaje principal de REXX que contiene información sobre los valores de conexión actuales del proceso de la aplicación. En las expresiones que siguen, XXX representa el nombre de la variable de lenguaje principal.

XXX.1 Valor de la conexión actual para el tipo de conexión

XXX.2 Valor de la conexión actual para las reglas de SQL

XXX.3 Valor de la conexión actual que indica qué conexiones se liberarán cuando se emita una sentencia COMMIT.

XXX.4 Valor de la conexión actual de la opción SYNCPOINT. La opción SYNCPOINT no se tiene en cuenta y solamente se proporciona con fines de compatibilidad con versiones anteriores. Indica si se debe utilizar un gestor de transacciones para aplicar la semántica de las confirmaciones de dos fases, si el gestor de bases de datos debe asegurarse de que solamente se actualice una única base de datos cuando una transacción individual acceda a varias bases de datos o si no debe utilizarse ninguna de estas dos opciones.

XXX.6 Valor de la conexión actual para la sentencia PREPARE diferida.

sqleqryi - Consultar información sobre el cliente

Devuelve información sobre el cliente cumplimentando los campos de la estructura de datos sqle_client_info. Puesto que esta API permite la especificación de un alias de base de datos, una aplicación puede consultar información sobre el cliente para una conexión determinada. Esta API devuelve un valor nulo si previamente la API sqleseti no ha establecido un valor.

Si se solicita información para una conexión específica, esta API devuelve los valores más recientes para esa conexión. Si se especifican todas las conexiones, la API devuelve los valores correspondientes a todas las conexiones; es decir, los valores pasados en la última llamada a sqleseti (en la que se especifican todas las conexiones).

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqleqryi (
        unsigned short DbAliasLen,
        char * pDbAlias,
        unsigned short NumItems,
        struct sqle_client_info* pClient_Info,
        struct sqlca * pSqlca);
```

Parámetros de la API sqleqryi

DbAliasLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud, expresada en bytes, del alias de base de datos. Si se proporciona un valor mayor que cero, pDbAlias debe apuntar al nombre de alias. Devuelve los valores correspondientes a la última llamada a sqleseti para este alias (o una llamada a sqleseti que especificaba un alias de longitud cero). Si se especifica cero, devuelve los valores correspondientes a la última llamada a sqleseti que especificaba un alias de longitud cero.

pDbAlias
Entrada. Puntero a una serie que contiene el alias de base de datos.

NumItems
Entrada. Número de entradas que se modifican. El valor mínimo es 1.

pClient_Info
Entrada. Puntero a una matriz de estructuras NumItems sqle_client_info, cada una de las cuales contiene un campo de tipo que indica qué valor hay que devolver y un puntero al valor devuelto. El área apuntada debe ser suficientemente grande para albergar el valor que se solicita.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Los valores se pueden consultar en cualquier momento durante la ejecución. Si la llamada a la API es satisfactoria, se devuelven los valores actuales a las áreas especificadas. La API devuelve una longitud igual a cero y una serie con terminación nula (\0) para los campos que no se han definido mediante una llamada a la API sqleseti.

sqlesact - Establecer serie de contabilidad

Proporciona información contable que se enviará a un servidor DRDA con la siguiente petición de conexión de la aplicación.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlesact (
        char * pAccountingString,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgsact (
        unsigned short AccountingStringLen,
        char * pAccountingString,
        struct sqlca * pSqlca);
```

Parámetros de la API sqlesact

pAccountingString
Entrada. Serie que contiene los datos contables.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlgsact

AccountingStringLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud de la serie de contabilidad expresada en bytes.

Notas de uso

Para enviar datos contables con una petición de conexión, una aplicación debe invocar esta API antes de conectar con una base de datos. La serie de contabilidad puede cambiarse antes de conectar con otra base de datos llamando de nuevo a la API; de lo contrario, el valor permanece en vigor hasta el final de la aplicación. La longitud de la serie de contabilidad puede ser como máximo la correspondiente a los bytes de SQL_ACCOUNT_STR_SZ (definido en sqlenv); las series más largas se truncarán. Para asegurar que la serie de contabilidad se convierte correctamente al ser transmitida al servidor DRDA, utilice los caracteres A a la Z, 0 al 9, el carácter de subrayado (_) y el espacio en blanco.

sqlesdeg - Establecer el nivel o grado máximo de paralelismo intrapartición para la ejecución de sentencias de SQL

Establece el grado máximo de tiempo de ejecución de paralelismo intrapartición para la ejecución de sentencias de SQL para aplicaciones activas especificadas. No tiene ningún efecto sobre el paralelismo de ejecución de sentencias CREATE INDEX.

Ámbito

Esta API afecta a todos los servidores de particiones de base de datos que están listados en el archivo db2nodes.cfg.

Autorización

Una de las siguientes:

- sysadm
- sysctrl

Conexión necesaria

Instancia. Para cambiar el grado máximo de tiempo de ejecución de paralelismo en un servidor remoto, primero es necesario conectarse a dicho servidor. Si no existe ninguna conexión, la sentencia SET RUNTIME DEGREE fallará.

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlesdeg (
        sqlint32 NumAgentIds,
        sqluint32 * pAgentIds,
        sqlint32 Degree,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgsdeg (
        struct sqlca * pSqlca,
        sqlint32 Degree,
        sqluint32 * pAgentIds,
        sqlint32 NumAgentIds);
```

Parámetros de la API sqlesdeg

NumAgentIds
Entrada. Número entero que representa el número de total de aplicaciones activas a las que se aplicará el nuevo valor. Este número debe ser igual al número de elementos de la matriz de los ID de agente.

Si el valor de este parámetro se establece en SQL_ALL_USERS (se define en sqlenv), el nuevo grado de paralelismo se aplicará a todas las aplicaciones activas. Si el parámetro se establece en cero, se devuelve un error.

pAgentIds
Entrada. Puntero a una matriz de enteros largos sin signo. Cada entrada describe el ID de agente de la aplicación correspondiente. Para listar los ID de agente de las aplicaciones activas, utilice la API db2GetSnapshot.

Degree
Entrada. El nuevo valor del grado máximo de paralelismo de ejecución. El valor debe estar comprendido entre 1 y 32767.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Se utilizan funciones del supervisor del sistema de base de datos para obtener los ID de agente y los grados de paralelismo de las aplicaciones activas.

Se realiza una validación mínima para la matriz de los ID de agente. El usuario debe asegurarse de que el puntero especifica una matriz que contiene el número total de elementos especificados. Si el valor de NumAgentIds se establece en SQL_ALL_USERS, no se tiene en cuenta la matriz.

Si no se puede encontrar uno o más de los ID de agente especificados, los ID de agente desconocidos no se tienen en cuenta y la función prosigue. No se devuelve

ningún error. Por ejemplo, un ID de usuario puede no encontrarse si el usuario se desconecta entre el momento en el que se obtiene un ID de agente y el momento en el que se invoca la API.

Los ID de agente se reciclan, y se utilizan para cambiar el grado de paralelismo de aplicaciones algún tiempo después de que el supervisor del sistema de base de datos haya obtenido los ID de agente. Por tanto, cuando un usuario se desconecta, otro usuario puede iniciar la sesión y obtener mediante ese proceso de reciclaje el mismo ID de agente, con lo que el nuevo grado de paralelismo se aplicará a un usuario equivocado.

Sintaxis de la API de REXX

Esta API se puede invocar desde REXX mediante la interfaz SQLDB2.

sqlesetc - Establecer valores de conexión del cliente

Especifica valores de conexión para la aplicación. Utilice la estructura de datos sqle_conn_setting para especificar los tipos y valores de las opciones de conexión.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlesetc (
        struct sqle_conn_setting * pConnectionSettings,
        unsigned short NumSettings,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgsetc (
        struct sqle_conn_setting * pConnectionSettings,
        unsigned short NumSettings,
        struct sqlca * pSqlca);
```

Parámetros de la API sqlesetc

pConnectionSettings
Entrada. Puntero a la estructura sqle_conn_setting, que especifica tipos y valores de conexión. Asigne una matriz de estructuras NumSettings sqle_conn_setting. Establezca el campo de tipo de cada elemento de esta matriz para indicar la opción de conexión que hay que establecer. Establezca el campo de valor en el valor deseado para la opción.

NumSettings
Entrada. Cualquier valor entero (comprendido entre 0 y 7) que representa el número de valores que se deben definir para las opciones de conexión.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Si esta API se ejecuta satisfactoriamente, las conexiones de las unidades de trabajo subsiguientes utilizarán los valores de conexión especificados. Si esta API no se ejecuta satisfactoriamente, los valores de conexión permanecen inalterados.

Los valores de conexión de la aplicación solamente se pueden modificar si no hay conexiones existentes (por ejemplo, antes de establecer cualquier conexión o después de emitir RELEASE ALL y COMMIT).

Después de ejecutar satisfactoriamente la API SET CLIENT, se fijan los valores de conexión y solamente se pueden modificar ejecutando de nuevo la API SET CLIENT. Se alteran temporalmente todas las opciones precompiladas correspondientes de los módulos de aplicación.

Sintaxis de la API de REXX

```
SET CLIENT USING :values
```

Parámetros de la API de REXX

values Variable compuesta de lenguaje principal de REXX que contiene los valores de conexión para el proceso de la aplicación. En las expresiones que siguen, XXX representa el nombre de la variable de lenguaje principal.

XXX.0 Número de valores de conexión que se deben definir.

XXX.1 Especifica cómo definir el tipo de conexión. Los valores válidos son:

1 Conexión de tipo 1

2 Conexión de tipo 2

XXX.2 Especifica cómo configurar SQLRULES en función de:

- Si se deben procesar los CONNECT de tipo 2 de acuerdo con las normas de DB2 o las normas Estándares (STD) basadas en ISO/ANS SQL92.
- Cómo una aplicación especifica el formato de columnas LOB en el conjunto de resultados.

DB2

- Permite a la sentencia CONNECT de SQL conmutar la conexión actual a otra conexión (*latente*) establecida.
- Este valor por omisión permite a una aplicación especificar si se recuperan valores de LOB o localizadores de LOB solamente durante la primera petición de captación. Las peticiones de captación subsiguientes deben utilizar el mismo formato para las columnas de LOB.

STD

- Permite a la sentencia CONNECT de SQL establecer solamente una conexión *nueva*. Se deberá utilizar la sentencia SET CONNECTION de SQL para conmutar a una conexión latente.

- La aplicación puede cambiar entre recuperar valores de LOB y localizadores de LOB con cada petición de captación. Esto significa que los cursores con una o más columnas LOB no se pueden bloquear, independientemente del valor de la opción de vinculación BLOCKING.

XXX.3 Especifica cómo definir el ámbito de la desconexión para bases de datos durante la confirmación. Los valores válidos son:

EXPLICIT
Desconectar solamente las marcadas con la sentencia RELEASE de SQL

CONDITIONAL
Desconectar solamente las que no tengan cursores WITH HOLD abiertos

AUTOMATIC
Desconectar todas las conexiones

XXX.4 Especifica cómo definir la coordinación entre varias conexiones de base de datos durante las confirmaciones o retrotracciones. Los valores válidos son:

TWOPHASE
Utilizar el Gestor de transacciones para coordinar las confirmaciones de dos fases. La opción SYNCPOINT no se tiene en cuenta y solamente se proporciona con fines de compatibilidad con versiones anteriores.

XXX.6 Especifica cuándo se debe ejecutar la sentencia PREPARE. Los valores válidos son:

NO La sentencia PREPARE se ejecutará en el momento de emitirse

YES La sentencia PREPARE no se ejecutará hasta que se emita la correspondiente sentencia OPEN, DESCRIBE o EXECUTE. Pero la sentencia PREPARE INTO no se aplaza

ALL Igual que YES, excepto que también se aplaza la ejecución de la sentencia PREPARE INTO

sqleseti - Establecer información sobre el cliente

Permite a una aplicación establecer información sobre el cliente (cumplimentando los campos de la estructura de datos sqle_client_info) correspondiente a una conexión determinada, siempre que ya exista una conexión.

En un entorno de aplicación de supervisor de transacciones (TP) o de cliente/servidor de 3 niveles, es necesario obtener información sobre el cliente, y no solamente sobre el servidor de aplicaciones que trabajo en nombre del cliente. Mediante esta API, el servidor de aplicaciones puede pasar el ID de usuario del cliente, la información de estación de trabajo, la información de programa y otra información de contabilidad al servidor DB2; en otro caso, sólo se pasa la información del servidor de aplicaciones y es probable que esa información sea la misma para muchas invocaciones de clientes que pasan por el mismo servidor de aplicaciones.

La aplicación puede seleccionar no especificar un alias, en cuyo caso la información de cliente se establecerá para todas las conexiones existentes y futuras. Esta API sólo permitirá que se cambie la información fuera de una unidad de trabajo, antes de ejecutar SQL o después de una confirmación o retrotracción. Si la llamada a la API es satisfactoria, los valores para la conexión se enviarán en la oportunidad siguiente, agrupados con la siguiente petición de SQL enviada en esa conexión; una llamada satisfactoria significa que los valores se han aceptado y se propagarán a conexiones subsiguientes.

Esta API se puede utilizar para establecer valores antes conectar con una base de datos, o para definir o modificar los valores una vez establecida una conexión.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqleseti (
        unsigned short DbAliasLen,
        char * pDbAlias,
        unsigned short NumItems,
        struct sqle_client_info* pClient_Info,
        struct sqlca * pSqlca);
```

Parámetros de la API sqleseti

DbAliasLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud, expresada en bytes, del alias de base de datos. Si se proporciona un valor mayor que cero, **pDbAlias** debe apuntar al nombre de alias, y los valores afectarán solamente a la conexión especificada. Si se especifica 0, los valores afectarán a todas las conexiones, existentes y futuras.

pDbAlias
Entrada. Puntero a una serie que contiene el alias de base de datos.

NumItems
Entrada. Número de entradas que se modifican. El valor mínimo es 1.

pClient_Info
Entrada. Puntero a una matriz de estructuras **NumItems** sqle_client_info, cada una de las cuales contiene un campo de tipo que indica qué valor se debe definir, la longitud de dicho valor y un puntero al valor nuevo.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Si se proporciona un nombre de alias, debe existir previamente una conexión con el alias y todas las conexiones con ese alias heredarán los cambios. La información

se conservará hasta que se interrumpa la conexión para ese alias. Si no se proporciona un alias, se cambiarán los valores para todas las conexiones existentes, y todas las conexiones futuras heredarán los cambios. La información se conservará hasta que termine el programa.

Los nombres de campo representan directrices para el tipo de información que se puede proporcionar. Por ejemplo, una aplicación del supervisor de proceso de transacciones (TP) puede proporcionar el ID de transacción del supervisor de proceso de transacciones junto con el nombre de la aplicación en el campo SQL_CLIENT_INFO_APPLNAM. Esto proporcionaría mejores funciones de supervisión y contabilidad en el servidor DB2, donde el ID de transacción de DB2 puede estar asociado con el ID de transacción del supervisor de proceso de transacciones.

Actualmente esta API pasa información DB2 OS/390 Versión 5 y superior, DB2 UDB Versión 7 y superior y a DB2 i5/OS V6R1 y superior. Toda la información (excepto la serie de contabilidad) se visualiza en el mandato DISPLAY THREAD, y se registra en los registros de contabilidad.

También puede accederse a los valores de datos suministrados con la API mediante un registro especial de SQL. Los valores de estos registros se almacenan en la página de códigos de la base de datos. Los valores de datos proporcionados con esta API se convierten a la página de códigos de base de datos antes de almacenarse en los registros especiales. Los valores de datos que sobrepasan el tamaño máximo soportado después de la conversión a la página de códigos de la base de datos se truncarán antes de almacenarse en el servidor. Estos valores truncados serán devueltos por los registros especiales. Los valores de datos originales también se almacenarán en el servidor y no se convertirán a la página de códigos de base de datos. Los valores no convertidos se pueden obtener invocando la API sqleqryi.

La llamada a la API sqleseti en un programa CLI antes de establecer una conexión no funcionará. La llamada a la API sqleseti desde un programa CLI después de establecer una conexión puede provocar un comportamiento imprevisible. Es recomendable utilizar en su lugar las correspondientes funciones de CLI SQLSetConnectAttr() o SQLSetEnvAttr().

sqleuncd - Descatalogar una base de datos del directorio de bases de datos del sistema

Suprime una entrada del directorio de bases de datos del sistema.

Autorización

Una de las siguientes:

- sysadm
- sysctrl

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqleuncd (
        _SQLOLDCHAR * pDbAlias,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlguncd (
        unsigned short DbAliasLen,
        struct sqlca * pSqlca,
        _SQLOLDCHAR * pDbAlias);
```

Parámetros de la API sqleuncd

pDbAlias
Entrada. Serie que contiene el alias de base de datos que se debe descatalogar.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlguncd

DbAliasLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud, expresada en bytes, del alias de base de datos.

Notas de uso

Solamente se pueden descatalogar entradas del directorio de bases de datos del sistema. Las entradas del directorio de bases de datos locales se pueden suprimir utilizando la API sqledrpd.

Para volver a catalogar la base de datos, utilice la API sqlecadb.

Para listar las bases de datos que están catalogadas en un nodo, utilice las API db2DbDirOpenScan, db2DbDirGetNextEntry y db2DbDirCloseScan.

El tipo de autentificación de una base de datos, utilizado al comunicarse con un servidor anterior, se puede cambiar descatalogando primero la base de datos y, a continuación, volviendo a catalogarla con un tipo diferente.

Si se habilita la puesta de directorios en antememoria mediante el parámetro de configuración dir_cache, los archivos de directorios de bases de datos, nodos y DCS se almacenan en la antememoria. La antememoria de directorio de una aplicación se crea durante la primera búsqueda de directorio. Dado que la antememoria sólo se renueva cuando la aplicación modifica alguno de los archivos de directorio, puede que los cambios de directorio efectuados por otras aplicaciones no entren en vigor hasta que se haya reiniciado la aplicación. Para renovar la antememoria compartida de DB2 (servidor solamente), detenga (db2stop) y, a continuación, reinicie (db2start) el gestor de bases de datos. Para renovar la antememoria de directorios para otra aplicación, detenga dicha aplicación y, a continuación, reiníciela.

Sintaxis de la API de REXX

```
UNCATALOG DATABASE dbname
```

Parámetros de la API de REXX

dbname
Alias de la base de datos que se debe descatalogar.

sqleuncn - Descatalogar una entrada del directorio de nodos

Suprime una entrada del directorio de nodos.

Autorización

Una de las siguientes:

- sysadm
- sysctrl

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlenv.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqleuncn (
        _SQLOLDCHAR * pNodeName,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlguncn (
        unsigned short NodeNameLen,
        struct sqlca * pSqlca,
        _SQLOLDCHAR * pNodeName);
```

Parámetros de la API sqleuncn

pNodeName
Entrada. Serie que contiene el nombre del nodo que se debe descatalogar.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlguncn

NodeNameLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud en bytes del nodo.

Notas de uso

Para volver a catalogar el nodo, utilice la API sqlectnd.

Para listar los nodos que están catalogados, utilice las API db2DbDirOpenScan, db2DbDirGetNextEntry y db2DbDirCloseScan.

Si se habilita la puesta de directorios en antememoria mediante el parámetro de configuración dir_cache, los archivos de directorios de bases de datos, nodos y DCS se almacenan en la antememoria. La antememoria de directorio de una

aplicación se crea durante la primera búsqueda de directorio. Dado que la antememoria sólo se renueva cuando la aplicación modifica alguno de los archivos de directorio, puede que los cambios de directorio efectuados por otras aplicaciones no entren en vigor hasta que se haya reiniciado la aplicación. Para renovar la antememoria compartida de DB2 (servidor solamente), detenga (db2stop) y, a continuación, reinicie (db2start) el gestor de bases de datos. Para renovar la antememoria de directorios para otra aplicación, detenga dicha aplicación y, a continuación, reiníciela.

Sintaxis de la API de REXX

```
UNCATALOG NODE nodename
```

Parámetros de la API de REXX

nodename
Nombre del nodo que se debe descatalogar.

sqlgaddr - Obtener la dirección de una variable

Coloca la dirección de una variable dentro de otra variable. Esta API se utiliza en lenguajes de programación de sistema principal, tales como FORTRAN y COBOL, que no proporcionan mecanismos para el manejo de punteros.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlgaddr (
   char * pVariable,
   char ** ppOutputAddress);
```

Parámetros de la API sqlgaddr

pVariable
Entrada. Variable cuya dirección se debe devolver.

ppOutputAddress
Salida. Área de 4 bytes dentro de la que se devuelve la dirección de la variable.

sqlgdref - Eliminar la referencia de una dirección

Copia datos desde un almacenamiento intermedio definido por un puntero a una variable que es directamente accesible por la aplicación. Esta API se utiliza en lenguajes de programación de sistema principal, tales como FORTRAN y COBOL, que no proporcionan mecanismos para el manejo de punteros. Esta API se puede utilizar para obtener resultados desde las API que devuelven un puntero a los datos deseados.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlgdref (
   unsigned int NumBytes,
   char * pTargetBuffer,
   char ** ppSourceBuffer);
```

Parámetros de la API sqlgdref

NumBytes
Entrada. Número entero que representa el número de bytes que se deben transferir.

pTargetBuffer
Salida. Área a la que se trasladan los datos.

ppSourceBuffer
Entrada. Puntero al área donde residen los datos deseados.

sqlgmcpy - Copiar datos de un área de memoria a otra

Copia datos de un área de memoria a otra. Esta API se utiliza en los lenguajes de programación de sistema principal (FORTRAN y COBOL) que no proporcionan funciones de copia de bloques de memoria.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlgmcpy (
   void * pTargetBuffer,
   const void * pSource,
   sqluint32 NumBytes);
```

Parámetros de la API sqlgmcpy

pTargetBuffer
Salida. Salida. Área a la que se trasladan los datos.

pSource
Entrada. Área desde la que se trasladan los datos.

NumBytes
Entrada. Número entero de 4 bytes que representa el número de bytes que se debe transferir.

sqlogstt - Obtener el mensaje de SQLSTATE

Obtiene el texto del mensaje asociado a un valor de SQLSTATE.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sql.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlogstt (
        char * pBuffer,
        short BufferSize,
        short LineWidth,
        char * pSqlstate);

SQL_API_RC SQL_API_FN
  sqlggstt (
        short BufferSize,
        short LineWidth,
        char * pSqlstate,
        char * pBuffer);
```

Parámetros de la API sqlogstt

pBuffer
Salida. Puntero a un almacenamiento intermedio de series donde se coloca el texto del mensaje. Si se debe truncar el mensaje para que quepa en el almacenamiento intermedio, el truncamiento tiene en cuenta el carácter de terminación de serie nula.

BufferSize
Entrada. Tamaño, en bytes, de un almacenamiento intermedio de texto que deberá contener el texto del mensaje recuperado.

LineWidth
Entrada. Ancho máxima de línea correspondiente a cada línea del texto de mensaje. Las líneas se dividen a nivel de palabras. Un valor cero indica que el texto del mensaje se devuelve sin divisiones de línea.

pSqlstate
Entrada. Serie que contiene el SQLSTATE para el que se recupera el texto del mensaje. Este campo es alfanumérico y debe contener cinco dígitos (SQLSTATE específico) o dos dígitos (clase de SQLSTATE, primeros dos dígitos de un SQLSTATE). No es necesario que este campo termine con nulos si se le pasan 5 dígitos, pero debe terminar con nulos si se transfieren 2 dígitos.

Notas de uso

Se devuelve un solo mensaje para cada llamada a la API.

Se coloca una secuencia LF/NULL al final de cada mensaje.

Si se especifica un ancho de línea positivo, se insertan secuencias LF/NULL entre las palabras para que las líneas no sobrepasen el ancho de línea.

Si una palabra es más larga que un ancho de línea, se colocan en la línea tantos caracteres como quepan, se inserta una secuencia LF/NULL y los caracteres restantes se colocan en la línea siguiente.

Códigos de retorno

Código	Mensaje
+i	Número entero positivo que indica el número de bytes del mensaje formateado. Si este valor es mayor que el tamaño de almacenamiento intermedio proporcionado por la aplicación solicitante, se trunca el mensaje.
-1	No hay memoria suficiente disponible para el funcionamiento de los servicios de formateo de mensajes. No se devuelve el mensaje solicitado.
-2	El SQLSTATE tiene un formato incorrecto. Debe ser alfanumérico y tener 2 o 5 dígitos de longitud.
-3	El archivo de mensajes es inaccesible o incorrecto.
-4	El ancho de línea es menor que cero.
-5	sqlca no válida, dirección de almacenamiento intermedio incorrecta o longitud de almacenamiento intermedio incorrecta.

Si el código de retorno es -1 o -3, el almacenamiento intermedio de mensajes contendrá información adicional sobre el problema.

Sintaxis de la API de REXX

```
GET MESSAGE FOR SQLSTATE sqlstate INTO :msg [LINEWIDTH width]
```

Parámetros de la API de REXX

sqlstate
El SQLSTATE para el que se debe obtener el texto del mensaje.

msg Variable de REXX en la que se coloca el mensaje.

width Ancho máximo de cada línea del texto del mensaje. Las líneas se dividen entre palabras. Si no se especifica un valor o este parámetro se establece en 0, se devuelve el texto del mensaje sin divisiones de línea.

sqluadau - Obtener autorizaciones del usuario actual

Notifica las autorizaciones a nivel de instancia y de base de datos del usuario actual a partir de valores encontrados en el archivo de configuración del gestor de bases de datos y la vista de catálogo del sistema de autorización (SYSCAT.DBAUTH) respectivamente. Las autorizaciones a nivel de instancia notificadas son las que se pueden definir en los parámetros de configuración sysadm_group, sysmaint_group y sysctrl_group del gestor de bases de datos; las autorizaciones a nivel de base de datos son las que se pueden otorgar mediante la sentencia GRANT (autorizaciones de base de datos).

Nota: Esta API quedará en desuso y se obtendrá la misma funcionalidad utilizando la función de tabla AUTH_LIST_AUTHORITIES_FOR_AUTHID.

Autorización

Ninguna

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqluadau (
   struct sql_authorizations * pAuthorizations,
   struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgadau (
   struct sql_authorizations * pAuthorizations,
   struct sqlca * pSqlca);
```

Parámetros de la API sqluadau

pAuthorizations
Entrada o salida. Puntero a la estructura sql_authorizations. Esta matriz de enteros cortos indica qué autorizaciones tiene el usuario actual.

El primer elemento de la estructura, sql_authorizations_len, debe tener como valor inicial el tamaño del almacenamiento intermedio que se está transfiriendo, antes de llamar a esta API.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Las autorizaciones directas se obtienen mediante mandatos explícitos que otorgan autorizaciones a un ID de usuario. Las autorizaciones indirectas se basan en autorizaciones adquiridas por los grupos a los que pertenece un usuario.

Nota: PUBLIC es un grupo especial al que pertenecen todos los usuarios.

Si no hay errores, cada elemento de la estructura sql_authorizations contiene un 0 ó un 1. El valor 1 indica que el usuario tiene esa autorización; el valor 0 indica que el usuario no tiene esa autorización.

Sintaxis de la API para REXX

```
GET AUTHORIZATIONS :value
```

Parámetros de la API para REXX

value Variable compuesta de lenguaje principal de REXX en la que se devuelve información sobre el nivel de autorización. En las expresiones que siguen, XXX representa el nombre de la variable de lenguaje principal. Los valores son 0 que indica no y 1 que indica sí.

XXX.0 Número de elementos de la variable (siempre 18)

XXX.1 Autorización SYSADM directa

XXX.2 Autorización DBADM directa

XXX.3 Autorización CREATETAB directa

XXX.4 Autorización BINDADD directa

XXX.5 Autorización CONNECT directa

XXX.6 Autorización SYSADM indirecta

XXX.7 Autorización DBADM indirecta

XXX.8 Autorización CREATETAB indirecta

XXX.9 Autorización BINDADD indirecta

XXX.10
Autorización CONNECT indirecta

XXX.11
Autorización SYSCTRL directa

XXX.12
Autorización SYSCTRL indirecta

XXX.13
Autorización SYSMAINT directa

XXX.14
Autorización SYSMAINT indirecta

XXX.15
Autorización CREATE_NOT_FENC directa

XXX.16
Autorización CREATE_NOT_FENC indirecta

XXX.17
Autorización IMPLICIT_SCHEMA directa

XXX.18
Autorización IMPLICIT_SCHEMA indirecta.

XXX.19
Autorización LOAD directa.

XXX.20
Autorización LOAD indirecta.

sqludrdt: redistribuir datos a través de un grupo de particiones de base de datos

Redistribuye datos en las particiones de base de datos de un grupo de particiones de base de datos. Puede especificar la distribución actual de datos, ya sea uniforme o desviada. El algoritmo de redistribución selecciona las particiones de la base de datos que se van a mover en función de la distribución actual de datos. Esta API no admite la opción NOT ROLLFORWARD RECOVERABLE del mandato REDISTRIBUTE DATABASE PARTITION GROUP.

Esta API solo puede llamarse desde la partición del catálogo. Utilice el mandato LIST DATABASE DIRECTORY para determinar qué servidor de particiones de base de datos es la partición de catálogo de cada base de datos.

Ámbito

Esta API afecta a todas las particiones de la base de datos del grupo de particiones de base de datos.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- dbadm

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqludrdt (
   char * pNodeGroupName,
   char * pTargetPMapFileName,
   char * pDataDistFileName,
   SQL_PDB_NODE_TYPE * pAddList,
   unsigned short AddCount,
   SQL_PDB_NODE_TYPE * pDropList,
   unsigned short DropCount,
   unsigned char DataRedistOption,
   struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdrdt (
   unsigned short NodeGroupNameLen,
   unsigned short TargetPMapFileNameLen,
   unsigned short DataDistFileNameLen,
   char * pNodeGroupName,
   char * pTargetPMapFileName,
   char * pDataDistFileName,
   SQL_PDB_NODE_TYPE * pAddList,
   unsigned short AddCount,
```

```
SQL_PDB_NODE_TYPE * pDropList,
unsigned short DropCount,
unsigned char DataRedistOption,
struct sqlca * pSqlca);
```

Parámetros de la API sqludrdt

pNodeGroupName
Nombre del grupo de particiones de base de datos que se va a redistribuir.

pTargetPMapFileName
Nombre del archivo que contiene el mapa de distribución de destino. Si no se especifica una vía de acceso al directorio como parte del nombre de archivo, se utilizará el directorio actual. Este parámetro se utiliza cuando el valor de DataRedistOption es T. El archivo debe estar en formato de caracteres y contener o bien 4.096 entradas (para un grupo de particiones de base de datos de varias particiones) o bien 1 entrada (para un grupo de particiones de base de datos de una sola partición). Las entradas en el archivo indican números de nodo. Las entradas pueden estar en el formato que desee.

pDataDistFileName
Nombre del archivo que contiene la información de distribución de entrada. Si no se especifica una vía de acceso al directorio como parte del nombre de archivo, se utilizará el directorio actual. Este parámetro se utiliza cuando el valor de DataRedistOption es U. El archivo debe estar en formato de caracteres y contener 4.096 entradas de enteros positivos. Cada entrada en el archivo debe indicar el grosor de la partición de la base de datos correspondiente. La suma de los 4.096 valores debe ser menor que, o igual a, 4.294.967.295.

pAddList
Lista de particiones de base de datos para añadir al grupo de particiones de base de datos durante la redistribución de datos. Las entradas en la lista deben estar en formato: SQL_PDB_NODE_TYPE.

AddCount
Número de particiones de base de datos para añadir al grupo de particiones de base de datos.

pDropList
Lista de particiones de base de datos para descartar del grupo de particiones de base de datos durante la redistribución de datos. Las entradas en la lista deben estar en formato: SQL_PDB_NODE_TYPE.

DropCount
Número de particiones de base de datos para descartar del grupo de particiones de base de datos.

DataRedistOption
Un único carácter que indica el tipo de redistribución de datos que debe realizarse. Los valores posibles son:

U Especifica la redistribución del grupo de particiones de base de datos para alcanzar una distribución equilibrada. Si pDataDistFileName es nulo, se da por hecho que la distribución actual de los datos es uniforme (es decir, cada partición de la base de datos representa la misma cantidad de datos). Si el parámetro pDataDistFileName no es nulo, se da por hecho que los valores en este archivo representan la distribución actual de los datos. Si el valor de DataRedistOption es U, el parámetro

pTargetPMapFileName debería ser nulo. Se añaden las particiones de la base de datos especificadas en la lista de adiciones, y se descartan del grupo de particiones de base de datos las particiones de base de datos especificadas en la lista de descartes.

T Especifica la redistribución del grupo de particiones de base de datos utilizando el parámetro pTargetPMapFileName. Para esta opción, los parámetros, pDataDistFileName, pAddList y pDropList deben ser nulos, y los parámetros AddCount y DropCount deben ser cero.

C Especifica que continúe una operación de redistribución que ha fallado. Para esta opción, los parámetros, pTargetPMapFileName, pDataDistFileName, pAddList y pDropList deben ser nulos, y los parámetros AddCount y DropCount deben ser cero.

R Especifica que se retrotraiga una operación de redistribución que ha fallado. Para esta opción, los parámetros, pTargetPMapFileName, pDataDistFileName, pAddList y pDropList deben ser nulos, y los parámetros AddCount y DropCount deben ser cero.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlgdrdt

NodeGroupNameLen
Longitud del nombre del grupo de particiones de base de datos.

TargetPMapFileNameLen
Longitud del nombre del archivo del mapa de distribución de destino.

DataDistFileNameLen
Longitud del nombre del archivo de distribución de datos.

Notas de uso

Cuando finaliza una operación de redistribución, se escribe un archivo de mensaje en:

- El directorio $HOME/sqllib/redist en los sistemas basados en UNIX, utilizando el formato siguiente para los subdirectorios y el nombre de archivo: nombre_basedatos.nombre_gruponodos.indicación_hora.
- El directorio $HOME\sqllib\redist\ en el sistema operativo Windows, utilizando el formato siguiente para los subdirectorios y nombre de archivo: nombre_basedatos\primeros_ocho_caracteres_del_nombre_grupo_nodo\fecha\hora.

El valor de indicación de fecha y hora a la que se llamó la API.

Este programa de utilidad realiza operaciones COMMIT intermitentes durante el proceso.

Utilice la sentencia ALTER DATABASE PARTITION GROUP para añadir particiones de base de datos a un grupo de particiones de base de datos. Esta sentencia permite definir los contenedores para los espacios de tablas asociados con el grupo de particiones de base de datos.

Todos los paquetes que tienen una dependencia en una tabla que ha sido sometida a una redistribución se invalidan. Se recomienda volver a vincular explícitamente dichos paquetes después de que se haya completado la operación de redistribución de grupo de particiones de base de datos. La revinculación explícita elimina el retardo inicial en la ejecución de la primera petición SQL para el paquete no válido. El archivo de mensajes de redistribución contiene una lista de todas las tablas que han sido sometidas a redistribución.

También se recomienda actualizar las estadísticas emitiendo la API db2Runstats después de que se haya completado la redistribución del grupo de particiones de base de datos.

Los grupos de particiones de base de datos que contienen tablas de resumen duplicadas o tablas definidas con la cláusula DATA CAPTURE CHANGES no pueden redistribuirse.

La redistribución no está permitida si existen espacios de tablas temporales de usuarios con tablas temporales declaradas existentes en el grupo de particiones de base de datos.

Sintaxis de la API de REXX

Esta API se puede invocar desde REXX mediante la interfaz SQLDB2.

sqlugrpn - Obtener el número de servidor de particiones de base de datos para una fila

Devuelve el número de partición de base de datos y el número de servidor de particiones de base de datos de acuerdo con los valores de claves de distribución. Una aplicación puede utilizar esta información para determinar el servidor de particiones de base de datos donde está almacenada una determinada fila de una tabla.

La estructura de datos de particionamiento, sqlupi, es la entrada para esta API. La estructura puede ser devuelta por la API sqlugtpi. También se utiliza como entrada la representación de tipo carácter de los correspondientes valores de clave de distribución. La salida de la API es un número de partición de base de datos generado por la estrategia de distribución y el correspondiente número de servidor de particiones de base de datos procedente del mapa de distribución. Si no se proporciona la información sobre el mapa de distribución, solamente se devuelve el número de partición de base de datos. Esto puede ser útil al analizar la distribución de los datos.

No es necesario que el gestor de bases de datos esté en ejecución cuando se invoca esta API.

Ámbito

Esta API se debe invocar desde un servidor de particiones de base de datos definido en el archivo db2nodes.cfg. Esta API no se debe invocar desde un cliente, pues se podría devolver información errónea sobre el particionamiento de la base de datos, debido a diferencias en la página de códigos y en la ordenación de los bytes entre el cliente y el servidor.

Autorización

Ninguna

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlugrpn (
   unsigned short num_ptrs,
   unsigned char ** ptr_array,
   unsigned short * ptr_lens,
   unsigned short territory_ctrycode,
   unsigned short codepage,
   struct sqlupi * part_info,
   short * part_num,
   SQL_PDB_NODE_TYPE * node_num,
   unsigned short chklvl,
   struct sqlca * sqlca,
   short dataformat,
   void * pReserved1,
   void * pReserved2);

SQL_API_RC SQL_API_FN
  sqlggrpn (
   unsigned short num_ptrs,
   unsigned char ** ptr_array,
   unsigned short * ptr_lens,
   unsigned short territory_code,
   unsigned short codepage,
   struct sqlupi * part_info,
   short * part_num,
   SQL_PDB_NODE_TYPE * node_num,
   unsigned short chklvl,
   struct sqlca * sqlca,
   short dataformat,
   void * pReserved1,
   void * pReserved2);
```

Parámetros de la API sqlugrpn

num_ptrs
Número de punteros existentes en ptr_array. El valor debe ser el mismo que el especificado para el parámetro part_info; es decir, part_info->sqld.

ptr_array
Matriz de punteros que apunta a las representaciones de tipo carácter de los valores correspondientes de cada parte de la clave de distribución especificada en part_info. Si es necesario un valor nulo, el puntero correspondiente se establece en nulo. Para las columnas generadas, esta función no genera valores para la fila. El usuario debe proporcionar un valor que produzca el particionamiento correcto de la fila.

ptr_lens
Matriz de enteros sin signo que contiene las longitudes de las representaciones de tipo carácter de los valores correspondientes de cada parte de la clave de particionamiento especificada en part_info.

territory_ctrycode
Código de país/región de la base de datos de destino. Este valor también se puede obtener a partir del archivo de configuración de la base de datos utilizando el mandato GET DATABASE CONFIGURATION.

codepage
Página de códigos de la base de datos de destino. Este valor también se puede obtener a partir del archivo de configuración de la base de datos utilizando el mandato GET DATABASE CONFIGURATION.

part_info
Puntero a la estructura sqlupi.

part_num
Puntero a un número entero sin signo, de 2 bytes, que se utiliza para almacenar el número de partición de base de datos.

node_num
Puntero a un campo SQL_PDB_NODE_TYPE que se utiliza para almacenar el número de nodo. Si el puntero es nulo, no se devuelve ningún número de nodo.

chklvl Número entero sin signo que especifica el nivel de la comprobación realizada sobre los parámetros de entrada. Si el valor especificado es cero, no se realiza ninguna comprobación. Si se especifica un valor cualquiera distinto de cero, se comprueban todos los parámetros de entrada.

sqlca Salida. Puntero a la estructura sqlca.

dataformat
Especifica la representación de los valores de clave de distribución. Los valores válidos son:

SQL_CHARSTRING_FORMAT
Todos los valores de clave de distribución se representan mediante series de caracteres. Es el valor por omisión.

SQL_IMPLIEDDECIMAL_FORMAT
La ubicación de una coma decimal implícita se determina mediante la definición de la columna. Por ejemplo, si la definición de la columna es DECIMAL(8,2), el valor 12345 se procesa como 123,45.

SQL_PACKEDDECIMAL_FORMAT
Todos los valores de clave de distribución de las columnas decimales están en formato decimal empaquetado.

SQL_BINARYNUMERICS_FORMAT
Todos los valores numéricos de clave de distribución están en formato binario big-endian (byte más significativo primero).

pReserved1
Reservado para una utilización futura.

pReserved2
Reservado para una utilización futura.

Notas de uso

Los tipos de datos que se pueden utilizar en el sistema operativo son los mismos que los que se pueden definir como clave de distribución.

Nota: Los tipos de datos CHAR, VARCHAR, GRAPHIC y VARGRAPHIC se deben convertir a la página de códigos de la base de datos antes de invocar a esta API.

Para los tipos de datos NUMERIC y DATETIME, las representaciones de tipo carácter deben utilizar la página de códigos del sistema respectivo desde donde se invoca la API.

Si node_num no es nulo, se debe proporcionar el mapa de distribución; es decir, el campo pmaplen del parámetro part_info (part_info->pmaplen) es 2 o 8192. De lo contrario, se devuelve el SQLCODE -6038. Se debe definir la clave de distribución; es decir, el campo sqld del parámetro part_info (part_info->sqld) debe ser mayor que cero. De lo contrario, se devuelve el SQLCODE -2032.

Si se asigna un valor nulo a una columna de particionamiento que no puede contener nulos, se devuelve el SQLCODE -6039.

Los espacios en blanco iniciales y finales se eliminan de la serie de entrada, excepto para los tipos de datos CHAR, VARCHAR, GRAPHIC y VARGRAPHIC, donde solamente se eliminan los blancos de cola.

qlugtpi - Obtener información de distribución de tablas

Permite a una aplicación obtener la información de distribución para una tabla. La información de distribución incluye la correlación de distribución y las definiciones de columna de la clave de distribución. La información devuelta por esta API se puede pasar a la API sqlugrpn para determinar el número de partición de base de datos y el número de servidor de particiones de base de datos para cualquier fila de la tabla.

Para utilizar esta API, la aplicación debe estar conectada a la base de datos donde reside la tabla para la cual se solicita información de distribución.

Ámbito

Esta API se puede ejecutar en cualquier servidor de particiones de base de datos definido en el archivo db2nodes.cfg.

Autorización

Para la tabla especificada, el usuario debe tener como mínimo una de las autorizaciones o privilegios siguientes:

- Autorización sysadm
- Autorización dbadm
- Privilegio CONTROL
- Privilegio SELECT

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqlugtpi (
   unsigned char * tablename,
   struct sqlupi * part_info,
   struct sqlca * sqlca);

SQL_API_RC SQL_API_FN
  sqlggtpi (
```

```
unsigned short tn_length,
unsigned char * tablename,
struct sqlupi * part_info,
struct sqlca * sqlca);
```

Parámetros de la API sqlugtpi

nombretabla
Nombre totalmente calificado de la tabla.

part_info
Puntero a la estructura sqlupi.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlggtpi

tn_length
Número entero sin signo, de 2 bytes, que contiene la longitud del nombre de tabla.

sqluvqdp - Inmovilizar espacios de tablas para una tabla

Inmoviliza los espacios de tablas para una tabla. Existen tres modalidades de inmovilización válidas: compartimiento, intención de actualización y exclusiva. Como resultado de la función de inmovilización, existen tres estados posibles para un espacio de tablas:

- Inmovilizado: SHARE
- Inmovilizado: UPDATE
- Inmovilizado: EXCLUSIVE

Ámbito

En un entorno de bases de datos de una sola partición, esta API inmoviliza todos los espacios de tablas implicados en una operación de carga en modalidad exclusiva durante el tiempo que dura la carga. En un entorno de bases de datos particionadas, esta API actúa localmente en una partición de base de datos. Sólo inmoviliza la parte de espacios de tablas que pertenecen a la partición de base de datos en la que se realiza la carga.

Autorización

Una de las siguientes:

- sysadm
- sysctrl
- sysmaint
- dbadm
- load

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlutil.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqluvqdp (
   char * pTableName,
   sqlint32 QuiesceMode,
   void * pReserved,
   struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgvqdp (
   unsigned short TableNameLen,
   char * pTableName,
   sqlint32 QuiesceMode,
   void * pReserved,
   struct sqlca * pSqlca);
```

Parámetros de la API sqluvqdp

pTableName

Entrada. Una serie que contiene el nombre de tabla correspondiente al utilizado en el catálogo del sistema. Este nombre puede estar formado por el nombre de esquema y el nombre de tabla separados por un punto (.).. Si no se proporciona el esquema, se utiliza el esquema actual (CURRENT SCHEMA).

La tabla no puede ser una tabla de catálogos del sistema. Este campo es obligatorio.

QuiesceMode

Entrada. Especifica la modalidad de inmovilización. Los valores válidos (definidos en sqlutil) son:

SQLU_QUIESCEMODE_SHARE
Para la modalidad de compartimiento

SQLU_QUIESCEMODE_INTENT_UPDATE
Para la modalidad de intento de actualización

SQLU_QUIESCEMODE_EXCLUSIVE
Para la modalidad exclusiva

SQLU_QUIESCEMODE_RESET
Restaura el estado de los espacios de tablas a normal si se cumple alguna de estas dos condiciones:

- El que realiza la llamada es el propietario de la inmovilización
- El que realiza la llamada que establece la inmovilización se desconecta, creando una ″inmovilización fantasma″.

SQLU_QUIESCEMODE_RESET_OWNED
Restaura el estado de los espacios de tablas a normal si el que realiza la llamada es el propietario de la inmovilización.

Este campo es obligatorio.

pReserved

Reservado para una utilización futura.

pSqlca

Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlgvqdp

TableNameLen
Entrada. Número entero sin signo, de 2 bytes, que representa la longitud en bytes del nombre de tabla.

Notas de uso

Esta API no se puede utilizar para tablas temporales declaradas.

Cuando se recibe una petición de compartimiento de inmovilización, la transacción solicita intentar bloqueos de compartimiento para los espacios de tablas y un bloqueo de compartimiento para la tabla. Cuando la transacción obtiene los bloqueos, el estado de los espacios de tablas cambia a QUIESCED SHARE. El estado sólo se otorga al inmovilizador si no hay ningún estado en conflicto mantenido por otros usuarios. El estado de los espacios de tablas, junto con el ID de autorización y el ID de agente de base de datos del inmovilizador, se registran en la tabla de espacios de tablas, para que el estado sea permanente.

No se puede cambiar la tabla mientras los espacios de tablas para la tabla están en estado QUIESCED SHARE. Se permitirán otras peticiones de modalidad de compartimiento en la tabla y los espacios de tabla. Cuando la transacción se confirma o se retrotrae, se liberan los bloqueos, pero los espacios de tablas para la tabla permanecen en estado QUIESCED SHARE hasta que se restablece explícitamente el estado.

Cuando se realiza una petición de inmovilización exclusiva, la transacción solicita bloqueos superexclusivos para los espacios de tablas y un bloqueo superexclusivo para la tabla. Cuando la transacción obtiene los bloqueos, el estado de los espacios de tablas cambia a QUIESCED EXCLUSIVE. El estado de los espacios de tablas, junto con el ID de autorización y el ID de agente de base de datos del inmovilizador, se registran en la tabla de espacios de tablas. Dado que los espacios de tablas se mantienen en modalidad superexclusiva, no se permite ningún otro acceso a los espacios de tablas. Sin embargo, el usuario que invoca la función de inmovilización (inmovilizador) tiene acceso exclusivo a la tabla y a los espacios de tablas.

Cuando se realiza una petición de actualización de inmovilización, los espacios de tablas se bloquean en la modalidad de intento exclusivo (IX) y la tabla se bloquea en la modalidad de actualización (U). El estado de los espacios de tablas, con el inmovilizador, se registra en la tabla de espacios de tablas.

Existe un límite de cinco inmovilizadores en un espacio de tablas en un momento determinado. Dado que QUIESCED EXCLUSIVE es incompatible con cualquier otro estado, y QUIESCED UPDATE es incompatible con otro QUIESCED UPDATE, si se alcanza el límite de cinco inmovilizadores, éstos deben tener como mínimo cuatro estados QUIESCED SHARE y como máximo un estado QUIESCED UPDATE.

Un inmovilizador puede actualizar el estado de un espacio de tablas de un estado menos restrictivo a otro más restrictivo (por ejemplo, S a U o U a X). Si un usuario solicita un estado más bajo que uno que ya se mantiene, se vuelve al estado original. Los estados no pueden disminuir de nivel.

El estado de inmovilización de un espacio de tablas se debe restaurar explícitamente utilizando SQLU_QUIESCEMODE_RESET.

Sintaxis de la API de REXX

```
QUIESCE TABLESPACES FOR TABLE table_name
{SHARE | INTENT TO UPDATE | EXCLUSIVE | RESET}
```

Parámetros de la API de REXX

table_name

Nombre de la tabla que se utiliza en el catálogo del sistema. Este nombre puede estar formado por el nombre de esquema y el nombre de tabla separados por un punto (.).. Si no se proporciona el esquema, se utiliza el esquema actual (CURRENT SCHEMA).

Capítulo 5. Llamada a las API de DB2 en REXX

Utilice la rutina SQLDBS para llamar a las API de DB2 con la sintaxis siguiente:

```
CALL SQLDBS 'command string'
```

Si no puede llamar a una API de DB2 que desea utilizar mediante la rutina SQLDBS, puede llamar a la API efectuando una llamada al procesador de línea de mandatos (CLP) de DB2 desde dentro de la aplicación REXX. Sin embargo, puesto que el CLP de DB2 dirige la salida al dispositivo de salida estándar o a un archivo especificado, la aplicación REXX no puede acceder directamente a la salida de la API de DB2 a la que se ha llamado, ni puede tomar fácilmente la determinación de si la API llamada ha resultado satisfactoria o no. La API SQLDB2 proporciona una interfaz al CLP de DB2 que proporciona información directa a la aplicación REXX sobre el éxito o fracaso de cada API llamada, estableciendo la variable compuesta SQLCA de REXX después de cada llamada.

Puede utilizar la rutina SQLDB2 para llamar a las API de DB2 utilizando la sintaxis siguiente:

```
CALL SQLDB2 'command string'
```

donde `'command string'` es una serie que puede procesar el procesador de línea de mandatos (CLP).

El hecho de llamar a una API de DB2 mediante SQLDB2 es equivalente a llamar directamente al CLP, excepto en los aspectos siguientes:

- La llamada al ejecutable del CLP se sustituye por la llamada a SQLDB2 (el resto de opciones y parámetros del CLP se especifican de la misma manera).
- La variable compuesta SQLCA de REXX se establece después de llamar a SQLDB2, pero no se establece después de llamar al ejecutable del CLP.
- La salida de visualización por omisión del CLP se establece como desactivada cuando se llama a SQLDB2, mientras que se establece como salida activada cuando se llama al ejecutable del CLP. Observe que puede activar la salida de visualización del CLP pasando las opciones `+o` o `-o-` a SQLDB2.

Puesto que la única variable de REXX que se establece después de llamar a SQLDB2 es la SQLCA, sólo debe utilizar esta rutina para llamar a las API de DB2 que no devuelvan datos distintos de la SQLCA y que no estén implementadas actualmente mediante la interfaz SQLDBS. Así pues, SQLDB2 únicamente soporta las siguientes API de DB2:

- Activar base de datos
- Añadir nodo
- Vincular para DB2 Versión 1[(1) (2)]
- Vincular para DB2 Versión 2 ó 5[(1)]
- Crear base de datos en nodo
- Descartar base de datos en nodo
- Verificar descarte de nodo
- Desactivar base de datos
- Desregistrar
- Cargar[(3)]
- Cargar consulta
- Precompilar programa[(1)]
- Revincular paquete[(1)]

- Redistribuir grupo de particiones de base de datos
- Registrar
- Iniciar gestor de bases de datos
- Detener gestor de bases de datos

Notas sobre las API de DB2 soportadas por SQLDB2:

1. Estos mandatos requieren una sentencia CONNECT a través de la interfaz SQLDB2. Las conexiones que utilizan la interfaz SQLDB2 no están accesibles para la interfaz SQLEXEC, y las conexiones que utilizan la interfaz SQLEXEC no están accesible para la interfaz SQLDB2.
2. Está soportado en las plataformas basadas en Windows mediante la interfaz SQLDB2.
3. El parámetro de salida opcional, `poLoadInfoOut` para la API Cargar no se devuelve a la aplicación en REXX.

Nota: Aunque la rutina SQLDB2 ha sido pensada para uso únicamente de las API de DB2 relacionadas anteriormente, también se puede utilizar para otras API de DB2 que no se soportan a través de la rutina SQLDBS. Alternativamente, se puede acceder a las API de DB2 a través del CLP desde dentro de la aplicación REXX.

Cambiar el nivel de aislamiento

Cambia el modo en que DB2 aísla los datos de otros procesos mientras se está accediendo a una base de datos. Esta API solamente se puede invocar desde una aplicación REXX.

Autorización

Ninguna

Conexión necesaria

Ninguna

Sintaxis de la API de REXX

```
CHANGE SQLISL TO {RR|CS|UR|RS|NC}
```

Parámetros de la API de REXX

RR Lectura repetible.

CS Estabilidad del cursor. Es el valor por omisión.

UR Lectura no confirmada.

RS Estabilidad de lectura.

NC Sin confirmación.

Capítulo 6. Las API de gestión de transacciones dudosas

Las bases de datos se pueden utilizar en un entorno de proceso de transacciones distribuidas (DTP).

Se proporciona un conjunto de API para que los que escriben herramientas puedan realizar funciones heurísticas en transacciones dudosas cuando el propietario del recurso, por ejemplo, el administrador de base de datos, no puede esperar a que el gestor de transacciones (TM) efectúe la acción *re-sync*. Esta condición puede producirse si, por ejemplo, se interrumpe la línea de comunicaciones y una transacción dudosa está ocupando recursos necesarios. Para el gestor de bases de datos, estos recursos son bloqueos de tablas e índices, espacio de registro y almacenamiento que utiliza la transacción. Cada transacción dudosa disminuye también, en uno, el número máximo de transacciones simultáneas que puede procesar el gestor de bases de datos.

Las API heurísticas pueden consultar, confirmar y retrotraer las transacciones dudosas y cancelar las transacciones que se han confirmado o retrotraído de forma heurística, suprimiendo los registros y liberando las páginas del registro.

Atención: Las API heurísticas se deben utilizar con precaución y únicamente como último registro. El TM debe dirigir los sucesos de resincronización. Si el TM tiene un mandato de operador para iniciar la acción de resincronización, debe utilizarlo. Si el usuario no puede esperar a una resincronización iniciada por el TM, es necesario realizar las acciones heurísticas.

Aunque no hay un modo establecido de realizar estas acciones, las siguientes directrices pueden resultar útiles:

- Utilice la función db2XaListIndTrans para visualizar las transacciones dudosas. Su estado es 'P' (preparadas) y no están conectadas. La parte *gtrid* de un *xid* es el ID de transacción global que es idéntico en los otros gestores de recursos (RM) que participan en la transacción global.
- Utilice sus conocimientos sobre la aplicación y el entorno operativo para identificar a los otros RM participantes.
- Si el gestor de transacciones es CICS y el único RM es un recurso CICS, efectúe una retrotracción heurística.
- Si el gestor de transacciones no es CICS, utilícelo para determinar el estado de la transacción que tiene el mismo *gtrid* que la transacción dudosa.
- Si se ha confirmado o retrotraído al menos un RM, efectúe una operación de confirmación o retrotracción heurística.
- Si están todas en estado preparado, efectúe una retrotracción heurística.
- Si al menos un RM no está disponible, efectúe una retrotracción heurística.

Si el gestor de transacciones está disponible y la transacción dudosa es el resultado de que el RM no está disponible en la segunda fase o de una resincronización anterior, el DBA debe determinar a partir del registro del TM la acción que se ha llevado a cabo en los otros RM y, a continuación, realizar la misma acción. El *gtrid* es la clave coincidente entre el TM y los RM.

No ejecute sqlxhfrg a menos que al retrotraer o confirmar una transacción se produzca una condición de registro lleno. La función forget de la modalidad

heurística libera el espacio del registro que ocupa una transacción dudosa. Si eventualmente un gestor de transacciones realiza una acción de resincronización para esta transacción dudosa, el TM puede cometer la decisión equivocada de confirmar o retrotraer los otros RM, debido a que no se ha encontrado un registro en este RM. En general, si falta un registro significa que el RM se ha retrotraído.

db2XaGetInfo - Obtener información para un gestor de recursos

Extrae información para un gestor de recursos determinado una vez realizada una llamada a xa_open.

Autorización

Instancia - Conexión con nombre de SPM

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlxa.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2XaGetInfo(db2Uint32 versionNumber,
               void * pParmStruct,
               struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2XaGetInfoStruct
{
   db2int32 iRmid;
   struct sqlca         oLastSqlca;
} db2XaGetInfoStruct;
```

Parámetros de la API db2XaGetInfo

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2XaGetInfoStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2XaGetInfoStruct

iRmid
Entrada. Especifica el gestor de recursos para el que se solicita información.

oLastSqlca
Salida. Contiene la sqlca de la última llamada a la API de XA.

Nota: Solamente se puede recuperar la sqlca de la última API de XA anómala.

db2XaListIndTrans - Listar transacciones dudosas

Proporciona una lista de todas las transacciones dudosas para la base de datos conectada actualmente.

Ámbito

Esta API sólo afecta a la partición de base de datos en la que se emite.

Autorización

Ninguna

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlxa.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  db2XaListIndTrans (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2XaListIndTransStruct
{
   db2XaRecoverStruct * piIndoubtData;
   db2Uint32            iIndoubtDataLen;
   db2Uint32            oNumIndoubtsReturned;
   db2Uint32            oNumIndoubtsTotal;
   db2Uint32            oReqBufferLen;
} db2XaListIndTransStruct;

typedef SQL_STRUCTURE db2XaRecoverStruct{
   sqluint32 timestamp;
   SQLXA_XID    xid;
   char dbalias[SQLXA_DBNAME_SZ];
   char applid[SQLXA_APPLID_SZ];
   char sequence_no[SQLXA_SEQ_SZ];
   char auth_id[SQLXA_USERID_SZ];
   char log_full;
   char connected;
   char indoubt_status;
   char originator;
   char reserved[8];
   sqluint32 rmn;
   rm_entry rm_list[SQLXA_MAX_FedRM];
   } db2XaRecoverStruct;

typedef SQL_STRUCTURE rm_entry
{
  char name[SQLQG_MAX_SERVER_NAME_LEN];
  SQLXA_XID    xid;
} rm_entry;
```

Parámetros de la API db2XaListIndTrans

versionNumber
Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro, pParmStruct.

pParmStruct
Entrada. Puntero a la estructura db2XaListIndTransStruct.

pSqlca
Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2XaListIndTransStruct

piIndoubtData
Entrada. Puntero al almacenamiento intermedio suministrado por la aplicación al que se devolverán los datos dudosos. Los datos dudosos están en el formato db2XaRecoverStruct. La aplicación puede recorrer la lista de transacciones dudosas mediante el tamaño de la estructura de db2XaRecoverStruct, empezando por la dirección proporcionada por este parámetro.

Si el valor es NULL, DB2 calculará el tamaño del almacenamiento necesario y devolverá este valor en oReqBufferLen. oNumIndoubtsTotal contendrá el número total de transacciones dudosas. La aplicación puede asignar el tamaño de almacenamiento intermedio necesario y emitir de nuevo la API.

iIndoubtDataLen
Entrada. Tamaño del almacenamiento intermedio apuntado por el parámetro piIndoubtData en bytes.

oNumIndoubtsReturned
Salida. El número de registros de transacciones dudosas devuelto en el almacenamiento intermedio especificado por pIndoubtData.

oNumIndoubtsTotal
Salida. El número total de registros de transacciones dudosas disponibles en el momento de la invocación de la API. Si el almacenamiento intermedio de piIndoubtData es demasiado pequeño para contener todos los registros, oNumIndoubtsTotal será mayor que el total de oNumIndoubtsReturned. La aplicación puede volver a emitir la API para poder obtener todos los registros.

Nota: Este número puede cambiar entre invocaciones de la API como resultado de la resincronización de transacciones dudosas automáticas o heurísticas, o bien como resultado de que otras transacciones acceden al estado dudoso.

oReqBufferLen
Salida. Longitud del almacenamiento intermedio necesario para contener todos los registros de transacciones dudosas en el momento de la invocación de la API. La aplicación puede utilizar este valor para determinar el tamaño del almacenamiento intermedio necesario invocando la API con IndoubtData establecido en NULL. A continuación, este valor se puede utilizar para asignar el almacenamiento intermedio necesario y se puede emitir la API con pIndoubtData establecido en la dirección del almacenamiento intermedio asignado.

Nota: El tamaño del almacenamiento intermedio necesario puede cambiar entre invocaciones de la API como resultado de la resincronización de transacciones dudosas automáticas o heurísticas o como resultado de que otras transacciones acceden al estado dudoso. La aplicación puede asignar un almacenamiento mayor para tener esto en cuenta.

Parámetros de la estructura de datos db2XaRecoverStruct

timestamp
Salida. Especifica la hora a la que la transacción ha entrado en el estado dudoso.

xid Salida. Especifica el identificador XA asignado por el gestor de transacciones para identificar de forma exclusiva una transacción global.

dbalias
Salida. Especifica el alias de la base de datos en la que se encuentra la transacción dudosa.

applid
Salida. Especifica el identificador de aplicación asignado por el gestor de bases de datos para esta transacción.

sequence_no
Salida. Especifica el número de secuencia asignado por el gestor de bases de datos como una extensión al applid.

auth_id
Salida. Especifica el ID de autorización del usuario que ejecutó la transacción.

log_full
Salida. Indica si esta transacción ha causado o no una condición de anotación cronológica llena. Los valores válidos son:

SQLXA_TRUE
Esta transacción dudosa ha causado una condición de anotación cronológica llena.

SQLXA_FALSE
Esta transacción dudosa no ha causado una condición de anotación cronológica llena.

connected
Indica si una aplicación está conectada.

Los valores posibles para CONNECTED (definidos en sqlxa) son:

SQLXA_TRUE
Verdadero. La transacción se está sometiendo a un punto de sincronismo normal y está esperando la segunda fase de la confirmación de dos fases.

SQLXA_FALSE
Falso. La transacción se ha quedado dudosa por un error anterior y ahora está esperando una resincronización de un gestor de transacciones.

indoubt_status
Salida. Indica el estado de esta transacción dudosa. Los valores válidos son:

- SQLXA_TS_PREP
La transacción está preparada. El parámetro conectado se puede utilizar para determinar si la transacción está esperando la segunda fase del proceso de confirmación normal o si se ha producido un error y es necesaria la resincronización con el gestor de transacciones.

- SQLXA_TS_HCOM
La transacción se ha confirmado heurísticamente.

- SQLXA_TS_HROL
La transacción se ha retrotraído heurísticamente.

- SQLXA_TS_MACK
En la transacción falta acuse de recibo de confirmación de un nodo en una base de datos particionada.

- SQLXA_TS_END
La transacción ha finalizado en esta base de datos. Esta transacción se puede reactivar, confirmar o retrotraer más adelante. También es posible que el gestor de transacciones haya detectado un error y la transacción no se completará. Si este fuera el caso, esta transacción requiere acciones heurísticas porque puede que esté manteniendo bloqueos e impidiendo que otras aplicaciones accedan a los datos.

Cuando el parámetro de origen se establece en el valor SQLXA_ORIG_FXA, los valores válidos del parámetro indoubt_status (definidos en el archivo sqlxa.h ubicado en el directorio de inclusión) son:

SQLXA_TS_MFCACK
Indica que en la transacción falta acuse de recibo de confirmación de una o más fuentes de datos federadas.

SQLXA_TS_MFRACK
Indica que en la transacción falta acuse de recibo de retrotracción de una o más fuentes de datos federadas.

originator
Identifica el origen de una transacción dudosa.

Los valores posibles para ORIGINATOR (definidos en el archivo sqlxa.h ubicado en el directorio de inclusión) son:

SQLXA_ORIG_PE
Transacción originada por DB2 en el entorno MPP.

SQLXA_ORIG_XA
Transacción originada por XA.

SQLXA_ORIG_FXA
Transacción originada en la segunda fase del proceso de confirmación de dos fases federado. Indica que esta transacción ha entrado en la segunda fase del protocolo de confirmación de dos fases; sin embargo, una o más fuentes de datos federadas no pueden completar la segunda fase o no pueden comunicarse con el servidor federado.

reserved
El primer byte se utiliza para indicar el tipo de transacción dudosa: 0 indica RM y 1 indica TM.

rmn Salida. Número de fuentes de datos federadas que no han conseguido confirmar o retrotraer una transacción.

rm_list
Salida. Lista de entradas de fuentes de datos federadas anómalas, cada una de las cuales contiene un nombre de servidor y un xid.

Parámetros de la estructura de datos rm_entry

name Salida. Nombre de una fuente de datos federada.

xid Salida. Especifica el identificador XA asignado por la base de datos federada para identificar de forma exclusiva una transacción federada.

Notas de uso

SQLXA_MAX_FEDRM se ha definido en 16. En la mayoría de transacciones federadas participan menos de 10 fuentes de datos. Si más de 16 fuentes de datos federadas no consiguen confirmarse ni retrotraerse en una transacción, la API db2XaListIndTrans sólo devolverá 16 de ellas para esta transacción dudosa. Para una transacción dudosa no federada, el parámetro rm se establecerá en 0, lo que indica que en la transacción dudosa no participa ninguna fuente de datos federada.

Si en una transacción dudosa federada se ven implicadas más de 16 fuentes de datos anómalas federadas, cuando se invoca el proceso heurístico, todas las fuentes de datos (independientemente de si las ha devuelto la API db2XaListIndTrans) confirmarán o retrotraerán la transacción dudosa. Las fuentes de datos federadas que hayan confirmado o retrotraído correctamente la transacción dudosa se eliminarán de la lista de fuentes de datos federadas anómalas para la transacción dudosa federada. En la siguiente llamada a la API db2XaListIndTrans, sólo las fuentes de datos federadas que sigan sin conseguir confirmar o retrotraer la transacción dudosa permanecerán en la lista de transacciones dudosas federadas.

Para obtener la lista de fuentes de datos en una transacción dudosa federada, debe compilar aplicaciones utilizando archivos de cabecera de DB2 Versión 9.1 y transferir un número de versión db2Version900 o superior (para releases posteriores) a la API db2XaListIndTrans. Si transfiere un número de versión inferior, la API seguirá devolviendo una lista de transacciones dudosas, pero se excluirá la información de fuente de datos federada. Sea como sea, la versión del archivo de cabecera utilizada por la aplicación debe estar sincronizada con el número de versión transferido a la API. De lo contrario, los resultados serán imprevisibles.

Una aplicación típica realizará los pasos siguientes después de establecer la conexión actual con la base de datos o con el nodo del coordinador de bases de datos particionadas:

1. Invoque db2XaListIndTrans con piIndoubtData establecido en el valor NULL. Se devolverán valores en oReqBufferLen y oNumIndoubtsTotal.
2. Utilice el valor devuelto en oReqBufferLen para asignar un almacenamiento intermedio. Es posible que este almacenamiento intermedio no sea suficientemente grande si hay más transacciones dudosas debido a la invocación inicial de esta API para obtener oReqBufferLen. La aplicación puede proporcionar un almacenamiento intermedio mayor que oReqBufferLen.
3. Determine si se han obtenido todos los registros de transacciones dudosas. Esto se puede conseguir comparando oNumIndoubtsReturned con oNumIndoubtsTotal. Si oNumIndoubtsTotal es mayor que oNumIndoubtsReturned, la aplicación puede repetir los pasos anteriores.

sqlxhfrg - Olvidar estado de transacción

Permite que el gestor de recursos libere recursos retenidos por una transacción finalizada heurísticamente (es decir, una transacción que se ha confirmado o retrotraído heurísticamente). Puede invocar esta API después de confirmar o retrotraer heurísticamente una transacción XA dudosa.

Autorización

Ninguna

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlxa.h
```

Sintaxis de la API y de las estructuras de datos

```
extern int SQL_API_FN sqlxhfrg(
   SQLXA_XID *pTransId,
   struct sqlca *pSqlca
   );
```

Parámetros de la API sqlxhfrg

pTransId
Entrada. Identificador XA de la transacción que se debe olvidar o eliminar heurísticamente del archivo de anotaciones de la base de datos.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

La operación FORGET solamente se puede aplicar a las transacciones cuyo estado sea el de transacción confirmada o retrotraída heurísticamente.

sqlxphcm - Confirmar una transacción dudosa

Confirma una transacción dudosa (es decir, una transacción que está preparada para ser confirmada). Si la operación se realiza satisfactoriamente, el estado de la transacción pasa a estar confirmado heurísticamente.

Ámbito

Esta API solamente afecta al nodo en el que se ejecuta.

Autorización

Ninguna

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlxa.h
```

Sintaxis de la API y de las estructuras de datos

```
extern int SQL_API_FN sqlxphcm(
   int exe_type,
   SQLXA_XID *pTransId,
   struct sqlca *pSqlca
   );
```

Parámetros de la API sqlxphcm

exe_type
Entrada. Si se especifica EXE_THIS_NODE, la operación solamente se ejecuta en este nodo.

pTransId
Entrada. Identificador XA de la transacción que se debe confirmar heurísticamente.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Solamente se pueden confirmar las transacciones cuyo estado sea el de preparado. Una vez confirmada la transacción heurísticamente, el gestor de bases de datos recuerda el estado de la transacción hasta que se invoca la API sqlxhfrg.

sqlxphrl - Retrotraer una transacción dudosa

Retrotrae una transacción dudosa (es decir, una transacción que se ha preparado). Si la operación se realiza satisfactoriamente, el estado de la transacción pasa a estar retrotraído heurísticamente.

Ámbito

Esta API solamente afecta al nodo en el que se ejecuta.

Autorización

Ninguna

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqlxa.h
```

Sintaxis de la API y de las estructuras de datos

```
extern int SQL_API_FN sqlxphrl(
   int exe_type,
   SQLXA_XID *pTransId,
   struct sqlca *pSqlca
   );
```

Parámetros de la API sqlxphrl

exe_type
Entrada. Si se especifica EXE_THIS_NODE, la operación solamente se ejecuta en este nodo.

pTransId
Entrada. Identificador XA de la transacción que se debe retrotraer heurísticamente.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Solamente se pueden retrotraer las transacciones cuyo estado sea el de preparado o inactivo. Una vez retrotraída la transacción heurísticamente, el gestor de bases de datos recuerda el estado de la transacción hasta que se invoca la API sqlxhfrg.

Capítulo 7. Aplicaciones por hebras con acceso simultáneo

sqleAttachToCtx - Conectar a contexto

Hace que la hebra actual utilice un contexto especificado. Todas las llamadas subsiguientes a la base de datos realizadas en esta hebra utilizarán este contexto. Si se enlaza más de una hebra a un contexto determinado, se serializa el acceso para estas hebras y las hebras comparten un ámbito de confirmación.

Ámbito

El ámbito de esta API está limitado al proceso inmediato.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sql.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqleAttachToCtx (
        void * pCtx,
        void * reserved,
        struct sqlca * pSqlca);
```

Parámetros de la API sqleAttachToCtx

pCtx Entrada. Contexto válido asignado previamente por qleBeginCtx.

reserved
Reservado para una utilización futura. Su valor se debe establecer en NULL.

pSqlca
Salida. Puntero a la estructura sqlca.

sqleBeginCtx - Crear y conectar a un contexto de aplicación

Crea un contexto de aplicación, o bien crea y luego conecta a un contexto de aplicación. Se puede crear más de un contexto de aplicación. Cada contexto tiene su propio de ámbito de confirmación. Hebras diferentes se pueden enlazar a contextos diferentes (vea la API sqleAttachToCtx). Las llamadas de API a la base de datos realizadas por estas hebras no se serializan unas con otras.

Ámbito

El ámbito de esta API está limitado al proceso inmediato.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sql.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqleBeginCtx (
        void ** ppCtx,
        sqlint32 lOptions,
        void * reserved,
        struct sqlca * pSqlca);
```

Parámetros de la API sqleBeginCtx

ppCtx Salida. Área de datos a la que se asigna memoria privada para el almacenamiento de información de contexto.

lOptions
Entrada. Los valores válidos son:

SQL_CTX_CREATE_ONLY
Se asigna memoria de contexto, pero no habrá enlace.

SQL_CTX_BEGIN_ALL
Se asigna memoria de contexto y luego se realiza una llamada a sqleAttachToCtx para la hebra actual. Si se utiliza esta opción, el parámetro ppCtx puede ser nulo. Si la hebra ya está enlazada a un contexto, la llamada fallará.

reserved
Reservado para una utilización futura. Su valor se debe establecer en NULL.

pSqlca
Salida. Puntero a la estructura sqlca.

sqleDetachFromCtx - Desconectar de contexto

Desconecta el contexto que está utilizando la hebra actual. El contexto se desconecta solamente si previamente se ha establecido una conexión a ese contexto.

Ámbito

El ámbito de esta API está limitado al proceso inmediato.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sql.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqleDetachFromCtx (
        void * pCtx,
        void * reserved,
        struct sqlca * pSqlca);
```

Parámetros de la API sqleDetachFromCtx

pCtx Entrada. Contexto válido asignado previamente por qleBeginCtx.

reserved
Reservado para una utilización futura. Su valor se debe establecer en NULL.

pSqlca
Salida. Puntero a la estructura sqlca.

---

sqleEndCtx - Desconectar y liberar la memoria asociada a un contexto de aplicación

Libera toda la memoria asociada a un contexto determinado.

Ámbito

El ámbito de esta API está limitado al proceso inmediato.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sql.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqleEndCtx (
        void ** ppCtx,
        sqlint32 lOptions,
        void * reserved,
        struct sqlca * pSqlca);
```

Parámetros de la API sqleEndCtx

ppCtx Salida. Área de datos situada en la memoria privada (utilizada para almacenar información de contexto) que se libera.

lOptions
Entrada. Los valores válidos son:

SQL_CTX_FREE_ONLY
Se libera la memoria de contexto solamente si antes se ha realizado un desenlace.

Nota: pCtx debe ser un contexto válido asignado previamente por qleBeginCtx.

SQL_CTX_END_ALL
Si es necesario, se realiza una llamada a sqleDetachFromCtx antes de liberar la memoria.

Nota: Se realiza un desenlace aunque el contexto esté todavía en uso. Si se utiliza esta opción, el parámetro ppCtx puede ser nulo, pero si se pasa el parámetro, debe ser un contexto válido asignado previamente por sqleBeginCtx. Se realiza una llamada a sqleGetCurrentCtx, y se libera el contexto actual.

reserved
Reservado para una utilización futura. Su valor se debe establecer en NULL.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Si existe una conexión de base de datos, o el contexto ha sido enlazado por otra hebra, esta llamada fallará.

Nota: Si un contexto llama a una API que establece un enlace de instancia (por ejemplo, db2CfgGet), es necesario desenlazar la instancia mediante sqledtin antes de invocar sqleEndCtx.

sqleGetCurrentCtx - Obtener contexto actual

Devuelve el contexto actual que está asociado a una hebra.

Ámbito

El ámbito de esta API está limitado al proceso inmediato.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

`sql.h`

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqleGetCurrentCtx (
        void ** ppCtx,
        void * reserved,
        struct sqlca * pSqlca);
```

Parámetros de la API sqleGetCurrentCtx

ppCtx Salida. Área de datos a la que se asigna memoria privada para el almacenamiento de información de contexto.

reserved
Reservado para una utilización futura. Su valor se debe establecer en NULL.

pSqlca
Salida. Puntero a la estructura sqlca.

---

sqleInterruptCtx - Interrumpir contexto

Interrumpe el contexto especificado.

Ámbito

El ámbito de esta API está limitado al proceso inmediato.

Autorización

Ninguna

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sql.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqleInterruptCtx (
        void * pCtx,
        void * reserved,
        struct sqlca * pSqlca);
```

Parámetros de la API sqleInterruptCtx

pCtx Entrada. Contexto válido asignado previamente por qleBeginCtx.

reserved
Reservado para una utilización futura. Su valor se debe establecer en NULL.

pSqlca
Salida. Puntero a la estructura sqlca.

Notas de uso

Durante el proceso, esta API:
- Conmuta al contexto que se ha pasado como entrada
- Envía una interrupción
- Conmuta al contexto original
- Sale.

sqleSetTypeCtx - Definir el tipo de contexto de aplicación

Define el tipo de contexto de aplicación. Esta API debe ser la primera API de base de datos a la que se llama dentro de una aplicación.

Ámbito

El ámbito de esta API está limitado al proceso inmediato.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sql.h
```

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN
  sqleSetTypeCtx (
        sqlint32 lOptions);
```

Parámetros de la API sqleSetTypeCtx

lOptions
Entrada. Los valores válidos son:

SQL_CTX_ORIGINAL
Todas las hebras utilizarán el mismo contexto, y se bloqueará el acceso simultáneo. Esto es el valor por omisión si no se invoca ninguna de estas API.

SQL_CTX_MULTI_MANUAL
Todas las hebras utilizarán contextos separados, y corresponde a la aplicación gestionar el contexto de cada hebra. Consulte
- API sqleBeginCtx
- API sqleAttachToCtx
- API sqleDetachFromCtx
- API sqleEndCtx

Son aplicables las restricciones/cambios siguientes cuando se utiliza esta opción:

- Cuando la terminación es normal, la confirmación automática durante la terminación del proceso está inhabilitado. Todas las transacciones pendientes se retrotraen y todas las confirmaciones se deben realizar explícitamente.
- La API sqleintr interrumpe todos los contextos. Para interrumpir un contexto determinado, utilice sqleInterruptCtx.

Notas de uso

Esta API se debe llamar antes que cualquiera otra llamada de base de datos, y solamente la primera llamada es efectiva.

Capítulo 8. Plugins del sistema de base de datos de DB2 para personalizar la gestión de bases de datos

Los productos de base de datos de DB2 se suministran con interfaces de plugin que los usuarios y terceros pueden utilizar para personalizar algunas funciones de gestión de bases de datos.

Actualmente, los sistemas de bases de datos de DB2 tienen tres tipos de plugins:

- Plugins de seguridad para personalizar el comportamiento de búsqueda de pertenencia de grupo y la autenticación del sistema de base de datos de DB2
- Plugins de copia de seguridad y restauración para realizar copias de seguridad de datos y restaurarlos en dispositivos que no reciben soporte de los recursos de copia de seguridad y restauración suministrados por los sistemas de bases de datos de DB2
- Plugin de compresión para comprimir y descomprimir imágenes de copia de seguridad

Las funciones proporcionadas por los tres plugins anteriores se suministran con productos de sistema de base de datos de DB2, no obstante, si desea personalizar o aumentar el comportamiento del sistema de base de datos de DB2 puede grabar su propio plugin o adquirir uno del proveedor.

Cada plugin es una biblioteca que se puede cargar dinámicamente de API y estructuras de datos. Los sistemas de bases de datos de DB2 proporcionan los prototipos para las API y las estructuras de datos, y el proveedor proporciona la implantación. Los sistemas de bases de datos de DB2 proporcionan la implantación de algunas API y estructuras de datos. Para obtener una lista de las API de plugin y de las estructuras de datos que los sistemas de bases de datos de DB2 implementan, consulte el tema del plugin correspondiente. La implementación es en forma de biblioteca compartida en sistemas UNIX y una DLL en plataformas Windows. Para conocer la ubicación real donde los sistemas de bases de datos de DB2 buscan un plugin concreto, consulte el tema del plugin correspondiente.

Una API de plugin se distingue de una API de DB2 (por ejemplo, db2Export, db2Backup) en dos aspectos. En primer lugar, en la mayoría de los casos el proveedor proporciona la implementación de una API de plugin, mientras que DB2 proporciona la implementación de una API de DB2. En segundo lugar, DB2 invoca la API de plugin, mientras que el usuario de una aplicación cliente invoca la API de DB2. Por lo tanto, si un tema sobre API de plugin lista un parámetro como entrada, esto indica que DB2 rellena un valor para el parámetro, y si el parámetro se lista como salida la implementación del proveedor de la API es el responsable de rellenar un valor para el parámetro.

Habilitación de los plugins

Despliegue de un plugin de recuperación de grupos

Si desea personalizar el mecanismo de recuperación de grupos del sistema de seguridad de DB2 puede desarrollar su propio plugin de recuperación de grupos o adquirirlo a terceros.

Una vez obtenido un plugin de recuperación de grupos que sea apropiado para su sistema de gestión de bases de datos, puede desplegar el plugin.

- Para desplegar un plugin de recuperación de grupos en el servidor de bases de datos, siga estos pasos:
  1. Copie la biblioteca de plugins de recuperación de grupos en el directorio de plugins de grupo del servidor.
  2. Actualice el parámetro de configuración *group_plugin* del gestor de bases de datos con el nombre del plugin.
- Para desplegar un plugin de recuperación de grupos en clientes de bases de datos, siga estos pasos:
  1. Copie la biblioteca de plugins de recuperación de grupos en el directorio de plugins de grupo del cliente.
  2. En el cliente de base de datos, actualice el parámetro de configuración *group_plugin* del gestor de bases de datos con el nombre del plugin.

Despliegue de un plugin de ID de usuario/contraseña

Si desea personalizar el mecanismo de autentificación basado en un ID de usuario/contraseña del sistema de seguridad de DB2, puede desarrollar sus propios plugins de autentificación por ID de usuario/contraseña o adquirirlos de terceros.

Dependiendo de la utilización prevista, todos los plugins de autentificación por ID de usuario/contraseña se deben colocar en el directorio de plugins del cliente o en el directorio de plugins del servidor. Si un plugin se coloca en el directorio de plugins del cliente, se utilizará para comprobar la autorización local y para validar al cliente cuando este intente conectar con el servidor. Si el plugin se coloca en el directorio de plugins del servidor, se utilizará para gestionar las conexiones entrantes dirigidas al servidor y para comprobar si un ID de autorización existe y es válido cada vez que se emita una sentencia GRANT sin especificar las palabras claves USER ni GROUP. En la mayoría de los casos, la autentificación basada en un ID de usuario/contraseña necesita solamente un plugin en el extremo servidor. También es posible, aunque generalmente se considera menos útil, tener solamente un plugin de ID de usuario/contraseña en el cliente. Es posible, aunque poco habitual, que sea necesario tener tanto en el cliente como en el servidor plugins de ID de usuario/contraseña que sean iguales.

Nota: Es necesario detener el servidor de DB2 o las aplicaciones que utilicen los plugins antes de desplegar una versión *nueva* de un plugin *existente*. Si un proceso sigue utilizando un plugin cuando se copia una versión nueva (con el mismo nombre) sobre el mismo, el comportamiento es imprevisible y puede conllevar interrupciones. Esta restricción no es aplicable la primera vez que se despliega un plugin o cuando éste no se está utilizando.

Una vez obtenidos los plugins de autentificación por ID de usuario/contraseña que sean apropiados para su sistema de gestión de bases de datos, puede desplegar los plugin.

- Para desplegar un plugin de autentificación por ID de usuario/contraseña en el servidor de bases de datos, siga estos pasos en el servidor de bases de datos:
  1. Copie la biblioteca de plugins de autentificación por ID de usuario/contraseña en el directorio de plugins del servidor.
  2. Actualice el parámetro de configuración *srvcon_pw_plugin* del gestor de bases de datos con el nombre del plugin del servidor. El servidor utiliza este plugin para gestionar peticiones de conexión (CONNECT y ATTACH).

3. Efectúe una de estas dos acciones:
   - Asigne el tipo de autentificación CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT o DATA_ENCRYPT_CMP al parámetro de configuración *srvcon_auth* del gestor de bases de datos. O bien:
   - Asigne el valor NOT_SPECIFIED al parámetro de configuración *srvcon_auth* del gestor de bases de datos y asigne el tipo de autentificación CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT o DATA_ENCRYPT_CMP al parámetro *authentication*.

- Para desplegar un plugin de autentificación por ID de usuario/contraseña en clientes de bases de datos, siga estos pasos en cada cliente:
  1. Copie la biblioteca de plugins de autentificación por ID de usuario/contraseña en el directorio de plugins del cliente.
  2. Actualice el parámetro de configuración *clnt_pw_plugin* del gestor de bases de datos con el nombre del plugin de cliente. Este plugin se carga e invoca sin importar dónde se realiza la autentificación, no solamente cuando se asigna el valor CLIENT al parámetro de configuración *authentication* de la base de datos.
- Para la autorización local en un cliente, servidor o pasarela, que haga uso del plugin de autentificación por ID de usuario/contraseña, siga estos pasos en cada cliente, servidor o pasarela:
  1. Copie la biblioteca del plugin de autentificación por ID de usuario/contraseña en el directorio de plugins de cliente del cliente, servidor o pasarela.
  2. Actualice el parámetro de configuración *clnt_pw_plugin* del gestor de bases de datos con el nombre del plugin.
  3. Asigne el valor CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT o DATA_ENCRYPT_CMP al parámetro de configuración *authentication* del gestor de bases de datos.

Despliegue de un plugin de GSS-API

Si desea personalizar el mecanismo de autentificación del sistema de seguridad de DB2, puede desarrollar sus propios plugins de autentificación utilizando GSS-API o adquirirlos de terceros.

En el caso de plugins que no sean de tipo Kerberos, el nombre del plugin del cliente y del servidor debe ser igual y ambos deben ser del mismo tipo. No es necesario que los plugins del cliente y el servidor sean del mismo proveedor, pero deben generar y utilizar símbolos de GSS-API compatibles. Es válido tener cualquier combinación de plugins Kerberos desplegados en el cliente y el servidor, pues los plugins de Kerberos están estandarizados. Sin embargo, implementaciones diferentes de mecanismos de GSS-API menos estandarizados, como los certificados *x.509*, podrían ser solo parcialmente compatibles con los sistemas de base de datos DB2. Dependiendo de la utilización prevista, todos los plugins de autentificación de GSS-API se deben colocar en el directorio de plugins del cliente o en el directorio de plugins del servidor. Si un plugin se coloca en el directorio de plugins del cliente, se utilizará para la comprobación de la autorización local y cada vez que un cliente intente conectar con el servidor. Si el plugin se coloca en el directorio de plugins del servidor, se utilizará para gestionar las conexiones entrantes dirigidas al servidor y para comprobar si un ID de autorización existe y es válido cada vez que se emita una sentencia GRANT sin especificar las palabras claves USER ni GROUP.

Nota: Es necesario detener el servidor de DB2 o las aplicaciones que utilicen los plugins antes de desplegar una versión *nueva* de un plugin *existente*. Si un proceso sigue utilizando un plugin cuando se copia una versión nueva (con el mismo nombre) sobre el mismo, el comportamiento es imprevisible y puede conllevar interrupciones. Esta restricción no es aplicable la primera vez que se despliega un plugin o cuando éste no se está utilizando.

Una vez obtenidos los plugins de autentificación de GSS-API que sean apropiados para su sistema de gestión de bases de datos, puede desplegar los plugin.

- Para desplegar un plugin de autentificación de GSS-API en el servidor de bases de datos, siga estos pasos en el servidor:
  1. Copie la biblioteca de plugins de autentificación de GSS-API en el directorio de plugins del servidor. Puede copiar varios plugins de GSS-API en ese directorio.
  2. Actualice el parámetro de configuración *srvcon_gssplugin_list* del gestor de bases de datos con una lista ordenada, delimitada por comas, de los plugins instalados en el directorio de plugins de GSS-API.
  3. Efectúe una de estas dos acciones:
     - Puede asignar el valor GSSPLUGIN o GSS_SERVER_ENCRYPT al parámetro de configuración *srvcon_auth* del gestor de bases de datos para hacer que el servidor utilice el método de autentificación GSSAPI PLUGIN. O bien:
     - Puede asignar el valor NOT_SPECIFIED al parámetro de configuración *srvcon_auth* del gestor de bases de datos y asignar el valor GSSPLUGIN o GSS_SERVER_ENCRYPT al parámetro *authentication* del gestor de bases de datos para hacer que el servidor utilice el método de autentificación GSSAPI PLUGIN.
- Para desplegar un plugin de autentificación de GSS-API en clientes de bases de datos, siga estos pasos en cada cliente:
  1. Copie la biblioteca de plugins de autentificación de GSS-API en el directorio de plugins del cliente. Puede copiar varios plugins de GSS-API en ese directorio. Durante una operación de conexión (CONNECT o ATTACH), el cliente elige un plugin de GSS-API para la autentificación seleccionando el primer plugin de GSS-API contenido en la lista de plugins del servidor que existe en el cliente.
  2. Opcional: catalogue las bases de datos a las que accederá el cliente, e indique que el cliente solo aceptará como mecanismo de autentificación un plugin de autentificación de GSS-API. Por ejemplo:

     ```
     CATALOG DB testdb AT NODE testnode AUTHENTICATION GSSPLUGIN
     ```
- Para la autorización local en un cliente, servidor o pasarela, que hace uso de un plugin de autentificación de GSS-API, siga estos pasos:
  1. Copie la biblioteca de plugins de autentificación de GSS-API en el directorio de plugins de cliente del cliente, servidor o pasarela.
  2. Actualice el parámetro de configuración *local_gssplugin* del gestor de bases de datos con el nombre del plugin.
  3. Asigne el valor GSSPLUGIN o GSS_SERVER_ENCRYPT al parámetro de configuración *authentication* del gestor de bases de datos.

Despliegue de un plugin de Kerberos

Si desea personalizar el mecanismo de autentificación Kerberos del sistema de seguridad de DB2, puede desarrollar sus propios plugins de autentificación Kerberos o adquirir uno de terceros. Observe que el plugin de seguridad de Kerberos no es compatible con IPv6.

Nota: Es necesario detener el servidor de DB2 o las aplicaciones que utilicen los plugins antes de desplegar una versión *nueva* de un plugin *existente*. Si un proceso sigue utilizando un plugin cuando se copia una versión nueva (con el mismo nombre) sobre el mismo, el comportamiento es imprevisible y puede conllevar interrupciones. Esta restricción no es aplicable la primera vez que se despliega un plugin o cuando éste no se está utilizando.

Una vez obtenidos los plugins de autentificación Kerberos que sean apropiados para su sistema de gestión de bases de datos, puede desplegar los plugin.

- Para desplegar un plugin de autentificación de Kerberos en el servidor de bases de datos, siga estos pasos en el servidor:
  1. Copie la biblioteca de plugins de autentificación de Kerberos en el directorio de plugins del servidor.
  2. Actualice el parámetro de configuración **srvcon_gssplugin_list** del gestor de bases de datos con el nombre del plugin Kerberos del servidor. Ese parámetro es una lista ordenada delimitada por comas. Un solo plugin de esta lista puede ser un plugin Kerberos. Si la lista está en blanco y **authentication** tiene el valor KERBEROS o KRB_SVR_ENCRYPT, se utilizará el plugin Kerberos de DB2 por omisión: IBMkrb5.
  3. Si es necesario, establezca el parámetro de configuración del gestor de bases de datos **srvcon_auth** para alterar temporalmente el tipo de autentificación actual. Si el parámetro de configuración del gestor de bases de datos **srvcon_auth** no está establecido, el gestor de bases de datos de DB2 utiliza el valor del parámetro **authentication**. Si el parámetro de configuración **authentication** está establecido en alguno de los tipos de autentificación siguientes, puede desplegar y utilizar un plugin de Kerberos:
     - KERBEROS
     - KRB_SERVER_ENCRYPT
     - GSSPLUGIN
     - GSS_SERVER_ENCRYPT

     Si necesita alterar temporalmente el tipo de autentificación actual, asigne al parámetro de configuración **srvcon_auth** uno de los tipos de autentificación siguientes:
     - KERBEROS
     - KRB_SERVER_ENCRYPT
     - GSSPLUGIN
     - GSS_SERVER_ENCRYPT
- Para desplegar un plugin de autentificación de Kerberos en clientes de bases de datos, siga estos pasos en cada cliente:
  1. Copie la biblioteca de plugins de autentificación de Kerberos en el directorio de plugins del cliente.
  2. Actualice el parámetro de configuración **clnt_krb_plugin** del gestor de bases de datos con el nombre del plugin Kerberos. Si **clnt_krb_plugin** está en blanco, DB2 considera que el cliente no puede utilizar la autentificación Kerberos. Esto solo es apropiado cuando el servidor no puede utilizar

plugins. Si tanto el servidor como el cliente pueden utilizar plugins de seguridad, el plugin por omisión del servidor, *IBMkrb5*, tiene preferencia sobre el valor especificado para **clnt_krb_plugin** en el cliente. Para la autorización local en un cliente, servidor o pasarela, que haga uso de un plugin de autentificación Kerberos, siga estos pasos:

a. Copie la biblioteca de plugins de autentificación Kerberos en el directorio de plugins de cliente del cliente, servidor o pasarela.
b. Actualice el parámetro de configuración **clnt_krb_plugin** del gestor de bases de datos con el nombre del plugin.
c. Asigne el valor KERBEROS o KRB_SERVER_ENCRYPT al parámetro de configuración **authentication** del gestor de bases de datos.

3. Opcional: catalogue las bases de datos a las que accederá el cliente, e indique que el cliente solo utilizará un plugin de autentificación Kerberos. Por ejemplo:

```
CATALOG DB testdb AT NODE testnode AUTHENTICATION KERBEROS
        TARGET PRINCIPAL service/host@REALM
```

Nota: Para las plataformas que dan soporte a Kerberos, la biblioteca `IBMkrb5` estará presente en el directorio de plugins del cliente. DB2 reconocerá esta biblioteca como un plugin de GSS-API válido, pues los plugins Kerberos se implementan utilizando el plugin GSS-API.

Escritura de plugins de seguridad

Cómo carga DB2 los plugins de seguridad

Cada biblioteca de plugins debe contener una función de inicialización junto con un nombre específico determinado por el tipo de plugin:

- Plugin de autentificación del extremo servidor: `db2secServerAuthPluginInit()`
- Plugin de autentificación del extremo cliente: `db2secClientAuthPluginInit()`
- Plugin de grupo: `db2secGroupPluginInit()`

Esta función se denomina función de inicialización del plugin. La función de inicialización del plugin inicializa el plugin especificado y proporciona a DB2 la información que necesita para invocar las funciones del plugin. La función de inicialización del plugin acepta los parámetros siguientes:

- El número de versión más alto de la estructura de puntero de función que sea compatible con la instancia de DB2 que realiza la invocación del plugin pueda soportar.
- Un puntero a una estructura que contiene punteros que señalan hacia todas las API que necesitan implementación
- Un puntero a una función que añade mensajes de anotaciones al archivo db2diag.log
- Un puntero a una serie representativa de un mensaje de error
- La longitud del mensaje de error

Lo siguiente es una signatura de función para la función de inicialización de un plugin de recuperación de grupos:

```
SQL_API_RC SQL_API_FN db2secGroupPluginInit(
  db2int32 version,
  void *group_fns,
  db2secLogMessage *logMessage_fn,
  char **errormsg,
  db2int32  *errormsglen);
```

Nota: Si la biblioteca de plugins se compila como C++, todas las funciones se deben declarar con: `extern "C"`. DB2 depende del cargador dinámico subyacente del sistema operativo para manejar los métodos constructores y destructores de C++ utilizados dentro de una biblioteca de plugins de C++ escritos por el usuario.

La función de inicialización es la única función de la biblioteca de plugins que utiliza un nombre de función prescrito. Las demás funciones de plugin se referencian mediante punteros de función devueltos por la función de inicialización. Los plugins de servidor se cargan cuando se inicia el servidor DB2. Los plugins de cliente se cargan en el cliente cuando sea necesario. Inmediatamente después de cargar una biblioteca de plugins,DB2 determina la ubicación de la función de inicialización e invoca la función. La tarea específica de esta función es la siguiente:

- Convertir el puntero de función en un puntero a una estructura de función apropiada
- Especificar los punteros que señala hacia las demás funciones contenidas en la biblioteca
- Especificar el número de versión de la estructura de puntero de función devuelta

DB2 puede potencialmente invocar la función de inicialización de plugin más de una vez. Esta situación se puede producir cuando una aplicación carga dinámicamente la biblioteca del cliente DB2, la descarga y la vuelve a cargar, y luego ejecuta funciones de autentificación desde un plugin tanto antes como después de la recarga. En este caso, la biblioteca de plugins podría no descargarse y luego cargarse de nuevo; pero este comportamiento varía según el sistema operativo.

Otro ejemplo en el que DB2 emite varias llamadas a una función de inicialización de plugin es cuando se ejecutan procedimientos almacenados o llamadas de sistema federado, donde el propio servidor de bases de datos puede actuar como cliente. Si los plugins de cliente y servidor contenidos en el servidor de bases de datos residen en el mismo archivo,DB2 podría invocar dos veces la función de inicialización de plugin.

Si el plugin detecta que `db2secGroupPluginInit` se invoca más de una vez, debe interpretar esto como si le indicara que debe finalizar y reinicializar la biblioteca de plugins. De esta manera, la función de inicialización de plugin debe realizar el proceso completo de limpieza que realizaría una llamada a `db2secPluginTerm` antes de devolver de nuevo el conjunto de punteros de función.

En un servidor DB2 que se ejecute en un sistema operativo basado en UNIX o Linux, DB2 puede potencialmente cargar e inicializar bibliotecas de plugins más de una vez en procesos diferentes.

Restricciones en el desarrollo de bibliotecas de plugins de seguridad

A continuación se indican las restricciones para el desarrollo de bibliotecas de plugins.

C-linkage

Las bibliotecas de plugins se deben enlazar con C-linkage. Los archivos de cabecera donde residen los prototipos, las estructuras de datos necesarias para implementar los plugin, y las definiciones de códigos de error se

proporcionan solo para C/C++. Las funciones que DB2 resolverá durante la carga se deben declarar con ″C″ externo si la biblioteca de plugins se ha compilado como C++.

El CLR (common language runtime) .NET no está soportado
El CLR (common language runtime) .NET no está soportado para compilar y enlazar el código fuente de bibliotecas de plugins.

Manejadores de señales
Las bibliotecas de plugins no deben instalar manejadores de señales ni cambiar la máscara de señales, pues ello interferiría con los manejadores de señales de DB2. La interferencia con los manejadores de señales de DB2 podría interferir gravemente con la capacidad de DB2 para notificar los errores y recuperarse de los mismos, incluidas las interrupciones contenidas en el propio código del plugin. Las bibliotecas de plugins no deben tampoco emitir nunca excepciones de C++, pues ello puede también interferir con el manejo de errores realizado por DB2.

Hebras protegidas
Las bibliotecas de plugins deben tener hebras protegidas y ser reentrantes. La función de inicialización de plugin es la única API que no es necesario que sea reentrante. La función de inicialización de plugin podría potencialmente ser invocada varias veces desde procesos diferentes. En este caso, el plugin libera todos los recursos utilizados y se reinicializa a sí mismo.

Manejadores de rutinas de salida y cancelación de las llamadas de biblioteca C estándar y de sistema operativo
Las bibliotecas de plugins no deben cancelar las llamadas de biblioteca C estándar ni de sistema operativo. Las bibliotecas de plugins no deben tampoco instalar manejadores de rutinas de salida ni manejadores `pthread_atfork`. El uso de manejadores de rutinas de salida no es recomendable, pues pueden descargarse antes de que finalice el programa.

Dependencias de biblioteca
En Linux o UNIX, los procesos que cargan las bibliotecas de plugins pueden ser `setuid` o `setgid`, lo que significa que no podrán depender en las variables de entorno `$LD_LIBRARY_PATH`, `$SHLIB_PATH` o `$LIBPATH` para encontrar bibliotecas dependientes. Por tanto, las bibliotecas de plugins no deben depender de otras bibliotecas, a menos que las bibliotecas dependientes sean accesibles a través de otros métodos, tales como los siguientes:

- Las bibliotecas residen en `/lib` o `/usr/lib`
- Los directorios donde residen las bibliotecas están especificados a nivel del sistema operativo (por ejemplo, en el archivo `ld.so.conf` en Linux)
- Las bibliotecas están especificadas en la variable `RPATH` en la propia biblioteca de plugins

Esta restricción no es aplicable a los sistemas operativos Windows.

Colisiones de símbolos
Cuando sea posible, las bibliotecas de plugins se deben compilar y enlazar utilizando las opciones disponibles que reduzcan la probabilidad de colisiones de símbolos, tales como las opciones que reducen las referencias simbólicas externas no vinculadas. Por ejemplo, la utilización de la opción del editor de enlaces ″-Bsymbolic″ en HP, Sun Solaris y Linux puede ayudar a prevenir los problemas relacionados con la colisión de símbolos. Sin embargo, para los plugins escritos en AIX, no utilice la opción `"-brtl"` del editor de enlaces, ni explícita ni implícitamente.

Aplicaciones de 32 y 64 bits

Las aplicaciones de 32 bits deben utilizar plugins de 32 bits. Las aplicaciones de 64 bits deben utilizar plugins de 64 bits. Consulte el tema sobre las consideraciones de 32 bits y 64 bits para obtener más detalles.

Series de texto

No es seguro que las series de texto de entrada tengan terminación nula, y no es obligatorio que las series de salida tengan terminación nula. En lugar de eso, se asignan longitudes enteras a todas las series de entrada, y se asignan punteros a enteros para las longitudes a devolver.

Pase de parámetros de ID de autorización

El parámetro authid (ID de autorización) que DB2 pasa a un plugin (parámetro de entrada authid) contendrá un ID de autorización escrito en mayúsculas, donde se han eliminado los blancos de relleno. El parámetro authid que un plugin devuelve a DB2 (parámetro authid de salida) no necesita ningún tratamiento especial, pero DB2 lo convierte a mayúsculas y lo rellena con blancos de acuerdo con el estándar interno de DB2.

Límites de tamaño para parámetros

Las API de plugins utilizan lo siguiente como límites de longitud para los parámetros:

```
#define DB2SEC_MAX_AUTHID_LENGTH 255
#define DB2SEC_MAX_USERID_LENGTH 255
#define DB2SEC_MAX_USERNAMESPACE_LENGTH 255
#define DB2SEC_MAX_PASSWORD_LENGTH 255
#define DB2SEC_MAX_DBNAME_LENGTH 128
```

Una implementación de plugin determinada puede necesitar o aplicar longitudes máximas menores para los ID de autorización, los ID de usuario y las contraseñas. En particular, los plugins de autentificación del sistema operativo proporcionados con los sistemas de base de datos DB2 deben ajustarse a los límites máximos de longitud impuestos por el sistema operativo para el nombre de usuario, de grupo y de espacio de nombres en los casos en que esos límites sean más restrictivos que los indicados más arriba.

Extensiones de la biblioteca de plugins de seguridad en AIX

En los sistemas AIX, las bibliotecas de plugins de seguridad pueden tener la extensión de nombre de archivo *.a* o *.so*. El mecanismo utilizado para cargar la biblioteca de plugins depende de la extensión utilizada:

- Las bibliotecas de plugins cuya extensión de nombre de archivo es *.a* se considera que son archivadores que contienen miembros de objeto compartido. El nombre de estos miembros debe ser *shr.o* (32 bits) o *shr64.o* (64 bits). Un archivador individual puede contener miembros de 32 bits y de 64 bits, lo que permite desplegar el archivador en ambos tipos de plataformas.

  Por ejemplo, para crear una biblioteca de plugins para archivadores de 32 bits:

  ```
  xlc_r -qmkshrobj -o shr.o MyPlugin.c -bE:MyPlugin.exp
  ar rv MyPlugin.a shr.o
  ```

- Las bibliotecas de plugins cuya extensión de nombre de archivo es *.so* se considera que son objetos compartidos de carga dinámica. Un objeto de esta clase puede ser de 32 bits o 64 bits, dependiendo de las opciones utilizadas para el compilador y el editor de enlaces cuando se creó el objeto. Por ejemplo, para crear una biblioteca de plugins de 32 bits:

  ```
  xlc_r -qmkshrobj -o MyPlugin.so MyPlugin.c -bE:MyPlugin.exp
  ```

En todas las plataformas excepto AIX, se considera siempre que las bibliotecas de plugins de seguridad son objetos compartidos de carga dinámica.

Restricciones para plugins de seguridad

Existen las restricciones siguientes en el uso de los plugins de seguridad:

Restricciones de soporte de la familia de productos de base de datos DB2

No puede utilizar un plugin de GSS-API para autentificar las conexiones establecidas entre clientes de DB2 en Linux, UNIX y Windows y otros servidores de la familia DB2 como, por ejemplo,DB2 para z/OS. Tampoco es posible autentificar las conexiones establecidas entre otro producto de la familia de base de datos DB2 que actúe como cliente y un servidor de DB2 en Linux, UNIX, o Windows.

Si utiliza un cliente de DB2 en Linux, UNIX o Windows para conectar con otros servidores de la familia de base de datos DB2 puede utilizar los plugins de ID de usuario/contraseña del extremo cliente (como el plugin de autentificación del sistema operativo proporcionado por IBM) o escribir su propio plugin de ID de usuario/contraseña. También puede utilizar los plugins integrados de Kerberos o implementar plugins propios.

Con un cliente de DB2 en Linux, UNIX o Windows, no debe catalogar una base de datos utilizando el tipo de autentificación GSSPLUGIN.

Restricciones sobre el identificador AUTHID. La Versión 9.5 y posteriores del sistema de base de datos DB2 permite tener un ID de autorización de 128 bytes pero cuando el ID de autorización se interpreta como un ID de usuario del sistema operativo o nombre de grupo, se aplican las restricciones de denominación del sistema operativo (por ejemplo, una limitación a 8 o 30 caracteres para los ID de usuario y a 30 caracteres para los nombres de grupos). Por lo tanto, aunque es posible otorgar un ID de autorización de 128 bytes, resulta imposible conectarse como un usuario que tenga este ID de autorización. Si el usuario escribe su propio plugin de seguridad, podrá beneficiarse al máximo de los tamaños ampliados para el ID de autorización. Por ejemplo, podrá proporcionar al plugin de seguridad un ID de usuario de 30 bytes que podrá devolver un ID de autorización de 128 bytes durante la autentificación a la que el usuario puede conectarse.

Restricciones de soporte de WebSphere Federation Server

DB2 II no da soporte a la utilización de credenciales delegadas procedentes de un plugin de GSS_API para establecer conexiones de salida con fuentes de datos. Las conexiones con fuentes de datos deben seguir utilizando el mandato CREATE USER MAPPING.

Restricciones de soporte del Servidor de administración de bases de datos

El Servidor de administración de DB2 (DAS) no da soporte a los plugins de seguridad. El servidor DAS solo da soporte al mecanismo de autentificación del sistema operativo.

Problema de los plugin de seguridad y restricción para los clientes de DB2 (Windows)

Cuando se desarrollan plugins de seguridad que se desplegarán en clientes DB2 en sistemas operativos Windows no puede descargarse ninguna biblioteca auxiliar en la función de terminación del plugin. Esta restricción es aplicable a todos los tipos de plugins de seguridad de cliente, incluidos los plugins de grupo, ID de usuario y contraseña, Kerberos y GSS-API. Debido a que estas API de terminación como, por ejemplo, db2secPluginTerm, db2secClientAuthPluginTerm y db2secServerAuthPluginTerm no se invocan en ninguna plataforma Windows es necesario llevar a cabo la limpieza de recursos apropiada.

Esta restricción está asociada con cuestiones de limpieza referentes a la descarga de las DLL en Windows.

Carga de bibliotecas de plugins en AIX cuya extensión es .a o .so

En AIX, las bibliotecas de plugins de seguridad pueden tener la extensión de nombre de archivo `.a` o `.so`. El mecanismo utilizado para cargar la biblioteca de plugins depende de la extensión utilizada:

- Bibliotecas de plugins cuya extensión de nombre de archivo es `.a`

  Las bibliotecas de plugins cuya extensión de nombre de archivo es .a se considera que son archivadores que contienen miembros de objeto compartido. El nombre de estos miembros debe ser `shr.o` (32 bits) o `shr64.o` (64 bits). Un archivador individual puede contener miembros de 32 bits y de 64 bits, lo que permite desplegar el archivador en ambos tipos de plataformas.

  Por ejemplo, para crear una biblioteca de plugins para archivadores de 32 bits:

  ```
  xlc_r -qmkshrobj -o shr.o MyPlugin.c -bE:MyPlugin.exp
  ar rv MyPlugin.a shr.o
  ```

- Bibliotecas de plugins cuya extensión de nombre de archivo es `.so`

  Las bibliotecas de plugins cuya extensión de nombre de archivo es .so se considera que son objetos compartidos de carga dinámica. Un objeto de esta clase puede ser de 32 bits o 64 bits, dependiendo de las opciones utilizadas para el compilador y el editor de enlaces cuando se creó el objeto. Por ejemplo, para crear una biblioteca de plugins de 32 bits:

  ```
  xlc_r -qmkshrobj -o MyPlugin.so MyPlugin.c -bE:MyPlugin.exp
  ```

En todas las plataformas excepto AIX, se considera siempre que las bibliotecas de plugins de seguridad son objetos compartidos de carga dinámica.

Los plugins de seguridad de GSS-API no son compatibles con el cifrado y firma de mensajes

El cifrado y firma de mensajes no se pueden utilizar en los plugins de seguridad de GSS-API.

Códigos de retorno para plugins de seguridad

Todas las API de plugins de seguridad deben devolver un valor entero para indicar si la API se ha ejecutado con éxito o no. El valor de código de retorno 0 indica que la API se ejecutó satisfactoriamente. Todos los códigos de retorno negativos, con la excepción de -3, -4 y -5, indican que la API encontró un error.

Todos los códigos de retorno negativos devueltos por las API de plugins de seguridad se correlacionan con los códigos de SQL -1365, -1366 o -30082, excepto los códigos de retorno -3, -4 o -5. Los valores -3, -4 y -5 se utilizan para indicar si un ID de autorización representa o no un usuario o grupo válido.

Todos los códigos de retorno de las API de plugins de seguridad están definidos en db2secPlugin.h, que reside en el directorio include de DB2: SQLLIB/include.

La tabla siguiente proporciona detalles sobre todos los códigos de retorno de los plugins de seguridad:

Tabla 10. Códigos de retorno de plugins de seguridad

Código de retorno	Valor de sentencia define	Significado	API aplicable
0	DB2SEC_PLUGIN_OK	La API de plugins se ejecutó satisfactoriamente.	Todas
-1	DB2SEC_PLUGIN_UNKNOWNERROR	La API de plugins encontró un error inesperado.	Todas
-2	DB2SEC_PLUGIN_BADUSER	El ID de usuario pasado como entrada no está definido.	db2secGenerateInitialCred db2secValidatePassword db2secRemapUserid db2secGetGroupsForUser
-3	DB2SEC_PLUGIN_INVALIDUSERORGROUP	No existe el usuario o grupo especificado.	db2secDoesAuthIDExist db2secDoesGroupExist
-4	DB2SEC_PLUGIN_USERSTATUSNOTKNOWN	Estado de usuario desconocido. DB2 no trata esta condición como un error. Una sentencia GRANT la utiliza para determinar si un ID de autorización representa un usuario o grupo de sistema operativo.	db2secDoesAuthIDExist
-5	DB2SEC_PLUGIN_GROUPSTATUSNOTKNOWN	Estado de grupo desconocido. DB2 no trata esta condición como un error. Una sentencia GRANT la utiliza para determinar si un ID de autorización representa un usuario o grupo de sistema operativo.	db2secDoesGroupExist
-6	DB2SEC_PLUGIN_UID_EXPIRED	ID de usuario caducado.	db2secValidatePassword db2GetGroupsForUser db2secGenerateInitialCred
-7	DB2SEC_PLUGIN_PWD_EXPIRED	La contraseña ha caducado.	db2secValidatePassword db2GetGroupsForUser db2secGenerateInitialCred
-8	DB2SEC_PLUGIN_USER_REVOKED	Se ha revocado al usuario.	db2secValidatePassword db2GetGroupsForUser
-9	DB2SEC_PLUGIN_USER_SUSPENDED	Se ha suspendido al usuario.	db2secValidatePassword db2GetGroupsForUser

Tabla 10. Códigos de retorno de plugins de seguridad (continuación)

Código de retorno	Valor de sentencia define	Significado	API aplicable
-10	DB2SEC_PLUGIN_BADPWD	Contraseña incorrecta.	db2secValidatePassword db2secRemapUserid db2secGenerateInitialCred
-11	DB2SEC_PLUGIN _BAD_NEWPASSWORD	Contraseña nueva incorrecta.	db2secValidatePassword db2secRemapUserid
-12	DB2SEC_PLUGIN _CHANGEPASSWORD _NOTSUPPORTED	Cambio de contraseña no soportado.	db2secValidatePassword db2secRemapUserid db2secGenerateInitialCred
-13	DB2SEC_PLUGIN_NOMEM	El plugin no ha podido asignar memoria debido a que no existe memoria suficiente.	Todas
-14	DB2SEC_PLUGIN_DISKERROR	El plugin ha encontrado un error de disco.	Todas
-15	DB2SEC_PLUGIN_NOPERM	El plugin no ha podido acceder a un archivo debido a que no tiene los permisos correctos para el archivo.	Todas
-16	DB2SEC_PLUGIN_NETWORKERROR	El plugin ha encontrado un error de red.	Todas
-17	DB2SEC_PLUGIN _CANTLOADLIBRARY	El plugin no puede cargar una biblioteca necesaria.	db2secGroupPluginInit db2secClientAuthPluginInit db2secServerAuthPluginInit
-18	DB2SEC_PLUGIN_CANT _OPEN_FILE	El plugin no puede abrir y leer un archivo por una razón que no es la ausencia de un archivo ni la falta de permisos adecuados para el archivo.	Todas
-19	DB2SEC_PLUGIN_FILENOTFOUND	El plugin no puede abrir y leer un archivo porque el archivo falta en el sistema de archivos.	Todas
-20	DB2SEC_PLUGIN _CONNECTION_DISALLOWED	El plugin está rechazando la conexión debido a la restricción que establece a qué base de datos se puede conectar el plugin, o la dirección TCP/IP no puede conectar con una base de datos determinada.	Todas las API de plugins del extremo servidor.
-21	DB2SEC_PLUGIN_NO_CRED	Plugin de la API de GSS solamente: falta la credencial inicial del cliente.	db2secGetDefaultLoginContext db2secServerAuthPluginInit
-22	DB2SEC_PLUGIN_CRED_EXPIRED	Plugin de la API de GSS solamente: la credencial del cliente ha caducado.	db2secGetDefaultLoginContext db2secServerAuthPluginInit
-23	DB2SEC_PLUGIN _BAD_PRINCIPAL_NAME	Plugin de la API de GSS solamente: el nombre de principal no es válido.	db2secProcessServer PrincipalName

Tabla 10. Códigos de retorno de plugins de seguridad (continuación)

Código de retorno	Valor de sentencia define	Significado	API aplicable
-24	DB2SEC_PLUGIN _NO_CON_DETAILS	La llamada db2secGetConDetails devuelve este código de retorno (por ejemplo, de DB2 al plugin) para indicar que DB2 no puede determinar la dirección TCP/IP del cliente.	db2secGetConDetails
-25	DB2SEC_PLUGIN _BAD_INPUT_PARAMETERS	Algunos parámetros no son válidos o faltan cuando se invoca la API de plugins.	Todas
-26	DB2SEC_PLUGIN _INCOMPATIBLE_VER	La versión de las API notificada por el plugin no es compatible con DB2.	db2secGroupPluginInit db2secClientAuthPluginInit db2secServerAuthPluginInit
-27	DB2SEC_PLUGIN_PROCESS_LIMIT	No existen recursos suficientes para que el plugin cree un nuevo proceso.	Todas
-28	DB2SEC_PLUGIN_NO_LICENSES	El plugin ha encontrado un problema de licencia de usuario. Existe la posibilidad de que la licencia del mecanismo subyacente haya alcanzado su límite.	Todas

Manejo de mensajes de error para los plugins de seguridad

Cuando se produce un error en una API de plugins de seguridad, la API puede devolver una serie de texto ASCII en el campo errormsg para proporcionar una descripción del problema más específica que el código de retorno.

Por ejemplo, la serie de texto proporcionada en errormsg puede contener lo siguiente: "El archivo /home/db2inst1/mypasswd.txt no existe". DB2 escribirá la serie de texto completa en el archivo de notificaciones de administración de DB2 y también incluirá una versión abreviada como símbolo en algunos mensajes de SQL. Debido a que los símbolos contenidos en los mensajes de SQL solo pueden tener una longitud limitada, es recomendable que esos mensajes se mantengan cortos y que las partes variables importantes de esos mensajes aparezcan al principio de la serie de texto. Para facilitar la depuración de errores, puede añadir el nombre del plugin de seguridad al mensaje de error.

Para los errores no urgentes, tales como los errores de contraseña caducada, la serie de texto contenida en errormsg solo se descargará a un archivo cuando el parámetro de configuración DIAGLEVEL del gestor de bases de datos tenga el valor 4.

El plugin de seguridad es el encargado de asignar la memoria para estos mensajes de error. Por tanto, los plugins deben también proporcionar una API para liberar esta memoria: db2secFreeErrormsg.

DB2 sólo comprueba el campo errormsg si una API devuelve un valor distinto de cero. Por tanto, el plugin no debe asignar memoria para el mensaje de error devuelto si no existe ningún error.

Durante la inicialización, se pasa un puntero de función para el registro de mensajes, `logMessage_fn`, a los plugins de grupo, de cliente y de servidor. Los plugins pueden utilizar la función para registrar en db2diag.log la información de depuración disponible. Por ejemplo:

```
// Registrar un mensaje para indicar la inicialización satisfactoria
(*(logMessage_fn))(DB2SEC_LOG_CRITICAL,
                   "db2secGroupPluginInit successful",
                    strlen("db2secGroupPluginInit successful"));
```

Para obtener más detalles sobre cada parámetro de la función `db2secLogMessage`, consulte la API de inicialización correspondiente a cada tipo de plugin.

Secuencias de llamada para las API de plugins de seguridad

Éstas son las situaciones principales en las que el gestor de bases de datos DB2 llamará a las API de los plugins de seguridad:

- Desde un cliente para una conexión de base de datos (implícita y explícita)
  - CLIENT
  - Basado en servidor (SERVER, SERVER_ENCRYPT, DATA_ENCRYPT)
  - GSSAPI y Kerberos
- Desde un cliente, servidor o pasarela para una autorización local
- Desde un servidor para una conexión de base de datos
- Desde un servidor para una sentencia GRANT
- Desde un servidor para obtener una lista de los grupos a los que pertenece un ID de autorización

Nota: Los servidores de base de datos DB2 tratan como aplicaciones cliente las acciones de base de datos que requieren autorización local, como las acciones db2start, db2stop y db2trc.

Para cada una de estas operaciones, la secuencia con la que el gestor de bases de datos DB2 llama a las API de plugins de seguridad es diferente. Las secuencias de las API a las que llama el gestor de bases de datos DB2 en cada una de estas situaciones son las siguientes.

CLIENT - implícita

Cuando el tipo de autentificación configurado por el usuario es CLIENT, la aplicación cliente DB2 llama a las siguientes API de plugins de seguridad:

- `db2secGetDefaultLoginContext();`
- `db2secValidatePassword();`
- `db2secFreetoken();`

En el caso de una autentificación implícita, es decir, cuando el usuario se conecta sin especificar un ID de usuario o una contraseña concretos, se llama a la API `db2secValidatePassword` si el usuario está utilizando un plugin de ID de usuario/contraseña. Esta API permite al desarrollador del plugin prohibir la autentificación implícita si es necesario.

CLIENT - explícita

En una autentificación explícita, es decir, cuando el usuario se conecta a una base de datos en la que se especifican el ID de usuario y la contraseña, si se establece que el parámetro *authentication* de configuración del gestor de bases de datos sea igual a CLIENT, la aplicación cliente DB2 llamará varias veces a las siguientes API de plugins de seguridad si así lo exige la implementación:

- db2secRemapUserid();
- db2secValidatePassword();
- db2secFreeToken();

Basado en servidor (SERVER, SERVER_ENCRYPT, DATA_ENCRYPT) - implícita
En una autentificación implícita, cuando el cliente y el servidor han negociado la autentificación por ID de usuario/contraseña (por ejemplo, cuando se establece que el parámetro *srvcon_auth* en el servidor sea igual a SERVER, SERVER_ENCRYPT, DATA_ENCRYPT o DATA_ENCRYPT_CMP), la aplicación cliente llamará a las siguientes API de plugins de seguridad:

- db2secGetDefaultLoginContext();
- db2secFreeToken();

Basado en servidor (SERVER, SERVER_ENCRYPT, DATA_ENCRYPT) - explícita
En una autentificación explícita, cuando el cliente y el servidor han negociado la autentificación por ID de usuario/contraseña (por ejemplo, cuando se establece que el parámetro srvcon_auth en el servidor sea igual a SERVER, SERVER_ENCRYPT, DATA_ENCRYPT o DATA_ENCRYPT_CMP), la aplicación cliente llamará a las siguientes API de plugins de seguridad:

- db2secRemapUserid();

GSSAPI y Kerberos - implícita
En una autentificación implícita, cuando el cliente y el servidor han negociado la autentificación GSS-API o Kerberos (por ejemplo, cuando se establece que el parámetro *srvcon_auth* en el servidor sea igual a KERBEROS, KRB_SERVER_ENCRYPT, GSSPLUGIN o GSS_SERVER_ENCRYPT), la aplicación cliente llamará a las siguientes API de plugins de seguridad. (La llamada a gss_init_sec_context() utiliza GSS_C_NO_CREDENTIAL como credencial de entrada).

- db2secGetDefaultLoginContext();
- db2secProcessServerPrincipalName();
- gss_init_sec_context();
- gss_release_buffer();
- gss_release_name();
- gss_delete_sec_context();
- db2secFreeToken();

Con el soporte GSS-API multiflujo, se puede llamar a gss_init_sec_context() varias veces si así lo exige la implementación.

GSSAPI y Kerberos - explícita
Si el tipo de autentificación negociado es GSS-API o Kerberos, la aplicación cliente llama a las siguientes API de plugins de seguridad para los plugins de GSS-API en la siguiente secuencia. Estas API se utilizan para la autentificación implícita y para la explícita, a menos que se indique lo contrario.

- db2secProcessServerPrincipalName();
- db2secGenerateInitialCred(); (para la autentificación explícita solamente)
- gss_init_sec_context();
- gss_release_buffer ();
- gss_release_name();

- gss_release_cred();
- db2secFreeInitInfo();
- gss_delete_sec_context();
- db2secFreeToken();

Se puede llamar varias veces a la API gss_init_sec_context() si se devuelve un símbolo de autentificación mutua desde el servidor y así lo exige la implementación.

Desde un cliente, servidor o pasarela para una autorización local

Para una autorización local, el mandato de DB2 utilizado llamará a las siguientes API de plugins de seguridad:

- db2secGetDefaultLoginContext();
- db2secGetGroupsForUser();
- db2secFreeToken();
- db2secFreeGroupList();

Estas API se invocan para los mecanismos de autentificación basados en un ID de usuario/contraseña y para los mecanismos de autentificación basados en GSS-API.

Desde un servidor para una conexión de base de datos

Para una conexión de base de datos desde el servidor de bases de datos, el proceso agente o hebra de DB2 llama a las siguientes API de plugins de seguridad para el mecanismo de autentificación basado en un ID de usuario/contraseña:

- db2secValidatePassword(); solo si el parámetro *authentication* de configuración de base de datos no es igual a CLIENT
- db2secGetAuthIDs();
- db2secGetGroupsForUser();
- db2secFreeToken();
- db2secFreeGroupList();

Para una conexión (CONNECT) a una base de datos, el proceso agente o la hebra de DB2 llamará a las siguientes API de plugins de seguridad en el caso del mecanismo de autentificación basado en GSS-API:

- gss_accept_sec_context();
- gss_release_buffer();
- db2secGetAuthIDs();
- db2secGetGroupsForUser();
- gss_delete_sec_context();
- db2secFreeGroupListMemory();

Desde un servidor para una sentencia GRANT

Para una sentencia GRANT que no especifique la palabra clave USER o GROUP, (por ejemplo, "GRANT CONNECT ON DATABASE TO user1"), el proceso agente o la hebra de DB2 debe poder determinar si user1 es un usuario, un grupo o ambas cosas. Por lo tanto, el proceso agente o la hebra de DB2 llamará a las siguientes API de plugins de seguridad:

- db2secDoesGroupExist();
- db2secDoesAuthIDExist();

Desde un servidor para obtener una lista de los grupos a los que pertenece un ID de autorización

Desde el servidor de bases de datos, cuando el usuario necesita obtener una lista de los grupos a los que pertenece un ID de autorización, el proceso agente o la hebra de DB2 llamará a la API de plugins de seguridad utilizando solamente el ID de autorización como dato de entrada:

- `db2secGetGroupsForUser();`

No se obtendrán símbolos procedentes de otros plugins de seguridad.

Plugins de seguridad

Para el sistema de base de datos DB2 la autentificación se realiza utilizando *plugins de seguridad*. Un plugin de seguridad es una biblioteca de carga dinámica que proporciona servicios de seguridad de autentificación.

El sistema de base de datos DB2 proporciona los tipos siguientes de plugins:

- Plugin de recuperación de grupos: recupera información sobre pertenencia a grupos para un usuario determinado.
- Plugin de autentificación de cliente: gestiona la autentificación en un cliente DB2.
- Plugin de autentificación de servidor: gestiona la autentificación en un servidor DB2.

DB2 admite dos mecanismos para la autentificación por plugin:

Autentificación por ID de usuario/contraseña

Este mecanismo utiliza un ID de usuario y contraseña para realizar la autentificación. Los plugins de autentificación por ID de usuario/contraseña aplican los tipos siguientes de autentificación:

- CLIENT
- SERVER
- SERVER_ENCRYPT
- DATA_ENCRYPT
- DATA_ENCRYPT_CMP

Estos tipos de autentificación determinan cómo y dónde se autentifica un usuario. El tipo de autentificación utilizado es el especificado en el parámetro de configuración *authentication* del gestor de bases de datos. Si se especifica el parámetro SRVCON_AUTH, tiene prioridad sobre el parámetro AUTHENTICATION para las operaciones de conexión, física o lógica.

Autentificación GSS-API

La autentificación GSS-API se conoce formalmente como *Generic Security Service Application Program Interface, Versión 2* (IETF RFC2743) y *Generic Security Service API Versión 2: C-Bindings* (IETF RFC2744). La autentificación Kerberos también se implementa utilizando GSS-API. Los plugins de autentificación de GSS-API aplican los tipos de autentificación siguientes:

- KERBEROS
- GSSPLUGIN
- KRB_SERVER_ENCRYPT
- GSS_SERVER_ENCRYPT

KRB_SERVER_ENCRYPT y GSS_SERVER_ENCRYPT son compatibles con la autentificación GSS-API y la autentificación por ID de usuario/contraseña, pero la autentificación GSS-API es el tipo de autentificación preferido.

Nota: Los tipos de autentificación determinan cómo y dónde se autentifica un usuario. Para utilizar un tipo de autentificación determinado, actualice el parámetro de configuración authentication del gestor de bases de datos.

Cada plugin se puede utilizar por separado o en combinación con uno o más de los demás plugins. Por ejemplo, puede utilizar solamente un plugin de autentificación de servidor y aceptar los valores por omisión de DB2 para la autentificación de cliente y de grupo. Como alternativa, puede utilizar solamente un plugin de autentificación de grupo o de cliente. La única situación en la que son necesarios tanto un plugin de cliente como un plugin de servidor es para los plugins de autentificación de GSS-API.

La opción por omisión es utilizar un plugin de ID de usuario/contraseña que implementa un mecanismo de autentificación a nivel del sistema operativo. En los releases anteriores, el comportamiento por omisión es utilizar directamente la autentificación a nivel del sistema operativo sin implementar un plugin. Se proporciona soporte Kerberos para el cliente en los sistemas operativos Solaris, AIX, Windows y Linux. Para plataformas Windows, el soporte de Kerberos está habilitado por omisión.

Los sistemas de base de datos DB2 incluyen conjuntos de plugins para la recuperación de grupos, la autentificación por ID de usuario/contraseña y la autentificación Kerberos. La arquitectura del plugin de seguridad le permite personalizar la función de autentificación de cliente y servidor de DB2 desarrollando sus propios plugins o adquiriendo plugins de terceros.

Despliegue de plugins de seguridad en clientes DB2

Los clientes de DB2 pueden utilizar un plugin de grupo, un plugin de autentificación por ID de usuario/contraseña, y negociar un plugin de GSS-API determinado con el servidor DB2. En esta negociación, el cliente examina la lista de plugins de GSS-API implementados del servidor DB2 y busca el primer nombre de plugin de autentificación que coincida con un plugin de autentificación implementado en el cliente. La lista de plugins del servidor se especifica como valor del parámetro de configuración *srvcon_gssplugin_list* del gestor de bases de datos. Este parámetro contiene los nombres de todos los plugins utilizados en el servidor. La figura siguiente muestra la infraestructura del plugin de seguridad en un cliente DB2.

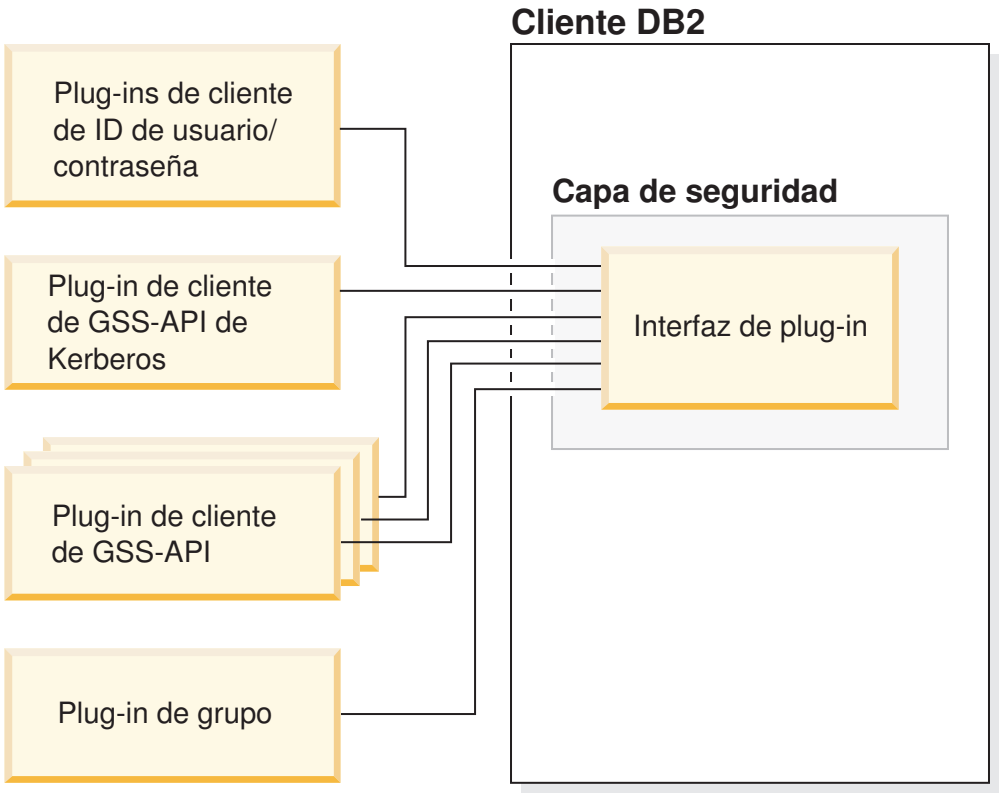


Figura 1. Despliegue de los plugins de seguridad en clientes DB2

Despliegue de plugins de seguridad en servidores DB2

Los servidores DB2 pueden soportar un plugin de grupo, un plugin de autentificación por ID de usuario/contraseña y varios plugins de GSS-API. Los diversos plugins de GSS-API se especifican en una lista como valor del parámetro de configuración *srvcon_gssplugin_list* del gestor de bases de datos. Solamente un único plugin de GSS-API contenido en esa lista puede ser un plugin de Kerberos.

Además de desplegar plugins de seguridad del servidor, puede también necesitar desplegar plugins de autorización del cliente en su servidor de bases de datos. Cuando el usuario ejecuta operaciones a nivel de instancia, tales como db2start y db2trc, el gestor de bases de datos DB2 comprueba la autorización para esas operaciones utilizando los plugins de autentificación del cliente. Por tanto, el usuario debe instalar el plugin de autentificación de cliente correspondiente al plugin de servidor especificado por el parámetro de configuración *authentication* del gestor de bases de datos. La principal diferencia entre *authentication* y *srvcon_auth* consiste en que éstos se pueden establecer en valores diferentes para hacer que se utilice un mecanismo para autentificar las conexiones de base de datos y otro mecanismo para la autorización local. La forma de proceder más habitual es asignar el valor GSSPLUGIN a *srvcon_auth* y asignar el valor SERVER a *authentication*. Si no utiliza plugins de autentificación de cliente en el servidor de bases de datos, las operaciones a nivel de instancia, tales como db2start, fallarán. Por ejemplo, si el tipo de autentificación es SERVER y no se utiliza ningún plugin de cliente proporcionado por el usuario, el sistema de base de datos DB2 utilizará el plugin de cliente por omisión que IBM proporciona, que actúa a nivel del sistema operativo. La figura siguiente muestra la infraestructura del plugin de seguridad en un servidor DB2.

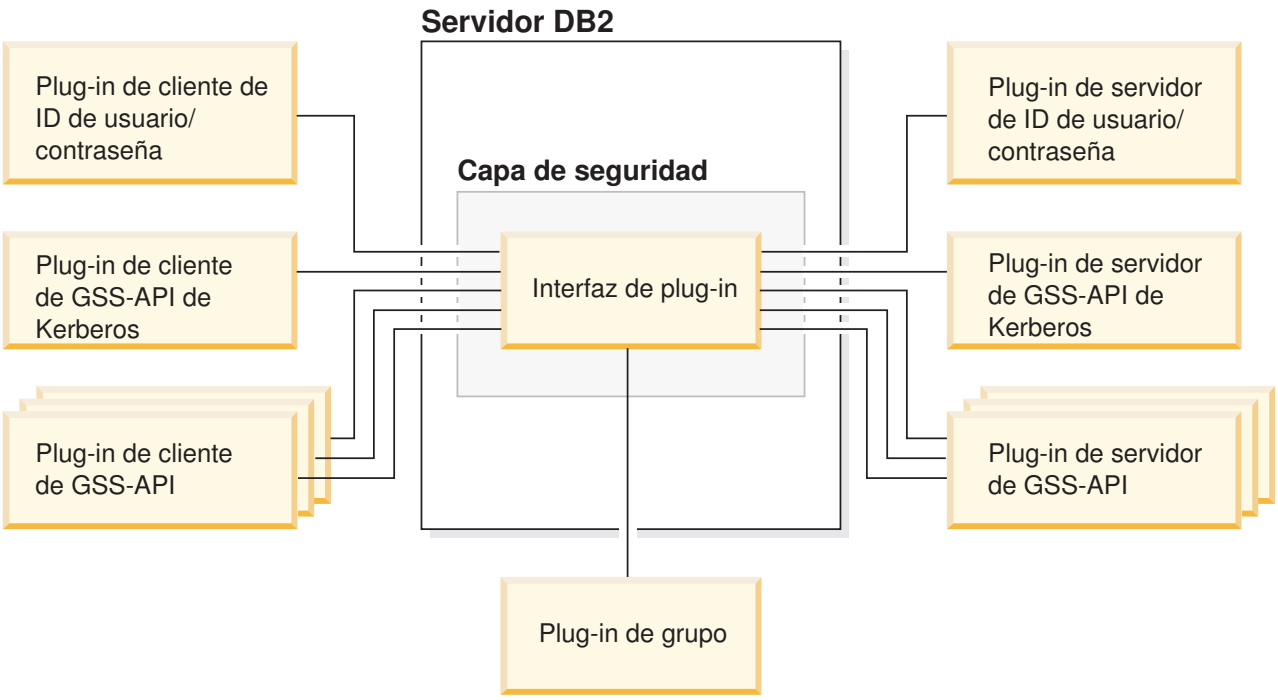


Figura 2. Despliegue de los plugins de seguridad en servidores DB2

Nota: La integridad de la instalación del sistema de base de datos DB2 puede estar en peligro si el despliegue de los plugins de seguridad no se codifica, revisa y prueba debidamente. El sistema de base de datos DB2 toma precauciones frente a muchos tipos de errores habituales, pero no puede garantizar una integridad completa cuando se despliegan plugins de seguridad escritos por el usuario.

Habilitación de los plugins de seguridad

El administrador del sistema puede especificar los nombres de los plugins que se deben utilizar para cada mecanismo de autentificación; para ello actualiza determinados parámetros de configuración del gestor de bases de datos que están asociados a plugins. Si estos parámetros son nulos, se utilizan por omisión los plugins proporcionados por DB2 para la recuperación de grupos, la gestión por ID de usuario/contraseña, o Kerberos (si el parámetro authentication está definido como Kerberos, en el servidor). DB2 no proporciona un plugin de GSS-API por omisión. Por tanto, si el administrador del sistema especifica el tipo de autentificación GSSPLUGIN en el parámetro *authentication*, debe también especificar un plugin de autentificación GSS-API en *srvcon_gssplugin_list*.

Cómo carga DB2 los plugins de seguridad

Todos los plugins admitidos que están identificados por los parámetros de configuración del gestor de bases de datos se cargan cuando se inicia el gestor de bases de datos.

El cliente DB2 carga un plugin apropiado para el mecanismo de seguridad que se negocia con el servidor durante las operaciones de conexión. Una aplicación cliente puede también hacer que se carguen y utilicen simultáneamente varios plugins de seguridad. Esta situación se puede producir, por ejemplo, en un programa multihebra que tiene conexiones simultáneas con diversas bases de datos desde instancias diferentes.

Las acciones que no sean de conexión necesitan también una autorización (tales como actualizar la configuración del gestor de bases de datos, iniciar y detener el gestor de bases de datos, así como activar y desactivar la función de rastreo de DB2). Para tales acciones, el programa cliente de DB2 carga un plugin especificado en otro parámetro de configuración del gestor de bases de datos. Si *authentication* se establece en GSSPLUGIN, el gestor de base de datos de DB2 utilizará el plugin especificado por *local_gssplugin*. Si *authentication* se establece en KERBEROS, el gestor de base de datos de DB2 utilizará el plugin especificado por medio de *clnt_krb_plugin*. En caso contrario, el gestor de base de datos de DB2 utilizará el plugin especificado por medio de *clnt_pw_plugin*.

Las API de plugins de seguridad se pueden invocar desde una plataforma IPv4 o una plataforma IPv6. Una dirección IPv4 es una dirección de 32 bits que tiene el formato legible a.b.c.d, donde cada a-d representa un número decimal del 0 al 255. Una dirección IPv6 es una dirección de 128 bits de la forma a:b:c:d:e:f:g:h, donde cada a-h representa 4 dígitos hexadecimales.

Desarrollo de plugins de seguridad

Si está desarrollando un plugin de seguridad, es necesario que implemente las funciones de autentificación estándar que serán utilizadas por el gestor de bases de datos de DB2. Si está utilizando su propio plugin de seguridad personalizado, puede utilizar un ID de usuario de hasta 255 caracteres en una sentencia CONNECT emitida mediante el CLP (procesador de línea de mandatos) o una sentencia de SQL dinámico. Para los tipos de plugins disponibles, la funcionalidad que necesita implementar es la siguiente:

Recuperación de grupos
Obtiene la lista de grupos a los que pertenece un usuario.

Autentificación por ID de usuario/contraseña

- Identifica el contexto de seguridad por omisión (cliente solamente).
- Valida y opcionalmente cambia una contraseña.
- Determina si una serie proporcionada representa un usuario válido (servidor solamente).
- Modifica el ID de usuario o contraseña proporcionados en el cliente antes de ser enviados al servidor (cliente solamente).
- Devuelve el ID de autorización de DB2 correspondiente a un usuario determinado.

Autentificación GSS-API

- Implementa las funciones de GSS-API necesarias.
- Identifica el contexto de seguridad por omisión (cliente solamente).
- Crea credenciales iniciales de acuerdo con un ID de usuario y contraseña y opcionalmente cambia la contraseña (cliente solamente).
- Crea y acepta certificados de seguridad.
- Devuelve el ID de autorización de DB2 correspondiente a un contexto de seguridad de GSS-API proporcionado.

Ubicaciones de las bibliotecas de plugins de seguridad

Después de obtener sus plugins de seguridad (ya sea desarrollando sus propios plugins o adquiriéndolos de terceros), debe copiarlos en lugares determinados del servidor de bases de datos.

Los clientes DB2 buscan plugins de autentificación de usuario de la parte cliente en el directorio siguiente:

- UNIX de 32 bits: `$DB2PATH/security32/plugin/client`
- UNIX de 64 bits: `$DB2PATH/security64/plugin/client`
- WINDOWS de 32 y 64 bits: `$DB2PATH\security\plugin\`*`nombre de instancia`*`\client`

Nota: En las plataformas Windows, los subdirectorios *nombre de instancia* y *cliente* no se crean automáticamente. El propietario de la instancia debe crearlos manualmente.

El gestor de bases de datos DB2 busca plugins de autentificación del lado del servidor en el directorio siguiente:

- UNIX de 32 bits: `$DB2PATH/security32/plugin/server`
- UNIX de 64 bits: `$DB2PATH/security64/plugin/server`
- WINDOWS de 32 y 64 bits: `$DB2PATH\security\plugin\`*`nombre de instancia`*`\server`

Nota: En las plataformas Windows, los subdirectorios *nombre de instancia* y *server* no se crean automáticamente. El propietario de la instancia debe crearlos manualmente.

El gestor de bases de datos DB2 busca plugins de grupo en el directorio siguiente:

- UNIX de 32 bits: `$DB2PATH/security32/plugin/group`
- UNIX de 64 bits: `$DB2PATH/security64/plugin/group`
- WINDOWS de 32 y 64 bits: `$DB2PATH\security\plugin\`*`nombre de instancia`*`\group`

Nota: En las plataformas Windows, los subdirectorios *nombre de instancia* y *group* no se crean automáticamente. El propietario de la instancia debe crearlos manualmente.

Convenios de denominación para los plugins de seguridad

Las bibliotecas de plugins de seguridad deben tener una extensión de nombre de archivo que es específica de la plataforma. Las bibliotecas de plugins de seguridad escritas en C o C++ deben tener una extensión de nombre de archivo que es específica de la plataforma:

- Windows: `.dll`
- AIX: `.a` o `.so` y si existen ambas extensiones, se utiliza la extensión `.a`.
- Linux, HP IPF y Solaris: `.so`
- HPUX sobre PA-RISC: `.sl` o `.so`, y si existen ambas extensiones, se utiliza la extensión `.sl`.

Nota: Los usuarios pueden también desarrollar plugins de seguridad con el Controlador JDBC universal de DB2.
Por ejemplo, suponga que tiene una biblioteca de plugins de seguridad llamada `MyPlugin`. Para cada sistema operativo soportado, el nombre de archivo apropiado para la biblioteca es el siguiente:

- Windows de 32 bits: MyPlugin.dll
- Windows de 64 bits: MyPlugin64.dll
- AIX de 32 o 64 bits: MyPlugin.a o MyPlugin.so

- SUN de 32 o 64 bits, Linux de 32 o 64 bits, HP de 32 o 64 bits en IPF: MyPlugin.so
- HP-UX de 32 o 64 bits sobre PA-RISC: MyPlugin.sl o MyPlugin.so

Nota: El sufijo ″64″ solo es necesario en el nombre de biblioteca para los plugins de seguridad de Windows de 64 bits.

Cuando actualice la configuración del gestor de bases de datos con el nombre de un plugin de seguridad, utilice el nombre completo de la biblioteca sin el sufijo ″64″ y omita la extensión de archivo y cualquier vía de acceso calificada incluida en el nombre. Para cualquier sistema operativo, una biblioteca de plugins de seguridad llamada `MyPlugin` se registrará de esta manera:

```
UPDATE DBM CFG USING CLNT_PW_PLUGIN MyPlugin
```

El nombre del plugin de seguridad distingue entre mayúsculas y minúsculas, y debe coincidir exactamente con el nombre de biblioteca. Los sistemas de base de datos DB2 utilizan el valor procedente del correspondiente parámetro de configuración del gestor de bases de datos para formar la vía de acceso de la biblioteca y luego utiliza esa vía de acceso para cargar la biblioteca de plugins de seguridad.

Para evitar conflictos en los nombres de los plugins de seguridad, es recomendable que designe el plugin utilizando el método de autentificación empleado y un símbolo identificador de la empresa que escribió el plugin. Por ejemplo, si la empresa `Foo, Inc.` escribió un plugin que hace uso del método de autentificación `FOOsomemethod`, el nombre del plugin podría tener este aspecto: `FOOsomemethod.dll`.

La longitud máxima de un nombre de plugin (sin incluir la extensión de archivo ni el sufijo ″64″) está limitada a 32 bytes. No existe un límite en el número de plugins que pueden ser utilizados por el servidor de bases de datos, pero la longitud máxima de la lista de plugins separados por comas en la configuración del gestor de bases de datos es 255 bytes. Existen dos sentencias DEFINE en el archivo de inclusión `sqlenv.h` que establecen esos dos límites:

```
#define SQL_PLUGIN_NAME_SZ     32     /* nombre del plugin */
#define SQL_SRVCON_GSSPLUGIN_LIST_SZ 255 /* lista de plugins GSS API */
```

Los archivos de las bibliotecas de plugins de seguridad tienen los permisos de archivo siguientes:

- Perteneciente al propietario de la instancia.
- Legible por todos los usuarios del sistema.
- Ejecutable por todos los usuarios del sistema.

Soporte de plugin de seguridad para los ID de usuario de dos componentes

El gestor de bases de datos DB2 para Windows permite utilizar identificadores de usuario de dos componentes y correlacionar los ID de usuario de dos componentes con los ID de autorización de dos componentes.

Por ejemplo, supongamos que un ID de usuario del sistema operativo Windows está formado por dos componentes: un dominio y un ID de usuario como, por ejemplo, `MEDWAY\pieter`. En este ejemplo, `MEDWAY` es un dominio, y `pieter` es el nombre de usuario. En los sistemas de base de datos DB2 puede especificar si este ID de usuario de dos componentes se debe correlacionar con un ID de autorización de un solo componente o de dos componentes.

La correlación de un ID de usuario de dos componentes con un ID de autorización de dos componentes está permitida, pero no es el comportamiento por omisión. Por omisión, los ID de usuario de un solo componente y los ID de usuario de dos componentes se correlacionan con identificadores de autorización de un solo componente. La correlación de un ID de usuario de dos componentes con un ID de autorización de dos componentes está permitida, pero no es el comportamiento por omisión.

La correlación por omisión de un ID de usuario de dos componentes con un ID de usuario de un componente permite que un usuario se conecte a la base de datos utilizando:

```
db2 connect to db user MEDWAY\pieter using pw
```

En este caso, si se utiliza el comportamiento por omisión, el ID de usuario `MEDWAY\pieter` se correlaciona con el ID de autorización `PIETER`. Si está habilitada la correlación de un ID de usuario de dos componentes con un ID de autorización de dos componentes, el ID de autorización sería `MEDWAY\PIETER`.

Para permitir que DB2 correlacione un ID de usuario de dos componentes con un ID de autorización de dos componentes, DB2 proporciona dos conjuntos de plugins de autentificación:

- Un conjunto de plugins correlaciona exclusivamente un ID de usuario de un componente o un ID de usuario de dos componentes con un ID de autorización de un componente.
- Otro conjunto de plugins correlaciona un ID de usuario de un componente o un ID de usuario de dos componentes con un ID de autorización de dos componentes.

Si un nombre de usuario del entorno de trabajo se puede correlacionar con varias cuentas definidas en ubicaciones diferentes (tal como una cuenta local, una cuenta de dominio y cuentas de dominio fiable), puede especificar los plugins que permiten realizar la correlación con los ID de autorización de dos componentes.

Es importante tener en cuenta que un ID de autorización de un solo componente, tal como `PIETER`, y un ID de autorización de dos componentes que resulta de combinar un dominio y un ID de usuario, tal como `MEDWAY\pieter`, son unos ID de autorización funcionalmente distintos. El conjunto de privilegios asociados a uno de estos ID de autorización es completamente diferente del conjunto de privilegios asociados al otro ID de autorización. Debe tener cuidado al trabajar con los ID de autorización de uno y dos componentes.

La tabla siguiente muestra las clases de plugins proporcionadas por los sistemas de base de datos DB2, así como el nombre de los plugins correspondientes a cada implementación de la autentificación.

Tabla 11. Plugins de seguridad de DB2

Tipo de autentificación	Nombre de plugin para ID de usuario de un solo componente	Nombre de plugin para ID de usuario de dos componentes
ID de usuario/contraseña (cliente)	IBMOSauthclient	IBMOSauthclientTwoPart
ID de usuario/contraseña (servidor)	IBMOSauthserver	IBMOSauthserverTwoPart
Kerberos	IBMkrb5	IBMkrb5TwoPart

Nota: En las plataformas Windows de 64 bits, se añaden los caracteres ″64″ a los nombres de los plugins listados aquí.

Cuando especifica un tipo de autentificación que necesita hacer uso de un plugin de ID de usuario/contraseña o plugin Kerberos, se utilizan por omisión los plugins que están listados en la columna ″Nombre de plugin para ID de usuario de un solo componente″ de la tabla anterior.

Para correlacionar un ID de usuario de dos componentes con un ID de autorización de dos componentes, debe especificar que se utilice el plugin de dos componentes, el cual no es el plugin por omisión. Los plugins de seguridad se especifican a nivel de instancia definiendo los parámetros de configuración del gestor de bases de datos referentes a la seguridad, de esta manera:

Para la autentificación de servidor que correlaciona un ID de usuario de dos componentes con un ID de autorización de dos componentes, debe asignar los valores siguientes:

- `srvcon_pw_plugin` a `IBMOSauthserverTwoPart`
- `clnt_pw_plugin` a `IBMOSauthclientTwoPart`

Para la autentificación de cliente que correlaciona un ID de usuario de dos componentes con un ID de autorización de dos componentes, debe asignar los valores siguientes:

- `srvcon_pw_plugin` a `IBMOSauthserverTwoPart`
- `clnt_pw_plugin` a `IBMOSauthclientTwoPart`

Para la autentificación Kerberos que correlaciona un ID de usuario de dos componentes con un ID de autorización de dos componentes, debe asignar los valores siguientes:

- `srvcon_gssplugin_list` a `IBMOSkrb5TwoPart`
- `clnt_krb_plugin` a `IBMkrb5TwoPart`

Las bibliotecas de plugins de seguridad aceptan los ID de usuario de dos componentes que estén especificados en un formato compatible con un Administrador de cuentas de seguridad de Microsoft Windows. Por ejemplo, en el formato: *dominio\ID de usuario*. Durante la conexión, los procesos de autentificación y autorización de DB2 utilizan la información especificada tanto para el dominio como para el UD de usuario.

Puede ser conveniente implementar los plugins de dos componentes al crear nuevas bases de datos para evitar conflictos con los ID de autorización de un solo componente en las bases de datos existentes. Las nuevas bases de datos que hacen uso de los ID de autorización de dos componentes se deben crear en una instancia separada respecto de las bases de datos que hacen uso de los ID de autorización de un solo componente.

Mantenimiento de las versiones de las API de plugins de seguridad

El sistema de base de datos DB2 da soporte a la numeración de las versiones de las API de conectores de seguridad. Estos números de versión son enteros empezando por 1 para DB2 UDB, Versión 8.2.

El número de versión que DB2 pasa a las API de plugins de seguridad es el número de versión más alto de la API que DB2 puede utilizar y corresponde al

número de versión de la estructura. Si el plugin puede utilizar una versión superior de la API, debe devolver punteros de función para la versión que DB2 ha solicitado. Si el plugin solamente es compatible con una versión inferior de la API, el plugin debe proporcionar punteros de función para la versión inferior. En ambos casos, las API de plugins de seguridad deben devolver el número de versión de la API soportada en el campo de versión de la estructura de función.

Para DB2, los números de versión de los plugins de seguridad solamente cambiarán cuando sea necesario (por ejemplo, cuando haya cambios en los parámetros de las API). Los números de versión no cambiarán automáticamente con los números de release de DB2.

Consideraciones sobre los sistemas de 32 y 64 bits para los plugins de seguridad

En general, una instancia de DB2 de 32 bits utilizará el plugin de seguridad de 32 bits y una instancia de DB2 de 64 bits utilizará el plugin de seguridad de 64 bits. Sin embargo, en una instancia de 64 bits, DB2 soporta aplicaciones de 32 bits, las cuales necesitan la biblioteca de plugins de 32 bits.

Una instancia de base de datos en la que se pueden ejecutar tanto aplicaciones de 32 bits como de 64 bits se denomina instancia híbrida. Si dispone de una instancia híbrida y desea ejecutar aplicaciones de 32 bits, asegúrese de que los plugins de seguridad necesarios de 32 bits estén disponibles en el directorio de plugins de 32 bits. Para instancias de DB2 de 64 bits en los sistemas operativos Linux y UNIX, excluido Linux en IPF, aparecen los directorios `security32` y `security64`. Para una instancia de DB2 de 64 bits en Windows en X64 o IPF, tanto el plugin de seguridad de 32 bits como el de 64 bits se encuentran en el mismo directorio, pero el nombre de los plugins de 64 bits tiene el sufijo ″64″.

Si desea migrar desde una instancia de 32 bits a una instancia de 64 bits, debe obtener versiones de los plugins de seguridad que estén precompiladas para 64 bits.

Si ha adquirido los plugins de seguridad de un proveedor que no proporciona bibliotecas de plugins de 64 bits, puede aplicar una rutina auxiliar de 64 bits que ejecute una aplicación de 32 bits. En este caso, el plugin de seguridad es un programa externo en lugar de ser una biblioteca.

Determinación de problemas para plugins de seguridad

Los problemas producidos en los plugins de seguridad se notifican de dos maneras: mediante errores de SQL y a través del archivo de notificaciones de administración.

Estos son los valores de SQLCODE referentes a los plugins de seguridad:

- SQLCODE -1365: se devuelve cuando se produce un error de plugin durante una operación db2start o db2stop.
- SQLCODE - 1366: se devuelve cuando existe un problema de autorización local.
- SQLCODE - 30082: se devuelve para todos los errores de plugin referentes a la conexión.

El archivo de notificaciones de administración es un recurso útil para depurar y administrar plugins de seguridad. Para ver el archivo de notificaciones de administración en UNIX, examine `sqllib/db2dump/`*`nombre de instancia`*`.nfy`. Para ver el archivo de notificaciones de administración en los sistemas operativos

Windows, utilice la herramienta Visor de sucesos. Para acceder a la herramienta Visor de sucesos, seleccione el botón ″Inicio″ del sistema operativo Windows y seleccione `Configuración -> Panel de control -> Herramientas administrativas -> Visor de sucesos`. Estos son los valores del archivo de notificaciones de administración referentes a los plugins de seguridad:

- 13000: indica que ha fallado una llamada a una API de plugins de seguridad GSS-API y ha devuelto un mensaje de error opcional.

  ```
  SQLT_ADMIN_GSS_API_ERROR (13000)
  El plugin "nombre de plugin" ha recibido el código de error
  "código de error" de la API de GSS "nombre de api de
  gss" y ha devuelto el mensaje de error "mensaje de error"
  ```

- 13001 indica que una llamada a una API de plugins de seguridad de DB2 ha fallado con un error y ha devuelto un mensaje de error opcional.

  ```
  SQLT_ADMIN_PLUGIN_API_ERROR(13001)
  El plugin "nombre de plugin" ha recibido el código de error
  "código de error" de la API de plugins de seguridad de DB2
  "nombre de api de gss" y ha devuelto el mensaje de error
  "mensaje de error"
  ```

- 13002 indica que DB2 no ha podido descargar un plugin.

  ```
  SQLT_ADMIN_PLUGIN_UNLOAD_ERROR (13002)
  No se pueden descargar el plugin "nombre de plugin". No es
  necesaria ninguna acción adicional.
  ```

- 13003: indica un nombre de principal incorrecto.

  ```
  SQLT_ADMIN_INVALID_PRIN_NAME (13003)
  El nombre de principal "nombre de principal" utilizado para
  "nombre de plugin" no es válido. Corrija el nombre de
  principal.
  ```

- 13004: indica que el nombre de plugin no es válido. Los separadores de vía de acceso (″/″ en UNIX y ″\″ en Windows) no están permitidos en el nombre de plugin.

  ```
  SQLT_ADMIN_INVALID_PLGN_NAME (13004)
  El nombre de plugin "nombre de plugin" no es válido.
  Corrija el nombre de plugin.
  ```

- 13005: indica que no se ha podido cargar el plugin de seguridad. Asegúrese de que el plugin esté en el directorio correcto y que estén actualizados los parámetros de configuración apropiados del gestor de bases de datos.

  ```
  SQLT_ADMIN_PLUGIN_LOAD_ERROR (13005)
  No se puede cargar el plugin "nombre de plugin". Verifique
  que el plugin existe y que el directorio donde reside sea correcto.
  ```

- 13006: indica que se ha producido un error inesperado en un plugin de seguridad. Reúna toda la información proporcionada por db2support, si es posible capture un rastreo de db2trc y, a continuación, llame al centro de soporte de IBM para obtener ayuda adicional.

  ```
  SQLT_ADMIN_PLUGIN_UNEXP_ERROR (13006)
  El plugin ha encontrado un error inesperado. Consulte al centro de
  soporte de IBM para obtener ayuda.
  ```

Nota: Si está utilizando plugins de seguridad en un servidor de bases de datos de 64 bits Windows y recibe un error de carga para un plugin de seguridad, consulte los temas sobre consideraciones y convenios de denominación de los plugins de seguridad de 32 bits y 64 bits. La biblioteca de plugins de 64 bits necesita que exista el sufijo ″64″ en el nombre de biblioteca, pero este sufijo no debe aparecer en los parámetros de configuración del gestor de bases de datos correspondientes al plugin de seguridad.

Las API del plugin de seguridad

Para que el usuario pueda personalizar el comportamiento de autentificación y de búsqueda de miembros de grupos del sistema de base de datos DB2, el sistema de base de datos DB2 proporciona API que pueden utilizarse para modificar módulos de plugins existentes o crear módulos de plugins de seguridad nuevos.

Cuando el usuario desarrolla un módulo de plugins de seguridad, es necesario que implemente las funciones estándar de autentificación o búsqueda de miembros de grupos que el gestor de bases de datos DB2 invocará. Para los tres tipos de módulos de plugins disponibles, la funcionalidad que necesita implementar es la siguiente:

Recuperación de grupos
Recupera información sobre pertenencia a grupos para un usuario determinado y determina si una serie dada representa un nombre de grupo válido.

Autentificación por ID de usuario/contraseña
Autentificación que identifica el contexto de seguridad por omisión (cliente solamente), valida y opcionalmente cambia una contraseña, determina si una serie proporcionada representa un usuario válido (servidor solamente), modifica el ID de usuario o contraseña proporcionados en el cliente antes de enviarlos al servidor (cliente solamente) y devuelve el ID de autorización de DB2 asociado a un usuario determinado.

Autentificación GSS-API
Autentificación que implementa las funciones de GSS-API necesarias, identifica el contexto de seguridad por omisión (extremo cliente solamente), genera las credenciales iniciales basándose en un ID de usuario y contraseña y opcionalmente cambia la contraseña (extremo cliente solamente), crea y acepta certificados de seguridad, y devuelve el ID de autorización de DB2 asociado a un contexto de seguridad determinado de GSS-API.

Las definiciones siguientes describen la terminología utilizada en las descripciones de las API de los plugins.

Plugin
Biblioteca de carga dinámica que DB2 cargará para acceder a funciones de autentificación o de búsqueda de miembros de grupos escritas por el usuario.

Autentificación implícita
Conexión a una base de datos sin especificar un ID de usuario ni una contraseña.

Autentificación explícita
Conexión a una base de datos en la que se especifican tanto el ID de usuario como la contraseña.

ID de autorización
ID interno que representa a un individuo o grupo al cual se otorgan autorizaciones y privilegios dentro de la base de datos. Internamente, un ID de autorización de DB2 se convierte a letras mayúsculas y tiene un mínimo de 8 caracteres (con blancos de relleno hasta completar los 8 caracteres). Actualmente,DB2 necesita que los ID de autorización, los ID de

usuario, las contraseñas, los nombres de grupo, los espacios de nombres y los nombres de dominio se puedan representar utilizando el juego de caracteres ASCII de 7 bits.

Autorización local
Autorización que es local respecto del servidor o cliente que realiza la implementación de la autorización, y que comprueba si un usuario está autorizado para ejecutar una acción (que no sea la de conectar con la base de datos)), tal como iniciar y detener el gestor de bases de datos, activar y desactivar la función de rastreo de DB2 o actualizar la configuración del gestor de bases de datos.

Espacio de nombres
Colección o agrupación de usuarios en la que los identificadores de usuarios individuales deben ser exclusivos. Son ejemplos típicos de ello los dominios de Windows y Kerberos. Por ejemplo, dentro del dominio de Windows ″usa.company.com″ todos los nombres de usuario deben ser exclusivos. Por ejemplo, ″user1@usa.company.com″. Sin embargo, un mismo ID de usuario utilizado en otro dominio, tal como ″user1@canada.company.com″, hace referencia a una persona diferente. Un identificador de usuario totalmente calificado incluye un ID de usuario y un espacio de nombres; por ejemplo, ″usuario@nombre.dominio″ o ″dominio\usuario″.

Entrada
Indica que DB2 proporcionará el valor para el parámetro de la API del plugin de seguridad.

Salida Indica que la API del plugin de seguridad proporcionará el valor para el parámetro de la API.

API para plugins de recuperación de grupos

Para el módulo de plugins de recuperación de grupos, debe implementar las API siguientes:

- `db2secGroupPluginInit`

  Nota: La API db2secGroupPluginInit utiliza como entrada un puntero, `*logMessage_fn`, que señala hacia una API con el prototipo siguiente:

  ```
  SQL_API_RC (SQL_API_FN db2secLogMessage) (
  db2int32 level,
  void *data,
  db2int32 length);
  ```

  La API db2secLogMessage permite que el plugin registre mensajes en db2diag.log con fines de depuración o información. Esta API la proporciona el sistema de base de datos DB2 por lo que el usuario no necesita implementarla.
- `db2secPluginTerm`
- `db2secGetGroupsForUser`
- `db2secDoesGroupExist`
- `db2secFreeGroupListMemory`
- `db2secFreeErrormsg`
- La única API que se debe poder resolver externamente es `db2secGroupPluginInit`. Esta API utiliza `void *` como parámetro de entrada, que se debe convertir al tipo:

```
typedef struct db2secGroupFunctions_1
{
db2int32 version;
db2int32 plugintype;
SQL_API_RC ( SQL_API_FN *db2secGetGroupsForUser)
(
const char *authid,
db2int32  authidlen,
const char *userid,
db2int32 useridlen,
const char *usernamespace,
db2int32 usernamespacelen,
db2int32 usernamespacetype,
const char *dbname,
db2int32 dbnamelen,
const void *token,
db2int32 tokentype,
db2int32 location,
const char *authpluginname,
db2int32  authpluginnamelen,
void **grouplist,
db2int32  *numgroups,
char **errormsg,
db2int32  *errormsglen);

SQL_API_RC (SQL_API_FN * db2secDoesGroupExist)
(
const char *groupname,
db2int32  groupnamelen,
char **errormsg,
db2int32  *errormsglen);

SQL_API_RC ( SQL_API_FN *db2secFreeGroupListMemory)
(
void *ptr,
char **errormsg,
db2int32  *errormsglen);

SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)(
char *msgtobefree);

SQL_API_RC (SQL_API_FN * db2secPluginTerm)
(
char **errormsg,
db2int32  *errormsglen);

} db2secGroupFunctions_1;
```

La API `db2secGroupPluginInit` asigna las direcciones para el resto de las funciones disponibles externamente.

Nota: El sufijo _1 indica que la estructura corresponde a la versión 1 de la API. Las versiones de interfaz subsiguientes tendrán la extensión _2, _3 y así sucesivamente.

API db2secDoesGroupExist - Comprobar si existe el grupo

Determina si un authid representa un grupo.

Si el nombre de grupo (groupname) existe, la API debe poder devolver el valor DB2SEC_PLUGIN_OK, para indicar que lo ha conseguido. También debe poder devolver el valor DB2SEC_PLUGIN_INVALIDUSERORGROUP si el nombre del grupo no es válido. Está permitido que la API devuelva el valor DB2SEC_PLUGIN_GROUPSTATUSNOTKNOWN si resulta imposible determinar si

el dato de entrada es un grupo válido. Si se devuelve un valor de grupo no válido (DB2SEC_PLUGIN_INVALIDUSERORGROUP) o de grupo desconocido (DB2SEC_PLUGIN_GROUPSTATUSNOTKNOWN), es posible que DB2 no pueda determinar si el parámetro authid representa un grupo o un usuario al emitir la sentencia GRANT sin las palabras clave USER y GROUP, lo que provocaría la devolución del error SQLCODE -569, SQLSTATE 56092 al usuario.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC (SQL_API_FN * db2secDoesGroupExist)
                      ( const char *groupname,
                        db2int32  groupnamelen,
                        char **errormsg,
                        db2int32 *errormsglen );
```

Parámetros de la API db2secDoesGroupExist

groupname
Entrada. ID de autorización, escrito en mayúsculas y sin blancos de cola.

groupnamelen
Entrada. Longitud, en bytes, del valor del parámetro groupname.

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secDoesGroupExist no es satisfactoria.

errormsglen
Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

API db2secFreeErrormsg - Liberar la memoria de mensajes de error

Libera la memoria que sirve para contener un mensaje de error de una llamada anterior a la API. Esta es la única API que no devuelve un mensaje de error. Si esta API devuelve un error, DB2 lo anotará y continuará el proceso.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)(char *errormsg);
```

Parámetros de la API db2secFreeErrormsg

msgtofree
Entrada. Puntero al mensaje de error procedente de una llamada anterior a la API.

API db2secFreeGroupListMemory - Liberar memoria de lista de grupos

Libera la memoria que sirve para contener la lista de grupos de una anterior llamada a la API db2secGetGroupsForUser.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC ( SQL_API_FN *db2secFreeGroupListMemory)
                      ( void *ptr,
                        char **errormsg,
                        db2int32 *errormsglen );
```

Parámetros de la API db2secFreeGroupListMemory

ptr Entrada. Puntero a la memoria que se debe liberar.

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secFreeGroupListMemory no es satisfactoria.

errormsglen
Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

API db2secGetGroupsForUser - Obtener la lista de grupos del usuario

Devuelve la lista de grupos a los que pertenece un usuario.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC ( SQL_API_FN *db2secGetGroupsForUser)
                       ( const char *authid,
                         db2int32 authidlen,
                         const char *userid,
                         db2int32 useridlen,
                         const char *usernamespace,
                         db2int32 usernamespacelen,
                         db2int32 usernamespacetype,
                         const char *dbname,
                         db2int32 dbnamelen,
                         void *token,
                         db2int32 tokentype,
                         db2int32 location,
                         const char *authpluginname,
                         db2int32  authpluginnamelen,
                         void **grouplist,
                         db2int32  *numgroups,
                         char **errormsg,
                         db2int32 *errormsglen );
```

Parámetros de la API db2secGetGroupsForUser

authid Entrada. El valor de este parámetro es un authid SQL, es decir, que DB2 lo convierte en una serie en mayúsculas sin blancos de cola. DB2 siempre proporcionará un valor no nulo para el parámetro authid. La API debe poder devolver una lista de los grupos a los que pertenece el authid sin depender de los demás parámetros de entrada. Está permitido devolver una lista abreviada o vacía si esa información no se puede determinar.

Si un usuario no existe, la API debe devolver el código de retorno DB2SEC_PLUGIN_BADUSER. DB2 no trata como un error el caso de que un usuario no exista, pues está permitido que un authid no tenga grupos asociados al mismo. Por ejemplo, la API db2secGetAuthids puede devolver un authid que no exista en el sistema operativo. El authid no está asociado a ningún grupo, pero se le pueden seguir asignando privilegios directamente.

Si la API no puede devolver una lista de grupos completa utilizando solo el authid, habrá algunas restricciones en determinadas funciones SQL relacionadas con el soporte de grupos. Para obtener una lista de las posibles situaciones problemáticas, consulte la sección de notas de uso, en este tema.

authidlen
Entrada. Longitud, en bytes, del valor del parámetro authid. El gestor de bases de datos DB2 siempre proporciona un valor distinto de cero para el parámetro authidlen.

userid Entrada. Es el ID de usuario correspondiente al parámetro authid. Cuando se llama a esta API en el servidor en una situación de falta de conexión, DB2 no rellenará este parámetro.

useridlen
Entrada. Longitud, en bytes, del valor del parámetro userid.

usernamespace
Entrada. Espacio de nombres del que se obtuvo el ID de usuario. Cuando el ID de usuario no está disponible, el gestor de bases de datos DB2 no rellenará este parámetro.

usernamespacelen
Entrada. Longitud, en bytes, del valor del parámetro usernamespace.

usernamespacetype
Entrada. Es el tipo del espacio de nombres. Los valores válidos para el parámetro usernamespacetype (definido en db2secPlugin.h) son:

- DB2SEC_NAMESPACE_SAM_COMPATIBLE Se corresponde con un nombre de usuario del estilo dominio\minombre
- DB2SEC_NAMESPACE_USER_PRINCIPAL Se corresponde con un nombre de usuario del estilo minombre@dominio.ibm.com

Actualmente, el sistema de base de datos DB2 sólo da soporte al valor DB2SEC_NAMESPACE_SAM_COMPATIBLE. Cuando el ID de usuario no está disponible, el parámetro usernamespacetype se establece en el valor DB2SEC_USER_NAMESPACE_UNDEFINED (definido en db2secPlugin.h).

dbname
Entrada. Nombre de la base de datos con la que se establece conexión. Este parámetro puede ser NULL en una situación de falta de conexión.

dbnamelen
Entrada. Longitud, en bytes, del valor del parámetro dbname. Este parámetro se establece en 0 si el parámetro dbname es NULL en una situación de falta de conexión.

token Entrada. Puntero a los datos proporcionados por el plugin de autentificación. DB2 no lo utiliza. Proporciona al autor del plugin la capacidad de coordinar la información de usuario y de grupo. Este parámetro no se proporciona en todos los casos (por ejemplo, en una situación de falta de conexión), en cuyo caso el valor será NULL. Si el plugin de autentificación utilizado está basado en GSS-API, el valor del símbolo será igual al descriptor de contexto de GSS-API (gss_ctx_id_t).

tokentype
Entrada. Indica el tipo de datos proporcionado por el plugin de autentificación. Si el plugin de autentificación utilizado está basado en GSS-API, el valor del símbolo será igual al descriptor de contexto de GSS-API (gss_ctx_id_t). Si el plugin de autentificación está basado en el ID de usuario/contraseña, el tipo de datos será genérico. Los valores válidos del parámetro tokentype (definido en db2secPlugin.h) son:

- `DB2SEC_GENERIC`: Indica que el símbolo procede de un plugin basado en un ID de usuario/contraseña.

- DB2SEC_GSSAPI_CTX_HANDLE: Indica que el símbolo procede de un plugin basado en GSS-API (incluido Kerberos).

location
Entrada. Indica si DB2 llama a esta API en el extremo del cliente o del servidor. Los valores válidos del parámetro location (definido en db2secPlugin.h) son:

- DB2SEC_SERVER_SIDE: Se llamará a la API desde el servidor de base de datos.
- DB2SEC_CLIENT_SIDE: Se llamará a la API desde un cliente.

authpluginname
Entrada. Nombre del plugin de autentificación que proporcionó los datos contenidos en el símbolo. La API db2secGetGroupsForUser puede utilizar esta información al determinar las pertenencias a grupo correctas. Es posible que DB2 no rellene este parámetro si authid no se autentifica (por ejemplo, si el valor de authid no coincide con el usuario actual conectado).

authpluginnamelen
Entrada. Longitud, en bytes, del valor del parámetro authpluginname.

grouplist
Salida. Lista de grupos a los que pertenece el usuario. La lista de grupos se debe devolver en forma de puntero a una sección de memoria asignada por el plugin que contiene varchars concatenadas (varchar es una matriz de caracteres en la que el primer byte indica el número de bytes que la siguen). La longitud es un carácter sin signo (1 byte) y limita la longitud máxima de un nombre de grupo a 255 caracteres. Por ejemplo, "\006GROUP1\007MYGROUP\008MYGROUP3". Cada nombre de grupo debe ser un authid válido de DB2. La memoria para esta matriz debe estar asignada por el plugin. Por lo tanto, el plugin debe proporcionar una API, como la API db2secFreeGroupListMemory, a la que DB2 llamará para liberar la memoria.

numgroups
Salida. Número de grupos contenidos en el parámetro grouplist.

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secGetGroupsForUser no es satisfactoria.

errormsglen
Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

Notas de uso

A continuación se muestra una lista de situaciones en las que se pueden producir problemas si esta API devuelve una lista de grupos incompleta a DB2:

- Aplicación de SQL incorporado con DYNAMICRULES BIND (o DEFINEDBIND o INVOKEDBIND si los paquetes se ejecutan como una aplicación autónoma). DB2 comprueba si existe la pertenencia a SYSADM y la aplicación fallará si depende de la autorización DBADM implícita otorgada por el hecho de ser miembro del grupo SYSADM.

- Se proporciona una autorización alternativa en la sentencia CREATE SCHEMA. La búsqueda de grupos se efectuará respecto al parámetro AUTHORIZATION NAME si existen sentencias CREATE anidadas en la sentencia CREATE SCHEMA.
- Aplicaciones de SQL incorporado con DYNAMICRULES DEFINERUN/ DEFINEBIND y paquetes que se ejecutan en un contexto de rutina. DB2 comprueba si existe la pertenencia a SYSADM del definidor de la rutina y la aplicación fallará si depende de la autorización DBADM implícita otorgada por el hecho de ser miembro del grupo SYSADM.
- Proceso de un archivo jar en un entorno MPP. En un entorno MPP, la petición de proceso jar se envía desde el nodo coordinador con el authid de sesión. El nodo de catálogo recibe las peticiones y procesa los archivos jar basándose en el privilegio del authid de sesión (el usuario que ejecuta las peticiones de proceso de jar).
  - Instalar archivo jar. El authid de sesión debe tener uno de los siguientes derechos: SYSADM, DBADM o CREATEIN (implícito o explícito sobre el esquema jar). La operación falla si los derechos anteriores se otorgan a un grupo que contiene el authid de sesión, pero no explícitamente al authid de sesión o si solo se tiene SYSADM, dado que la pertenencia SYSADM viene determinada por la pertenencia contenida en el grupo definido por un parámetro de configuración de base de datos.
  - Eliminar archivo jar. El authid de sesión debe tener uno de los siguientes derechos: SYSADM, DBADM o DROPIN (implícito o explícito sobre el esquema jar) o bien es el definidor del archivo jar. La operación falla si los derechos anteriores se otorgan a un grupo que contiene el authid de sesión, pero no explícitamente al authid de sesión, y si el authid de sesión no corresponde al definidor del archivo jar o si solo se tiene SYSADM, dado que la pertenencia SYSADM viene determinada por la pertenencia contenida en el grupo definido por un parámetro de configuración de base de datos.
  - Sustituir archivo jar. Equivale a eliminar el archivo jar y luego instalar el archivo jar. Son aplicables las dos situaciones anteriores.
- Regenerar vistas. Se desencadena mediante las sentencias ALTER TABLE, ALTER COLUMN, SET DATA TYPE VARCHAR/VARGRAPHIC o durante migración. El gestor de bases de datos DB2 comprueba si existe la pertenencia a SYSADM del definidor de la vista. La aplicación fallará si depende de la autorización DBADM implícita otorgada por el hecho de ser miembro del grupo SYSADM.
- Cuando se emite la sentencia SET SESSION_USER. Las operaciones de DB2 ulteriores se ejecutan bajo el contexto del authid especificado por esta sentencia. Estas operaciones fallan si los privilegios necesarios que son propiedad de uno del grupo de usuarios con SESSION_USER no se otorgan explícitamente al authid de SESSION_USER.

API db2secGroupPluginInit - Inicializar plugin de grupo

API de inicialización, para el plugin de recuperación de grupo, a la que el gestor de bases de datos DB2 llama inmediatamente después de cargar el plugin.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN db2secGroupPluginInit
                            ( db2int32 version,
                              void *group_fns,
                              db2secLogMessage *logMessage_fn,
                              char **errormsg,
                              db2int32 *errormsglen );
```

Parámetros de la API db2secGroupPluginInit

version

Entrada. Versión más alta de la API soportada por la instancia que carga ese plugin. El valor DB2SEC_API_VERSION (en db2secPlugin.h) contiene el número de versión más reciente de la API a la que el gestor de bases de datos DB2 proporciona soporte actualmente.

group_fns

Salida. Puntero a la estructura db2secGroupFunctions_<número_versión> (que también se conoce como group_functions_<número_versión>). La estructura db2secGroupFunctions_<número_versión> contiene punteros a las API implementadas para el plugin de recuperación de grupo. En el futuro pueden existir diferentes versiones de las API (por ejemplo, db2secGroupFunctions_<número_versión>), por lo que el parámetro group_fns se convierte en un puntero a la estructura db2secGroupFunctions_<número_versión> correspondiente a la versión implementada por el plugin. El primer parámetro de la estructura group_functions_<número_versión> indica a DB2 la versión de las API que el plugin ha implementado. Nota: la conversión temporal solo se realiza si la versión de DB2 es mayor que o igual a la versión de las API que el plugin ha implementado. El número de versión representa la versión de las API que el plugin ha implementado, y el valor de pluginType debe ser igual a DB2SEC_PLUGIN_TYPE_GROUP.

logMessage_fn

Entrada. Puntero a la API db2secLogMessage, implementada por el sistema de base de datos DB2. La API db2secGroupPluginInit puede llamar a la API db2secLogMessage para anotar mensajes en db2diag.log con fines de depuración o informativos. El primer parámetro (level) de la API db2secLogMessage especifica el tipo de errores de diagnóstico que se anotarán en el archivo db2diag.log, y los dos últimos parámetros son, respectivamente, la serie del mensaje y su longitud. Los valores válidos del primer parámetro de la API dbesecLogMessage (definida en db2secPlugin.h) son:

- `DB2SEC_LOG_NONE`: (0) No se ha anotado nada
- `DB2SEC_LOG_CRITICAL`: (1) Se ha encontrado un error grave
- `DB2SEC_LOG_ERROR`: (2) Se ha encontrado un error
- `DB2SEC_LOG_WARNING`: (3) Aviso
- `DB2SEC_LOG_INFO`: (4) Informativo

El texto del mensaje solo aparecerá en el archivo diag.log si el valor del parámetro 'level' de la API db2secLogMessage es menor o igual que el parámetro diaglevel de configuración del gestor de bases de datos. Por lo tanto, si utiliza, por ejemplo, el valor `DB2SEC_LOG_INFO`, el texto del mensaje solo aparecerá en el archivo db2diag.log si el parámetro diaglevel de configuración del gestor de bases de datos se establece en 4.

errormsg

Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secGroupPluginInit no es satisfactoria.

errormsglen

Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

db2secPluginTerm - Liberar los recursos de plugin de grupo

Libera los recursos utilizados por el plugin de recuperación de grupo.

El gestor de bases de datos DB2 llama a esta API justo antes de descargar el plugin de recuperación de grupo. Se debe implementar de tal manera que haga una limpieza adecuada de los recursos que contenga la biblioteca de plugins; por ejemplo, liberar la memoria que el plugin haya asignado, cerrar los archivos que aún estén abiertos y cerrar las conexiones de red. El plugin se encarga de hacer un seguimiento de estos recursos para poder liberarlos. No se llama a esta API en ninguna plataforma Windows.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC (SQL_API_FN * db2secPluginTerm)
                       ( char     **errormsg,
                         db2int32 *errormsglen );
```

Parámetros de la API db2secPluginTerm

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secPluginTerm no es satisfactoria.

errormsglen
Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

Las API de los plugins de autentificación por ID de usuario/contraseña

Para el módulo de plugins de ID de usuario/contraseña, debe implementar las API siguientes en el extremo cliente:

- `db2secClientAuthPluginInit`

  Nota: La API db2secClientAuthPluginInit utiliza como entrada un puntero, `*logMessage_fn`, que señala hacia una API con el prototipo siguiente:

  ```
  SQL_API_RC (SQL_API_FN db2secLogMessage) (
  db2int32 level,
  void *data,
  db2int32 length);
  ```

  La API db2secLogMessage permite que el plugin registre mensajes en db2diag.log con fines de depuración o información. Esta API la proporciona el sistema de base de datos DB2 por lo que el usuario no necesita implementarla.
- `db2secClientAuthPluginTerm`
- `db2secGenerateInitialCred` (Utilizado solamente para gssapi)
- `db2secRemapUserid` (Opcional)
- `db2secGetDefaultLoginContext`
- `db2secValidatePassword`
- `db2secProcessServerPrincipalName` (Solamente para GSS-API)
- `db2secFreeToken` (Funciones para liberar memoria retenida por la DLL)
- `db2secFreeErrormsg`
- `db2secFreeInitInfo`

- La única API que se debe poder resolver externamente es `db2secClientAuthPluginInit`. Esta API utiliza `void *` como parámetro de entrada, que se debe convertir a uno de estos dos tipos:

```
typedef struct db2secUseridPasswordClientAuthFunctions_1
{
db2int32 version;
db2int32 plugintype;

SQL_API_RC ( SQL_API_FN *db2secGetDefaultLoginContext)
(
char authid[DB2SEC_MAX_AUTHID_LENGTH],
db2int32  *authidlen,
char userid[DB2SEC_MAX_USERID_LENGTH],
db2int32  *useridlen,
db2int32 useridtype,
char usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
db2int32  *usernamespacelen,
db2int32  *usernamespacetype,
const char *dbname,
db2int32 dbnamelen,
void      **token,
char **errormsg,
db2int32 *errormsglen);
/* Opcional */
SQL_API_RC (SQL_API_FN * db2secRemapUserid)
(
char userid[DB2SEC_MAX_USERID_LENGTH],
db2int32  *useridlen,
char usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
db2int32  *usernamespacelen,
db2int32  *usernamespacetype,
char password[DB2SEC_MAX_PASSWORD_LENGTH],
db2int32  *passwordlen,
char newpassword[DB2SEC_MAX_PASSWORD_LENGTH],
db2int32  *newpasswordlen,
const char *dbname,
db2int32 dbnamelen,
char **errormsg,
db2int32 *errormsglen);

SQL_API_RC ( SQL_API_FN *db2secValidatePassword)
(
const char *userid,
db2int32 useridlen,
const char *usernamespace,
db2int32 usernamespacelen,
db2int32 usernamespacetype,
const char *password,
db2int32 passwordlen,
const char *newpassword,
db2int32 newpasswordlen,
const char *dbname,
db2int32 dbnamelen,
db2Uint32  connection_details,
void      **token,
char **errormsg,
db2int32 *errormsglen);

SQL_API_RC ( SQL_API_FN *db2secFreeToken)
(
void **token,
char **errormsg,
db2int32 *errormsglen);

SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)(char *errormsg);
```

```
SQL_API_RC ( SQL_API_FN *db2secClientAuthPluginTerm)
(
char **errormsg,
db2int32 *errormsglen);
}
```

o bien

```
typedef struct db2secGssapiClientAuthFunctions_1
{
db2int32 version;
db2int32 plugintype;

SQL_API_RC ( SQL_API_FN *db2secGetDefaultLoginContext)
(
char authid[DB2SEC_MAX_AUTHID_LENGTH],
db2int32  *authidlen,
char userid[DB2SEC_MAX_USERID_LENGTH],
db2int32  *useridlen,
db2int32 useridtype,
char usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
db2int32  *usernamespacelen,
db2int32  *usernamespacetype,
const char *dbname,
db2int32 dbnamelen,
void      **token,
char **errormsg,
db2int32 *errormsglen);

SQL_API_RC ( SQL_API_FN *db2secProcessServerPrincipalName)
(
const void *data,
gss_name_t *gssName,
char **errormsg,
db2int32 *errormsglen);

SQL_API_RC ( SQL_API_FN *db2secGenerateInitialCred)
(
const char *userid,
db2int32 useridlen,
const char *usernamespace,
db2int32 usernamespacelen,
db2int32 usernamespacetype,
const char *password,
db2int32 passwordlen,
const char *newpassword,
db2int32 newpasswordlen,
const char *dbname,
db2int32 dbnamelen,
gss_cred_id_t *pGSSCredHandle,
void **initInfo,
char **errormsg,
db2int32 *errormsglen);

SQL_API_RC ( SQL_API_FN *db2secFreeToken)
(
void *token,
char **errormsg,
db2int32 *errormsglen);

SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)(char *errormsg);


SQL_API_RC (SQL_API_FN * db2secFreeInitInfo) (
void *initInfo,
char **errormsg,
db2int32 *errormsglen);
```

```
SQL_API_RC ( SQL_API_FN *db2secClientAuthPluginTerm)
(
char **errormsg,
db2int32 *errormsglen);

/* Funciones específicas de GSS-API  -- consulte
  db2secPlugin.h  para obtener la lista de parámetros */

  OM_uint32 (SQL_API_FN * gss_init_sec_context )(<lista de parámetros>);
  OM_uint32 (SQL_API_FN * gss_delete_sec_context )(<lista de parámetros>);
  OM_uint32 (SQL_API_FN * gss_display_status )(<lista de parámetros>);
  OM_uint32 (SQL_API_FN * gss_release_buffer )(<lista de parámetros>);
  OM_uint32 (SQL_API_FN * gss_release_cred )(<lista de parámetros>);
  OM_uint32 (SQL_API_FN * gss_release_name )(<lista de parámetros>);
  }
```

Debe utilizar la estructura db2secUseridPasswordClientAuthFunctions_1 si está escribiendo un plugin de ID de usuario/contraseña. Debe utilizar la estructura db2secGssapiClientAuthFunctions_1 si está escribiendo un plugin de GSS_API (incluido Kerberos).

Para la biblioteca de plugins de autentificación por ID de usuario/contraseña, debe implementar las API siguientes en el extremo servidor:

- db2secServerAuthPluginInit

  La API db2secServerAuthPluginInit utiliza como entrada un puntero, *logMessage_fn, que señala hacia la API db2secLogMessage, y un puntero, *getConDetails_fn, que señala hacia la API db2secGetConDetails con los prototipos siguientes:

```
SQL_API_RC (SQL_API_FN db2secLogMessage) (
db2int32 level,
void *data,
db2int32 length);

SQL_API_RC (SQL_API_FN db2secGetConDetails)(
db2int32 conDetailsVersion,
const void *pConDetails);
```

  La API db2secLogMessage permite que el plugin registre mensajes en db2diag.log con fines de depuración o información. La API db2secGetConDetails permite que el plugin obtenga detalles sobre el cliente que intenta establecer una conexión de base de datos. Tanto la API db2secLogMessage como la API db2secGetConDetails las proporciona el sistema de base de datos DB2 por lo que el usuario no necesita implementarlas. A su vez, la API db2secGetConDetails utiliza como segundo parámetro de entrada el puntero pConDetails, que señala una de las siguientes estructuras:

  db2sec_con_details_1:

```
typedef struct db2sec_con_details_1
{
  db2int32 clientProtocol;
  db2Uint32 clientIPAddress;
  db2Uint32 connect_info_bitmap;
  db2int32 dbnameLen;
  char dbname[DB2SEC_MAX_DBNAME_LENGTH + 1];
} db2sec_con_details_1;
```

  db2sec_con_details_2:

```
typedef struct db2sec_con_details_2
{
  db2int32  clientProtocol;     /* Véase SQL_PROTOCOL_ en sqlenv.h */
  db2Uint32 clientIPAddress;    /* Se define si el protocolo es TCPIP4 */
  db2Uint32 connect_info_bitmap;
```

```
  db2int32 dbnameLen;
  char dbname[DB2SEC_MAX_DBNAME_LENGTH + 1];
  db2Uint32 clientIP6Address[4];/* Se define si el protocolo es TCPIP6 */
} db2sec_con_details_2;
```

db2sec_con_details_3:

```
typedef struct db2sec_con_details_3
{
  db2int32  clientProtocol;     /* Véase SQL_PROTOCOL_ en sqlenv.h */
  db2Uint32 clientIPAddress;    /* Se define si el protocolo es TCPIP4 */
  db2Uint32 connect_info_bitmap;
  db2int32 dbnameLen;
  char dbname[DB2SEC_MAX_DBNAME_LENGTH + 1];
  db2Uint32 clientIP6Address[4];/* Se define si el protocolo es TCPIP6 */
  db2Uint32 clientPlatform; /* SQLM_PLATFORM_* de sqlmon.h */
  db2Uint32 _reserved[16];
} db2sec_con_details_3;
```

Los valores posibles de `conDetailsVersion` son DB2SEC_CON_DETAILS_VERSION_1, DB2SEC_CON_DETAILS_VERSION_2 y DB2SEC_CON_DETAILS_VERSION_3 que representan la versión de la API.

Nota: Cuando utilice db2sec_con_details_1, db2sec_con_details_2 o db2sec_con_details_3, tenga en cuenta lo siguiente:

- Los plugins existentes que hacen uso de la estructura db2sec_con_details_1 y el valor DB2SEC_CON_DETAILS_VERSION_1 continúan funcionando de la misma forma en que lo hacían en la versión 8.2 al invocar la API db2GetConDetails. Si esta API se invoca en una plataforma IPv4, se devuelve la dirección IP del cliente en el campo clientIPAddress de la estructura. Si esta API se invoca en una plataforma IPv6, se devuelve el valor 0 en el campo clientIPAddress. Para recuperar la dirección IP del cliente en una plataforma IPv6, se debe cambiar el código del plugin de seguridad para utilizar la estructura db2sec_con_details_2 structure y el valor DB2SEC_CON_DETAILS_VERSION_2, o la estructura db2sec_con_details_3 y el valor DB2SEC_CON_DETAILS_VERSION_3.
- Los nuevos plugins deben utilizar la estructura db2sec_con_details_3 y el valor DB2SEC_CON_DETAILS_VERSION_3. Si la API db2secGetConDetails se invoca en una plataforma IPv4, se devuelve la dirección IP del cliente en el campo clientIPAddress de la estructura db2sec_con_details_3; si la API se invoca en una plataforma IPv6, se devuelve la dirección IP del cliente en el campo clientIP6Address de la estructura db2sec_con_details_3. El campo *clientProtocol* de la estructura de detalles de la conexión se establece en uno de estos valores: SQL_PROTOCOL_TCPIP (IPv4, con v1 de la estructura), SQL_PROTOCOL_TCPIP4 (IPv4, con v2 de la estructura) o SQL_PROTOCOL_TCPIP6 (IPv6, con v2 o v3 de la estructura).
- La estructura db2sec_con_details_3 es idéntica a la estructura db2sec_con_details_2 exceptuando que esta contiene un campo adicional (*clientPlatform*) que identifica el tipo de plataforma del cliente (tal como ha informado la capa de comunicaciones) mediante la utilización de las constantes de tipo de plataforma definidas en sqlmon.h, como por ejemplo SQLM_PLATFORM_AIX.

- `db2secServerAuthPluginTerm`
- `db2secValidatePassword`
- `db2secGetAuthIDs`
- `db2secDoesAuthIDExist`
- `db2secFreeToken`
- `db2secFreeErrormsg`

- La única API que se debe poder resolver externamente es db2secServerAuthPluginInit. Esta API utiliza void * como parámetro de entrada, que se debe convertir a uno de estos dos tipos:

```
typedef struct db2secUseridPasswordServerAuthFunctions_1
{
db2int32 version;
db2int32 plugintype;

  /* Por razones de legibilidad, las listas de parámetros se han
     dejado en blanco; vea más arriba para conocer los parámetros */
SQL_API_RC (SQL_API_FN * db2secValidatePassword)(<lista de parámetros>);
SQL_API_RC (SQL_API_FN * db2secGetAuthIDs)(<lista de parámetros);
SQL_API_RC (SQL_API_FN * db2secDoesAuthIDExist)(<lista de parámetros>);
SQL_API_RC (SQL_API_FN * db2secFreeToken)(<lista de parámetros>);
SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)(<lista de parámetros>);
SQL_API_RC (SQL_API_FN * db2secServerAuthPluginTerm)();
} userid_password_server_auth_functions;
```

o bien

```
typedef struct db2secGssapiServerAuthFunctions_1
{
db2int32 version;
db2int32 plugintype;
gss_buffer_desc serverPrincipalName;
gss_cred_id_t ServerCredHandle;
SQL_API_RC (SQL_API_FN * db2secGetAuthIDs)(<lista de parámetros);
SQL_API_RC (SQL_API_FN * db2secDoesAuthIDExist)(<lista de parámetros>);
SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)(<lista de parámetros>);
SQL_API_RC (SQL_API_FN * db2secServerAuthPluginTerm)();

/* Funciones específicas de GSS-API;
consulte db2secPlugin.h para obtener la lista de parámetros*/
OM_uint32 (SQL_API_FN * gss_accept_sec_context )(<lista de parámetros>);
OM_uint32 (SQL_API_FN * gss_display_name )(<lista de parámetros>);
OM_uint32 (SQL_API_FN * gss_delete_sec_context )(<lista de parámetros>);
OM_uint32 (SQL_API_FN * gss_display_status )(<lista de parámetros>);
OM_uint32 (SQL_API_FN * gss_release_buffer )(<lista de parámetros>);
OM_uint32 (SQL_API_FN * gss_release_cred )(<lista de parámetros>);
OM_uint32 (SQL_API_FN * gss_release_name )(<lista de parámetros>);

} gssapi_server_auth_functions;
```

Debe utilizar la estructura db2secUseridPasswordServerAuthFunctions_1 si está escribiendo un plugin de ID de usuario/contraseña. Debe utilizar la estructura db2secGssapiServerAuthFunctions_1 si está escribiendo un plugin de GSS_API (incluido Kerberos).

API db2secClientAuthPluginInit - Inicializar el plugin de autentificación del cliente

API de inicialización, para el plugin de autentificación de cliente, a la que el gestor de bases de datos DB2 llama inmediatamente después de cargar el plugin.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN db2secClientAuthPluginInit
                            ( db2int32 version,
                              void *client_fns,
                              db2secLogMessage *logMessage_fn,
                              char **errormsg,
                              db2int32 *errormsglen );
```

Parámetros de la API db2secClientAuthPluginInit

version

Entrada. Número de versión más alto de la API que se puede utilizar actualmente en DB2. El valor DB2SEC_API_VERSION (en db2secPlugin.h) contiene el número de versión más reciente de la API que se puede utilizar actualmente en DB2.

client_fns

Salida. Puntero a la memoria proporcionado por el gestor de bases de datos DB2 para una estructura db2secGssapiClientAuthFunctions_<número_versión> (también conocida como gssapi_client_auth_functions_<número_versión>), si se utiliza la autentificación GSS-API o una estructura db2secUseridPasswordClientAuthFunctions_<número_versión> (también conocida como userid_password_client_auth_functions_<número_versión>), si se utiliza la autentificación por ID de usuario/contraseña. La estructura db2secGssapiClientAuthFunctions_<número_versión> y la estructura db2secUseridPasswordClientAuthFunctions_<número_versión> contienen respectivamente punteros que señalan hacia las API implementadas para el plugin de autentificación por GSS-API y para el plugin de autentificación por ID de usuario/contraseña. En las futuras versiones de DB2, pueden existir versiones diferentes de las API, por lo que el parámetro client_fns se convierte en un puntero a la estructura gssapi_client_auth_functions_<número_versión> correspondiente a la versión que el plugin ha implementado.

El primer parámetro de la estructura gssapi_client_auth_functions_<número_versión> o de la estructura userid_password_client_auth_functions_<número_versión> indica al gestor de bases de datos DB2 la versión de las API que el plugin ha implementado.

Nota: La conversión sólo se realiza si la versión de DB2 es mayor que o igual a la versión de las API que el plugin ha implementado.

Dentro de la estructura gssapi_server_auth_functions_<número_versión> o userid_password_server_auth_functions_<número_versión>, el parámetro plugintype debe tener uno de estos valores: DB2SEC_PLUGIN_TYPE_USERID_PASSWORD, DB2SEC_PLUGIN_TYPE_GSSAPI o DB2SEC_PLUGIN_TYPE_KERBEROS. Podrán definirse otros valores en versiones futuras de la API.

logMessage_fn

Entrada. Puntero a la API db2secLogMessage, implementada por el gestor de bases de datos DB2. La API db2secClientAuthPluginInit puede llamar a la API db2secLogMessage para anotar mensajes en db2diag.log con fines de depuración o informativos. El primer parámetro (level) de la API db2secLogMessage especifica el tipo de errores de diagnóstico que se anotarán en el archivo db2diag.log y los dos últimos parámetros son, respectivamente, la serie del mensaje y su longitud. Los valores válidos del primer parámetro de la API dbesecLogMessage (definida en db2secPlugin.h) son:

- DB2SEC_LOG_NONE (0) No se ha anotado nada
- DB2SEC_LOG_CRITICAL (1) Se ha encontrado un error grave
- DB2SEC_LOG_ERROR (2) Se ha encontrado un error

- DB2SEC_LOG_WARNING (3) Aviso
- DB2SEC_LOG_INFO (4) Informativo

El texto del mensaje solo aparecerá en el archivo db2diag.log si el valor del parámetro 'level' de la API db2secLogMessage es menor que o igual al parámetro diaglevel de configuración del gestor de bases de datos. Por ejemplo, si se utiliza el valor DB2SEC_LOG_INFO, el texto del mensaje sólo aparecerá en el archivo db2diag.log si el parámetro diaglevel de configuración del gestor de bases de datos está establecido en 4.

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secClientAuthPluginInit no es satisfactoria.

errormsglen
Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

API db2secClientAuthPluginTerm - Liberar los recursos de plugin de autentificación de cliente

Libera los recursos utilizados por el plugin de autentificación de cliente.

El gestor de bases de datos DB2 llama a esta API justo antes de descargar el plugin de autentificación de cliente. Se debe implementar de tal manera que haga una limpieza adecuada de los recursos que contenga la biblioteca de plugins; por ejemplo, liberar la memoria que el plugin haya asignado, cerrar los archivos que aún estén abiertos y cerrar las conexiones de red. El plugin se encarga de hacer un seguimiento de estos recursos para poder liberarlos. No se llama a esta API en ninguna plataforma Windows.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC ( SQL_API_FN *db2secClientAuthPluginTerm)
                        ( char     **errormsg,
                          db2int32  *errormsglen);
```

Parámetros de la API db2secClientAuthPluginTerm

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secClientAuthPluginTerm no es satisfactoria.

errormsglen
Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

db2secDoesAuthIDExist - Comprobar si existe el ID de autentificación

Determina si el parámetro authid representa un usuario individual (por ejemplo, si la API puede correlacionar el parámetro authid con un ID de usuario externo).

La API debe devolver el valor DB2SEC_PLUGIN_OK si es satisfactoria, o sea, si el authid es válido, el valor DB2SEC_PLUGIN_INVALID_USERORGROUP si no es válido o el valor DB2SEC_PLUGIN_USERSTATUSNOTKNOWN si no se puede determinar la existencia de authid.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC ( SQL_API_FN *db2secDoesAuthIDExist)
                      ( const char *authid,
                        db2int32 authidlen,
                        char **errormsg,
                        db2int32 *errormsglen );
```

Parámetros de la API db2secDoesAuthIDExist

authid Entrada. Parámetro authid que hay que validar. Es un valor escrito en mayúsculas, sin blancos de cola.

authidlen
Entrada. Longitud, en bytes, del valor del parámetro authid.

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secDoesAuthIDExist no es satisfactoria.

errormsglen
Salida. Puntero a un valor entero que indica la longitud de la serie del mensaje de error en el parámetro errormsg.

API db2secFreeInitInfo - Liberar los recursos retenidos por la API db2secGenerateInitialCred

Libera los recursos que se hayan asignado mediante la API db2secGenerateInitialCred. Esto puede incluir, por ejemplo, descriptores de contexto de mecanismos subyacentes o una antememoria de credenciales creada para la antememoria de credenciales GSS-API.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC ( SQL_API_FN *db2secFreeInitInfo)
                      ( void *initinfo,
                        char **errormsg,
                        db2int32  *errormsglen);
```

Parámetros de la API db2secFreeInitInfo

initinfo
Entrada. Puntero a los datos desconocidos para el gestor de bases de datos DB2. El plugin puede utilizar esta memoria para mantener una lista de recursos que se asignan durante el proceso de generación del descriptor de credencial. Estos recursos se liberan llamando a esta API.

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secFreeInitInfo no es satisfactoria.

errormsglen
Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

API db2secFreeToken - Liberar memoria retenida por símbolo (token)

Libera la memoria retenida por un símbolo (token). El gestor de bases de datos DB2 llama a esta API cuando ya no necesita la memoria retenida por el parámetro token.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC ( SQL_API_FN *db2secFreeToken)
                      ( void *token,
                        char **errormsg,
                        db2int32 *errormsglen );
```

Parámetros de la API db2secFreeToken

token Entrada. Puntero a la memoria que se debe liberar.

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secFreeToken no es satisfactoria.

errormsglen
Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

API db2secGenerateInitialCred - Generar credenciales iniciales

Obtiene las credenciales iniciales GSS-API basándose en el ID de usuario y contraseña que se pasen como parámetros. Para Kerberos, se trata del certificado de otorgamiento de certificados (TGT). El descriptor de credencial que se devuelve en el parámetro pGSSCredHandle es el descriptor de contexto utilizado con la API gss_init_sec_context y debe ser una credencial para INITIATE o para BOTH. Solo se llama a la API db2secGenerateInitialCred cuando se suministra un ID de usuario y, posiblemente, una contraseña. De lo contrario, el gestor de bases de datos DB2 especifica el valor GSS_C_NO_CREDENTIAL al llamar a la API gss_init_sec_context API para indicar que debe utilizarse la credencial por omisión que se obtiene a partir del contexto de inicio de sesión actual.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC ( SQL_API_FN *db2secGenerateInitialCred)
                      ( const char *userid,
                        db2int32 useridlen,
                        const char *usernamespace,
                        db2int32 usernamespacelen,
                        db2int32 usernamespacetype,
                        const char *password,
                        db2int32 passwordlen,
                        const char *newpassword,
                        db2int32 newpasswordlen,
                        const char *dbname,
                        db2int32 dbnamelen,
                        gss_cred_id_t *pGSSCredHandle,
                        void          **InitInfo,
                        char **errormsg,
                        db2int32 *errormsglen );
```

Parámetros de la API db2secGenerateInitialCred

userid Entrada. ID de usuario cuya contraseña hay que verificar en el servidor de base de datos.

useridlen
Entrada. Longitud, en bytes, del valor del parámetro userid.

usernamespace
Entrada. Espacio de nombres del que se obtuvo el ID de usuario.

usernamespacelen
Entrada. Longitud, en bytes, del valor del parámetro usernamespace.

usernamespacetype
Entrada. Es el tipo del espacio de nombres.

password
Entrada. Contraseña que se debe verificar.

passwordlen
Entrada. Longitud, en bytes, del valor del parámetro password.

newpassword
Entrada. Contraseña nueva, si la contraseña se debe cambiar. Si no hace falta cambiarla, el parámetro newpassword se establece en NULL. Si este parámetro no es NULL,la API debe validar la contraseña antigua antes de establecer el nuevo valor de la contraseña. La API no está obligada a respetar una petición de cambio de contraseña, pero si no lo hace, debe volver inmediatamente con el valor de retorno DB2SEC_PLUGIN_CHANGEPASSWORD_NOTSUPPORTED sin validar la antigua contraseña.

newpasswordlen
Entrada. Longitud, en bytes, del valor del parámetro newpassword.

dbname
Entrada. Nombre de la base de datos con la que se establece conexión. La API puede hacer caso omiso de este parámetro o bien devolver el valor DB2SEC_PLUGIN_CONNECTION_DISALLOWED si tiene una política de restringir el acceso a determinadas bases de datos a usuarios que, de otra manera, tienen contraseñas válidas.

dbnamelen
Entrada. Longitud, en bytes, del valor del parámetro dbname.

pGSSCredHandle
Salida. Puntero al descriptor de credencial de GSS-API.

InitInfo
Salida. Puntero a los datos desconocidos para DB2. El plugin puede utilizar esta memoria para mantener una lista de recursos que se asignan durante el proceso de generación del descriptor de credencial. El gestor de bases de datos DB2 llama a la API db2secFreeInitInfo al final del proceso de autentificación, momento en que se liberan estos recursos. Si la API db2secGenerateInitialCred no necesita la lista en cuestión, devolverá el valor NULL.

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secGenerateInitialCred no es satisfactoria.

Nota: Para esta API, no se deben crear mensajes de error si el valor de retorno indica la existencia de un ID de usuario o contraseña incorrectos. El mensaje de error solo se debe devolver si en la API existe un error interno que le impida finalizar debidamente su proceso.

errormsglen
Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

API db2secGetAuthIDs - Obtener los ID de autentificación

Devuelve un parámetro SQL authid de un usuario autentificado. Esta API se invoca durante las conexiones a base de datos para los métodos de autentificación por ID de usuario/contraseña y por GSS-API.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC ( SQL_API_FN *db2secGetAuthIDs)
                       ( const char *userid,
                         db2int32 useridlen,
                         const char *usernamespace,
                         db2int32 usernamespacelen,
                         db2int32 usernamespacetype,
                         const char *dbname,
                         db2int32 dbnamelen,
                         void      **token,
                         char SystemAuthID[DB2SEC_MAX_AUTHID_LENGTH],
                         db2int32 *SystemAuthIDlen,
                         char InitialSessionAuthID[DB2SEC_MAX_AUTHID_LENGTH],
                         db2int32 *InitialSessionAuthIDlen,
                         char username[DB2SEC_MAX_USERID_LENGTH],
                         db2int32 *usernamelen,
                         db2int32 *initsessionidtype,
                         char **errormsg,
                         db2int32 *errormsglen );
```

Parámetros de la API db2secGetAuthIDs

userid Entrada. El usuario autentificado. No suele utilizarse para la autentificación GSS-API a menos que se haya permitido un contexto de confianza para permitir las operaciones de conmutación de usuario sin autentificación. En estas situaciones, en este parámetro se pasa el nombre de usuario proporcionado para la solicitud de conmutación de usuario.

useridlen
Entrada. Longitud, en bytes, del valor del parámetro userid.

usernamespace
Entrada. Espacio de nombres del que se obtuvo el ID de usuario.

usernamespacelen
Entrada. Longitud, en bytes, del valor del parámetro usernamespace.

usernamespacetype
Entrada. Valor de namespacetype. Actualmente el único valor soportado para el tipo de espacio de nombres es DB2SEC_NAMESPACE_SAM_COMPATIBLE (que se corresponde con un estilo de nombre de usuario como, por ejemplo, dominio\minombre).

dbname
Entrada. Nombre de la base de datos con la que se establece conexión. La API puede hacer caso omiso de este parámetro o devolver valores de authid diferentes cuando un mismo usuario se conecta a distintas bases de datos. Este parámetro puede ser NULL.

dbnamelen
Entrada. Longitud, en bytes, del valor del parámetro dbname. Este parámetro se establece igual a 0 si el parámetro dbname es NULL.

token Entrada o salida. Datos que el plugin puede pasar a la API db2secGetGroupsForUser. Para GSS-API, esto es un descriptor de contexto (gss_ctx_id_t). Normalmente, el parámetro token es solo de entrada y su valor se obtiene de la API db2secValidatePassword. También puede ser un

parámetro de salida cuando la autentificación se realiza en el cliente y, por lo tanto, no se llama a la API db2secValidatePassword. En entornos en los que se haya definido un entorno de confianza que permita las operaciones de conmutación de usuario sin autentificación, la API db2secGetAuthIDs debe poder dar cabida a la recepción de un valor NULL para este parámetro de símbolo así como derivar un ID de autorización del sistema basado en los parámetros de entrada userid y useridlen anteriores.

SystemAuthID
Salida. ID de autorización del sistema que se corresponde con el ID del usuario autentificado. El tamaño es de 255 bytes, pero el gestor de bases de datos DB2 solo utiliza actualmente un máximo de 30 (inclusive).

SystemAuthIDlen
Salida. Longitud, en bytes, del valor del parámetro SystemAuthID.

InitialSessionAuthID
Salida. Authid utilizado para esta sesión de conexión. Suele ser igual que el parámetro SystemAuthID pero puede ser distinto en algunas situaciones, por ejemplo cuando se emite una sentencia SET SESSION AUTHORIZATION. El tamaño es de 255 bytes, pero el gestor de bases de datos DB2 solo utiliza actualmente un máximo de 30 (inclusive).

InitialSessionAuthIDlen
Salida. Longitud, en bytes, del valor del parámetro InitialSessionAuthID.

username
Salida. Nombre correspondiente al usuario autentificado y al valor de authid. Este nombre solo se utiliza con fines de auditoría y se anota en el campo ″ID de usuario″ del registro de auditoría de la sentencia CONNECT. Si la API no rellena el parámetro username, el gestor de bases de datos DB2 lo copia a partir del parámetro userid.

usernamelen
Salida. Longitud, en bytes, del valor del parámetro username.

initsessionidtype
Salida. Tipo de authid de sesión que indica si el parámetro InitialSessionAuthid es un rol o un authid. La API debe devolver uno de los siguientes valores (definidos en db2secPlugin.h):

- DB2SEC_ID_TYPE_AUTHID (0)
- DB2SEC_ID_TYPE_ROLE (1)

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secGetAuthIDs no es satisfactoria.

errormsglen
Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

API db2secGetDefaultLoginContext - Obtener contexto de conexión por omisión

Determina el usuario asociado al contexto de conexión por omisión, es decir, determina el ID de autorización de DB2 del usuario que invoca un mandato de DB2 sin especificar explícitamente un ID de usuario (ya sea una autentificación implícita ante una base de datos o una autorización local). Esta API debe devolver un ID de autorización y un ID de usuario.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC ( SQL_API_FN *db2secGetDefaultLoginContext)
                      ( char authid[DB2SEC_MAX_AUTHID_LENGTH],
                        db2int32  *authidlen,
                        char userid[DB2SEC_MAX_USERID_LENGTH],
                        db2int32  *useridlen,
                        db2int32 useridtype,
                        char usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
                        db2int32  *usernamespacelen,
                        db2int32  *usernamespacetype,
                        const char *dbname,
                        db2int32 dbnamelen,
                        void **token,
                        char **errormsg,
                        db2int32 *errormsglen );
```

Parámetros de la API db2secGetDefaultLoginContext

authid Salida. Parámetro en el que se debe devolver el ID de autorización. El valor devuelto debe cumplir las reglas de denominación para los ID de autorización de DB2; de lo contrario el usuario no tendrá autorización para ejecutar la acción solicitada.

authidlen
Salida. Longitud, en bytes, del valor del parámetro authid.

userid Salida. Parámetro en el que se debe devolver el ID de usuario asociado al contexto de conexión por omisión.

useridlen
Salida. Longitud, en bytes, del valor del parámetro userid.

useridtype
Entrada. Indica si se está especificando el ID de usuario real o efectivo del proceso. En Windows, sólo existe el ID de usuario real. en UNIX y Linux, el ID de usuario real y el ID de usuario eficaz pueden ser diferentes si el ID de usuario `uid` de la aplicación es diferente del ID de usuario que ejecuta el proceso. Los valores válidos del parámetro userid (definido en db2secPlugin.h) son:

DB2SEC_PLUGIN_REAL_USER_NAME
Indica que se especifica el ID de usuario real.

DB2SEC_PLUGIN_EFFECTIVE_USER_NAME
Indica que se especifica el ID de usuario efectivo.

Nota: En algunas implementaciones de plugins, no se distingue entre el id de usuario real del efectivo. En concreto, un plugin que no utilice la identidad UNIX o Linux del usuario para establecer el ID de autorización de DB2 puede hacer caso omiso de esta distinción con total seguridad.

usernamespace
Salida. Espacio de nombres del ID de usuario.

usernamespacelen
Salida. Longitud, en bytes, del valor del parámetro usernamespace. De acuerdo con la limitación de que el parámetro usernamespacetype se debe establecer igual al valor DB2SEC_NAMESPACE_SAM_COMPATIBLE (definido en db2secPlugin.h), la longitud máxima que se puede usar actualmente es de 15 bytes.

usernamespacetype
Salida. Valor de namespacetype. Actualmente el único tipo de espacio de nombres soportado es DB2SEC_NAMESPACE_SAM_COMPATIBLE (que se corresponde con un estilo de nombre de usuario como por ejemplo dominio\minombre).

dbname
Entrada. Contiene el nombre de la base de datos con la que se establece conexión, si la llamada se utiliza en el contexto de una conexión de base de datos. Para acciones de autorización local o conexiones de instancia, este parámetro se establece igual a NULL.

dbnamelen
Entrada. Longitud, en bytes, del valor del parámetro dbname.

token Salida. Puntero a los datos asignados por el plugin para pasarlos a llamadas de autentificación subsiguientes del plugin, o posiblemente al plugin de recuperación de grupos. La estructura de estos datos la debe determinar la persona que escribe el plugin.

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secGetDefaultLoginContext no es satisfactoria.

errormsglen
Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

API db2secProcessServerPrincipalName - Procesar nombre de principal de servicio devuelto desde servidor

Procesa el nombre de sujeto principal de servicio devuelto desde el servidor y devuelve el nombre de sujeto principal con el formato interno de gss_name_t que hay que usar con la API gss_init_sec_context. La API db2secProcessServerPrincipalName también procesa el nombre de sujeto principal de servicio catalogado con el directorio de base de datos cuando se utiliza la autentificación de Kerberos. Por lo general, esta conversión utiliza la API gss_import_name. Una vez establecido el contexto, el objeto gss_name_t se libera mediante la llamada a la API gss_release_name. La API db2secProcessServerPrincipalName devuelve el valor DB2SEC_PLUGIN_OK si el parámetro gssName señala hacia un nombre GSS válido; Se devuelve un código de error DB2SEC_PLUGIN_BAD_PRINCIPAL_NAME si el nombre del sujeto principal no es válido.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC ( SQL_API_FN *db2secProcessServerPrincipalName)
                      ( const char *name,
                        db2int32 namelen,
                        gss_name_t *gssName,
                        char **errormsg,
                        db2int32 *errormsglen );
```

Parámetros de la API db2secProcessServerPrincipalName

name Entrada. Nombre del principal de servicio, en formato GSS_C_NT_USER_NAME; por ejemplo, service/host@REALM.

namelen
Entrada. Longitud, en bytes, del valor del parámetro name.

gssName
Salida. Puntero al nombre de sujeto principal de servicio de salida con el formato interno de GSS-API.

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secProcessServerPrincipalName no es satisfactoria.

errormsglen
Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

API db2secRemapUserid - Volver a correlacionar el ID de usuario y la contraseña

El gestor de bases de datos DB2 llama a esta API en el extremo cliente para volver a correlacionar un ID de usuario y una contraseña en concreto (y posiblemente la nueva contraseña y el espacio de nombres de usuario) con valores distintos a los proporcionados en el momento de la conexión. El gestor de bases de datos DB2 sólo llama a esta API si se suministra un ID de usuario y una contraseña en el momento de la conexión. Ello impide que un plugin pueda volver a correlacionar por sí mismo un ID de usuario con un par ID de usuario/contraseña. Esta API es opcional y la llamada no se realiza si el plugin de seguridad no la proporciona o implementa.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC (SQL_API_FN * db2secRemapUserid)
                      ( char userid[DB2SEC_MAX_USERID_LENGTH],
                        db2int32  *useridlen,
                        char usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
                        db2int32  *usernamespacelen,
                        db2int32  *usernamespacetype,
                        char password[DB2SEC_MAX_PASSWORD_LENGTH],
                        db2int32  *passwordlen,
                        char newpasswd[DB2SEC_MAX_PASSWORD_LENGTH],
                        db2int32 *newpasswdlen,
                        const char *dbname,
                        db2int32 dbnamelen,
                        char **errormsg,
                        db2int32  *errormsglen);
```

Parámetros de la API db2secRemapUserid

userid Entrada o salida. ID de usuario que se tiene que volver a correlacionar. Si existe un valor de ID de usuario de entrada, la API debe proporcionar un valor de ID de usuario de salida que pueda coincidir o no con el valor del ID de usuario de entrada. Si no hay ningún valor de ID de usuario de entrada, la API no debe devolver un valor de ID de usuario de salida.

useridlen
Entrada o salida. Longitud, en bytes, del valor del parámetro userid.

usernamespace
Entrada o salida. Espacio de nombres del ID de usuario. Este valor se puede volver a correlacionar opcionalmente. Si no se especifica ningún valor de parámetro de entrada pero se devuelve un valor de salida, el gestor de bases de datos DB2 sólo utiliza el parámetro usernamespace para la autentificación de tipo CLIENT y no se tiene en cuenta para los otros tipos de autentificación.

usernamespacelen
Entrada o salida. Longitud, en bytes, del valor del parámetro usernamespace. De acuerdo con la limitación de que el parámetro usernamespacetype se debe establecer igual al valor DB2SEC_NAMESPACE_SAM_COMPATIBLE (definido en db2secPlugin.h), la longitud máxima que se puede usar actualmente es de 15 bytes.

usernamespacetype
Entrada o salida. Valor del parámetro namespacetype nuevo y antiguo. Actualmente el único valor soportado para el tipo de espacio de nombres es DB2SEC_NAMESPACE_SAM_COMPATIBLE (que se corresponde con un estilo de nombre de usuario como, por ejemplo, dominio\minombre).

password
Entrada o salida. Como parámetro de entrada, es la contraseña que hay que volver a correlacionar. Como parámetro de salida, es la contraseña recorrelacionada. Si se especifica un valor de entrada para este parámetro, la API debe poder devolver un valor de salida que sea distinto al valor de entrada. Si no se especifica ningún valor de entrada, la API no debe devolver un valor de contraseña de salida.

passwordlen
Entrada o salida. Longitud, en bytes, del valor del parámetro password.

newpasswd
Entrada o salida. Como parámetro de entrada, es la nueva contraseña que hay que definir. Como parámetro de salida, es la nueva contraseña confirmada.

Nota: Esta es la nueva contraseña que el gestor de bases de datos DB2 pasa en el parámetro newpassword de la API db2secValidatePassword en el cliente o en el servidor (en función del valor del parámetro de configuración de la autenticación del gestor de bases de datos). Si la nueva contraseña se ha pasado como parámetro de entrada, la API debe poder devolver un valor de salida y puede ser una nueva contraseña distinta. Si no se pasa ninguna nueva contraseña como parámetro de entrada, la API no debe devolver una nueva contraseña de salida.

newpasswdlen
Entrada o salida. Longitud, en bytes, del valor del parámetro newpasswd.

dbname
Entrada. Nombre de la base de datos a la que se conecta el cliente.

dbnamelen
Entrada. Longitud, en bytes, del valor del parámetro dbname.

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secRemapUserid no es satisfactoria.

errormsglen
Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

db2secServerAuthPluginInit - Inicializar el plugin de autentificación del servidor

API de inicialización, para el plugin de autentificación de servidor, a la que el gestor de bases de datos DB2 llama inmediatamente después de cargar el plugin.

En el caso de GSS-API, el plugin se encarga de rellenar el nombre de sujeto principal del servidor en el parámetro serverPrincipalName dentro de la estructura gssapi_server_auth_functions en el momento de la inicialización y que proporciona el descriptor de contexto de credencial del servidor en el parámetro serverCredHandle dentro de la estructura gssapi_server_auth_functions. La tarea de liberar la memoria asignada para contener el nombre de sujeto principal y el descriptor de contexto de credencial la debe realizar la API db2secServerAuthPluginTerm, llamando a las API gss_release_name y gss_release_cred.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC SQL_API_FN db2secServerAuthPluginInit
                              ( db2int32 version,
                                void *server_fns,
                                db2secGetConDetails *getConDetails_fn,
                                db2secLogMessage *logMessage_fn,
                                char **errormsg,
                                db2int32 *errormsglen );
```

Parámetros de la API db2secServerAuthPluginInit

version

Entrada. Número de versión más alto de la API que se puede utilizar actualmente en DB2. El valor DB2SEC_API_VERSION (en db2secPlugin.h) contiene el número de versión más reciente de la API a la que el gestor de bases de datos DB2 proporciona soporte actualmente.

server_fns

Salida. Puntero a la memoria proporcionado por el gestor de bases de datos DB2 para una estructura db2secGssapiServerAuthFunctions_<número_versión> (también conocida como gssapi_server_auth_functions_<número_versión>), si se utiliza la autentificación GSS-API o para una estructura db2secUseridPasswordServerAuthFunctions_<número_versión> (también conocida como userid_password_server_auth_functions_<número-versión>), si se utiliza la autentificación por ID de usuario/contraseña. La estructura db2secGssapiServerAuthFunctions_<número_versión> y la estructura db2secUseridPasswordServerAuthFunctions_<número_versión> contienen respectivamente punteros que señalan hacia las API implementadas para el plugin de autentificación por GSS-API y para el plugin de autentificación por ID de usuario/contraseña.

El parámetro server_fns se convierte en un puntero a la estructura gssapi_server_auth_functions_<número_versión> correspondiente a la versión que el plugin ha implementado. El primer parámetro de la estructura gssapi_server_auth_functions_<número_versión> o de la estructura userid_password_server_auth_functions_<número_versión> indica al gestor de bases de datos DB2 la versión de las API que el plugin ha implementado.

Nota: La conversión sólo se realiza si la versión de DB2 es mayor que o igual a la versión de las API que el plugin ha implementado.

Dentro de la estructura gssapi_server_auth_functions_<número_versión> o userid_password_server_auth_functions_<número_versión>, el parámetro plugintype debe tener uno de estos valores: DB2SEC_PLUGIN_TYPE_USERID_PASSWORD, DB2SEC_PLUGIN_TYPE_GSSAPI o DB2SEC_PLUGIN_TYPE_KERBEROS. Podrán definirse otros valores en versiones futuras de la API.

getConDetails_fn

Entrada. Puntero a la API db2secGetConDetails API, implementada por DB2. La API db2secServerAuthPluginInit puede llamar a la API db2secGetConDetails en cualquiera de las demás API de autentificación para obtener detalles relacionados con la conexión de base de datos. Estos detalles incluyen información sobre el mecanismo de comunicación asociado a la conexión (como la dirección IP, en el caso de TCP/IP), que el creador del plugin puede tener que especificar al definir la autentificación. Por ejemplo, el plugin podría rechazar una conexión para un determinado usuario, a menos que ese usuario se conecte desde una dirección IP determinada. El uso de la API db2secGetConDetails es opcional.

Si se llama a la API db2secGetConDetails en una situación que no implique una conexión de base de datos, la API devuelve el valor DB2SEC_PLUGIN_NO_CON_DETAILS; de lo contrario, devuelve 0 tras un resultado satisfactorio.

La API db2secGetConDetails tiene dos parámetros de entrada; pConDetails, que es un puntero a la estructura db2sec_con_details_<número_versión>, y conDetailsVersion, que es un número de versión que indica qué estructura db2sec_con_details se debe utilizar. Los valores posibles son DB2SEC_CON_DETAILS_VERSION_1, cuando se utiliza db2sec_con_details1, o DB2SEC_CON_DETAILS_VERSION_2, cuando se utiliza db2sec_con_details2. El número de versión recomendado es DB2SEC_CON_DETAILS_VERSION_2.

Si el retorno es satisfactorio, la estructura db2sec_con_details (ya sea db2sec_con_details1 o db2sec_con_details2) contendrá la siguiente información:

- El protocolo utilizado para la conexión con el servidor. La lista de definiciones de protocolo se encuentra en el archivo sqlenv.h (situado en el directorio include) (SQL_PROTOCOL_*). Esta información se rellena en el parámetro clientProtocol.
- La dirección TCP/IP de la conexión entrante con el servidor si el valor del parámetro clientProtocol es SQL_PROTOCOL_TCPIP o SQL_PROTOCOL_TCPIP4. Esta información se rellena en el parámetro clientIPAddress.
- El nombre de la base de datos con la que el cliente intenta establecer conexión. Este nombre no se define para conexiones físicas de instancia. Esta información se rellena en los parámetros dbname y dbnameLen.
- Un mapa de bits de información de conexión que contiene los mismos detalles que los documentados en el parámetro connection_details de la API db2secValidatePassword. Esta información se rellena en el parámetro connect_info_bitmap.
- La dirección TCP/IP de la conexión entrante con el servidor si el valor del parámetro clientProtocol es SQL_PROTOCOL_TCPIP6. Esta información se rellena en el parámetro clientIP6Address y solo está disponible si se utiliza DB2SEC_CON_DETAILS_VERSION_2 para la llamada a la API db2secGetConDetails.

logMessage_fn

Entrada. Puntero a la API db2secLogMessage, implementada por el gestor de bases de datos DB2. La API db2secClientAuthPluginInit puede llamar a la API db2secLogMessage para anotar mensajes en db2diag.log con fines de depuración o informativos. El primer parámetro (level) de la API

db2secLogMessage especifica el tipo de errores de diagnóstico que se anotarán en el archivo db2diag.log y los dos últimos parámetros son, respectivamente, la serie del mensaje y su longitud. Los valores válidos del primer parámetro de la API dbesecLogMessage (definida en db2secPlugin.h) son:

DB2SEC_LOG_NONE (0)
No se ha anotado nada

DB2SEC_LOG_CRITICAL (1)
Se ha encontrado un error grave

DB2SEC_LOG_ERROR (2)
Se ha encontrado un error

DB2SEC_LOG_WARNING (3)
Aviso

DB2SEC_LOG_INFO (4)
Informativo

El texto del mensaje solo aparecerá en el archivo db2diag.log si el valor del parámetro 'level' de la API db2secLogMessage es menor que o igual al parámetro diaglevel de configuración del gestor de bases de datos.

Por lo tanto, si utiliza, por ejemplo, el valor DB2SEC_LOG_INFO, el texto del mensaje solo aparecerá en el archivo db2diag.log si el parámetro diaglevel de configuración del gestor de bases de datos tiene el valor 4.

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secServerAuthPluginInit no es satisfactoria.

errormsglen
Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

API db2secServerAuthPluginTerm - Liberar los recursos de plugin de autentificación de servidor

Libera los recursos utilizados por el plugin de autentificación de servidor. El gestor de bases de datos DB2 llama a esta API justo antes de descargar el plugin de autentificación de servidor. Se debe implementar de tal manera que haga una limpieza adecuada de los recursos que contenga la biblioteca de plugins; por ejemplo, liberar la memoria que el plugin haya asignado, cerrar los archivos que aún estén abiertos y cerrar las conexiones de red. El plugin se encarga de hacer un seguimiento de estos recursos para poder liberarlos. No se llama a esta API en ninguna plataforma Windows.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC ( SQL_API_FN *db2secServerAuthPluginTerm)
                        ( char     **errormsg,
                          db2int32 *errormsglen );
```

Parámetros de la API db2secServerAuthPluginTerm

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secServerAuthPluginTerm no es satisfactoria.

errormsglen
Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

API db2secValidatePassword - Validar contraseña

Proporciona un método para realizar la autentificación por ID de usuario y contraseña durante una operación de conexión de base de datos.

Nota: Cuando la API se ejecuta en el extremo cliente, el código de la API se ejecuta con privilegios del usuario que ejecuta la sentencia CONNECT. Solo se llamará a esta API en el extremo cliente si el parámetro de configuración de autentificación se establece igual a CLIENT.

Cuando la API se ejecuta en el extremo servidor, el código de la API se ejecuta con privilegios del propietario de la instancia.

El autor del plugin debe tener en cuenta los puntos anteriores si la autentificación necesita privilegios especiales (como el acceso al sistema a nivel de root en UNIX).

Esta API debe devolver el valor DB2SEC_PLUGIN_OK (satisfactorio) si la contraseña es válida o devolver un código de error como DB2SEC_PLUGIN_BADPWD si la contraseña no es válida.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC ( SQL_API_FN *db2secValidatePassword)
                      ( const char *userid,
                        db2int32 useridlen,
                        const char *usernamespace,
                        db2int32 usernamespacelen,
                        db2int32 usernamespacetype,
                        const char *password,
                        db2int32 passwordlen,
                        const char *newpasswd,
                        db2int32  newpasswdlen,
                        const char *dbname,
                        db2int32 dbnamelen,
                        db2Uint32  connection_details,
                        void      **token,
                        char **errormsg,
                        db2int32 *errormsglen );
```

Parámetros de la API db2secValidatePassword

userid Entrada. ID de usuario cuya contraseña se debe verificar.

useridlen
Entrada. Longitud, en bytes, del valor del parámetro userid.

usernamespace
Entrada. Espacio de nombres del que se obtuvo el ID de usuario.

usernamespacelen
Entrada. Longitud, en bytes, del valor del parámetro usernamespace.

usernamespacetype
Entrada. Es el tipo del espacio de nombres. Los valores válidos para el parámetro usernamespacetype (definido en db2secPlugin.h) son:

- `DB2SEC_NAMESPACE_SAM_COMPATIBLE` Se corresponde con un nombre de usuario del estilo dominio\minombre

- DB2SEC_NAMESPACE_USER_PRINCIPAL Se corresponde con un nombre de usuario del estilo minombre@dominio.ibm.com

Actualmente, el sistema de base de datos DB2 sólo da soporte al valor DB2SEC_NAMESPACE_SAM_COMPATIBLE. Cuando el ID de usuario no está disponible, el parámetro usernamespacetype se establece en el valor DB2SEC_USER_NAMESPACE_UNDEFINED (definido en db2secPlugin.h).

password
Entrada. Contraseña que se debe verificar.

passwordlen
Entrada. Longitud, en bytes, del valor del parámetro password.

newpasswd
Entrada. Contraseña nueva, si la contraseña se debe cambiar. Si no hace falta cambiarla, este parámetro se establece en NULL. Si este parámetro no es NULL,la API debe validar la contraseña antigua antes de establecer el nuevo valor de la contraseña. La API no está obligada a satisfacer una petición de cambio de contraseña, pero si no lo hace, debe volver inmediatamente con el valor de retorno DB2SEC_PLUGIN_CHANGEPASSWORD_NOTSUPPORTED sin validar la antigua contraseña.

newpasswdlen
Entrada. Longitud, en bytes, del valor del parámetro newpasswd.

dbname
Entrada. Nombre de la base de datos con la que se establece conexión. La API puede hacer caso omiso del parámetro dbname o bien devolver el valor DB2SEC_PLUGIN_CONNECTIONREFUSED si tiene una política de restringir el acceso a determinadas bases de datos a usuarios que, de otra manera, tienen contraseñas válidas. Este parámetro puede ser NULL.

dbnamelen
Entrada. Longitud, en bytes, del valor del parámetro dbname. Este parámetro se establece igual a 0 si el parámetro dbname es NULL.

connection_details
Entrada. Parámetro de 32 bits, de los cuales 3 bits se utilizan para almacenar la siguiente información:

- El bit situado más a la derecha indica si el origen del ID de usuario es el valor por omisión de la API db2secGetDefaultLoginContext o si se ha proporcionado explícitamente durante la conexión.
- El segundo bit de la derecha indica si la conexión es local (mediante la comunicación entre procesos (IPC) o mediante una conexión de uno de los nodos contenidos en db2nodes.cfg en el entorno de base de datos particionada) o remota (mediante una red o un bucle de retorno). Esto da a la API la capacidad de decidir si los clientes situados en la misma máquina pueden conectarse al servidor DB2 sin una contraseña. Debido al plugin por omisión de ID de usuario/contraseña basado en el sistema operativo, las conexiones locales están permitidas sin una contraseña de los clientes en la misma máquina (suponiendo que el usuario tenga privilegios de conexión).
- El tercer bit de la derecha indica si el gestor de bases de datos DB2 llama a la API desde el extremo servidor o desde el extremo cliente.

Los valores de los bits están definidos en db2secPlugin.h:

- DB2SEC_USERID_FROM_OS (0x00000001) Indica que el ID de usuario se obtiene del sistema operativo y no se proporciona explícitamente en la sentencia de conexión.
- DB2SEC_CONNECTION_ISLOCAL (0x00000002) Indica que la conexión es local.
- DB2SEC_VALIDATING_ON_SERVER_SIDE (0x0000004) Indica si el gestor de bases de datos DB2 llama desde el extremo servidor o desde el extremo cliente para validar la contraseña. Si este valor de bits está establecido, el gestor de bases de datos DB2 llama desde el extremo servidor; de lo contrario, llama desde el extremo cliente.

El comportamiento por omisión del sistema de base de datos DB2 para una autentificación implícita es permitir la conexión sin realizar ninguna validación de contraseña. Sin embargo, los desarrolladores de plugins tienen la opción de no permitir la autentificación implícita devolviendo un error DB2SEC_PLUGIN_BADPASSWORD.

token Entrada. Puntero a los datos que se pueden pasar en forma de parámetro a las llamadas a API ulteriores durante la conexión actual. Las API posibles a las que se puede llamar son la API db2secGetAuthIDs y la API db2secGetGroupsForUser.

errormsg
Salida. Puntero a la dirección de una serie del mensaje de error ASCII asignada por el plugin y que se puede devolver en este parámetro si la ejecución de la API db2secValidatePassword no es satisfactoria.

errormsglen
Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro errormsg.

Las API y definiciones necesarias para los plugins de autentificación de GSS-API

A continuación se muestra una lista completa de las API de GSS necesarias para la interfaz de los plugins de seguridad de DB2.

Las API soportadas siguen estas especificaciones: *Generic Security Service Application Program Interface, Versión 2* (IETF RFC2743) y *Generic Security Service API Versión 2: C-Bindings* (IETF RFC2744). Antes de implementar un plugin basado en GSS-API, debe tener un conocimiento completo de esas especificaciones.

Tabla 12. Las API y definiciones necesarias para los plugins de autentificación de GSS-API

Nombre		Descripción
Las API del extremo cliente	gss_init_sec_context	Iniciar un contexto de seguridad con una aplicación homóloga.
Las API del extremo servidor	gss_accept_sec_context	Aceptar un contexto de seguridad iniciado por una aplicación homóloga.
Las API del extremo servidor	gss_display_name	Convertir un nombre de formato interno a texto.
Las API comunes	gss_delete_sec_context	Suprimir un contexto de seguridad establecido.
Las API comunes	gss_display_status	Obtener el mensaje de error de texto asociado a un código de estado de GSS-API.
Las API comunes	gss_release_buffer	Suprimir un almacenamiento intermedio.
Las API comunes	gss_release_cred	Liberar las estructuras de datos locales asociadas a una credencial de GSS-API.

Tabla 12. Las API y definiciones necesarias para los plugins de autentificación de GSS-API (continuación)

Nombre		Descripción
Las API comunes	`gss_release_name`	Suprimir un nombre con formato interno.
Definiciones necesarias	`GSS_C_DELEG_FLAG`	Solicita delegación.
Definiciones necesarias	`GSS_C_EMPTY_BUFFER`	Significa que `gss_buffer_desc` no contiene ningún dato.
Definiciones necesarias	`GSS_C_GSS_CODE`	Indica un código de estado principal GSS.
Definiciones necesarias	`GSS_C_INDEFINITE`	Indica que el mecanismo no es compatible con la caducidad de contexto.
Definiciones necesarias	`GSS_C_MECH_CODE`	Indica un código de estado secundario de GSS.
Definiciones necesarias	`GSS_C_MUTUAL_FLAG`	Se ha solicitado autentificación mutua.
Definiciones necesarias	`GSS_C_NO_BUFFER`	Denota que la variable `gss_buffer_t` no apunta a una estructura `gss_buffer_desc` válida.
Definiciones necesarias	`GSS_C_NO_CHANNEL_BINDINGS`	No existen vinculaciones de canales de comunicación.
Definiciones necesarias	`GSS_C_NO_CONTEXT`	Denota que la variable `gss_ctx_id_t` no apunta a un contexto válido.
Definiciones necesarias	`GSS_C_NO_CREDENTIAL`	Denota que la variable `gss_cred_id_t` no apunta a un descriptor de credencial válido.
Definiciones necesarias	`GSS_C_NO_NAME`	Denota que la variable `gss_name_t` no apunta a un nombre interno válido.
Definiciones necesarias	`GSS_C_NO_OID`	Utilizar el mecanismo de autentificación por omisión.
Definiciones necesarias	`GSS_C_NULL_OID_SET`	Utilizar el mecanismo por omisión.
Definiciones necesarias	`GSS_S_COMPLETE`	La API se ejecutó satisfactoriamente.
Definiciones necesarias	`GSS_S_CONTINUE_NEEDED`	El proceso no ha finalizado y se debe invocar de nuevo la API utilizando el símbolo de respuesta recibido de la entidad homóloga.

Restricciones para los plugins de autentificación de GSS-API

A continuación sigue una lista de restricciones para los plugins de autentificación de GSS-API.

- Se considera siempre que se utiliza el mecanismo de seguridad por omisión; por lo que no se hace ninguna consideración respecto a OID.
- Los únicos servicios de GSS solicitados en `gss_init_sec_context()` son la autentificación mutua y la delegación. El gestor de bases de datos DB2 solicita siempre un certificado para la delegación, pero no utiliza ese certificado para generar un nuevo certificado.
- Solo se solicita el tiempo de contexto por omisión.
- No se envían señales de contexto de `gss_delete_sec_context()` desde el cliente al servidor y viceversa.
- El anonimato no está soportado.
- La vinculación de canal no está soportada.

- Si las credenciales iniciales caducan, el gestor de bases de datos DB2 no las renueva automáticamente.
- La especificación GSS-API establece que aunque fallen las funciones `gss_init_sec_context()` o `gss_accept_sec_context()`, ambas funciones deben devolver una señal para enviar al nodo interlocutor. Sin embargo, debido a limitaciones de DRDA, el gestor de bases de datos DB2 sólo envía una señal si `gss_init_sec_context()` falla y genera una señal en la primera llamada.

Ejemplos de plugins de seguridad

Directorios de UNIX y Linux: los ejemplos de 'C' se encuentran en sqllib/samples/security/plugins y los ejemplos de plugins GSS-API de JCC (.java) se encuentran en sqllib/samples/java/jdbc

Directorio de Windows: los ejemplos de 'C' se encuentran en sqllib\samples\security\plugins y los ejemplos de GSS-API de JCC (.java) se encuentran en sqllib\samples\java\jdbc

Tabla 13. Archivos de programas de ejemplo de plugins de seguridad

Nombre del programa de ejemplo	**Descripción del programa**
combined.c	Ejemplo de autentificación combinada de IDusuario/contraseña y búsqueda de grupo.
group_file.c	Ejemplo de plugin de gestión de grupo simple basada en el archivo.
gssapi_simple.c	Ejemplo de plugin de autentificación GSS-API básica (tanto de cliente como de servidor).
IBMLDAPauthclient.c	Implementa un plugin de seguridad de DB2 del cliente que interactúa con un registro de usuarios LDAP.
IBMLDAPauthserver.c	Implementa un plugin de seguridad de DB2 del servidor que interactúa con un registro de usuarios LDAP.
IBMLDAPconfig.c	Contiene funciones relacionadas con la búsqueda y el análisis del archivo de configuración para un plugin de seguridad LDAP de DB2.
IBMLDAPgroups.c	Implementa un plugin de seguridad de DB2 para búsqueda de grupo basada en LDAP
IBMLDAPutils.c	Contiene funciones de programa de utilidad que se utilizan en el plugin de seguridad LDAP de DB2
IBMLDAPutils.h	Archivo de cabecera de plugin de seguridad LDAP
JCCKerberosPlugin.java	Implementa un plugin de GSS-API que lleva a cabo la autentificación de Kerberos utilizando IBM DB2 Universal Driver.
JCCKerberosPluginTest.java	Utilice JCCKerberosPlugin para obtener una conexión DB2 utilizando IBM DB2 Universal Driver.
JCCSimpleGSSPlugin.java	Implementa un plugin de GSS-API que realiza la comprobación de ID de usuario y contraseña utilizando IBM DB2 Universal Driver.
JCCSimpleGSSContext.java	Implemente un GSSContext para que sea utilizado por JCCSimpleGSSPlugin
JCCSimpleGSSCredential.java	Implemente una GSSCredential para que JCCSimpleGSSPlugin pueda utilizarlo

Tabla 13. Archivos de programas de ejemplo de plugins de seguridad (continuación)

Nombre del programa de ejemplo	Descripción del programa
JCCSimpleGSSException.java	Implemente un GSSException para que sea utilizado por JCCSimpleGSSPlugin
JCCSimpleGSSName.java	Implemente un GSSName para que sea utilizado por JCCSimpleGSSPlugin
JCCSimpleGSSPluginTest.java	Utilice JCCSimpleGSSPlugin para obtener la conexión DB2 utilizando IBM DB2 Universal Driver.

Las API de DB2 para hacer copias de seguridad y restauraciones en gestores de almacenamiento

DB2 proporciona una interfaz que puede ser utilizada por productos de gestión de soporte de almacenamiento de terceros para almacenar y recuperar datos para operaciones de copia de seguridad y restauración y archivos de anotaciones cronológicas. Esta interfaz está diseñada para ampliar los destinos de copia de seguridad, restauración y datos de archivado de anotaciones cronológicas de disquete, disco, cinta y Tivoli Storage Manager, que están soportados como una parte estándar de DB2.

En el resto de esta sección, estos productos de gestión de soporte de terceros se denominan productos de proveedor.

DB2 define un conjunto de prototipos de API que proporcionan una interfaz de datos de uso general para la copia de seguridad, restauración y archivado de anotaciones cronológicas que puede ser utilizada por muchos proveedores. El proveedor debe proporcionar estas API en una biblioteca compartida en sistemas basados en UNIX, o en una DLL en el sistema operativo Windows. Cuando DB2 invoca las API, se carga la biblioteca compartida o DLL especificada por la rutina de copia de seguridad, restauración o archivado de anotaciones cronológicas que ha realizado la llamada, y se invocan las API proporcionadas por el proveedor para realizar las tareas solicitadas.

Los archivos de ejemplo que muestran la funcionalidad de los proveedores de DB2 están ubicados en el directorio `sqllib/samples/BARVendor` en plataformas UNIX y en el directorio `sqllib\samples\BARVendor` en Windows.

Las definiciones siguientes describen la terminología utilizada en las descripciones de las API del plugin de copia de seguridad y restauración de almacenamiento de proveedor.

Plugin de copia de seguridad y restauración de almacenamiento de proveedor
Biblioteca de carga dinámica que DB2 cargará para acceder a las API de copia de seguridad y restauración escritas por el usuario para productos de proveedor.

Entrada
Indica que DB2 proporcionará el valor para el parámetro de la API del plugin de copia de seguridad y restauración de almacenamiento de proveedor.

Salida Indica que la API del plugin de copia de seguridad y restauración de almacenamiento de proveedor proporcionará el valor para el parámetro de la API.

db2VendorGetNextObj - Obtener el objeto siguiente en el dispositivo

Se llama a esta API una vez se ha configurado una consulta (utilizando la API sqluvint) para obtener el siguiente objeto (imagen o archivo de anotaciones archivado) que coincida con los criterios de búsqueda. Solamente puede configurarse una búsqueda a la vez para imagen o archivo de anotaciones archivado.

Autorización

Ninguna

Conexión necesaria

Base de datos.

Archivo de inclusión de la API

```
db2VendorApi.h
```

Sintaxis de la API y de las estructuras de datos

```
int db2VendorGetNextObj ( void                        * vendorCB,
                          struct db2VendorQueryInfo   * queryInfo,
                          struct Return_code          * returnCode);

typedef struct db2VendorQueryInfo
{
   db2Uint64 sizeEstimate;
   db2Uint32 type;
   SQL_PDB_NODE_TYPE dbPartitionNum;
   db2Uint16 sequenceNum;
   char db2Instance[SQL_INSTNAME_SZ + 1];
   char dbname[SQL_DBNAME_SZ + 1];
   char dbalias[SQL_ALIAS_SZ + 1];
   char timestamp[SQLU_TIME_STAMP_LEN+1];
   char filename[DB2VENDOR_MAX_FILENAME_SZ + 1];
   char owner[DB2VENDOR_MAX_OWNER_SZ + 1];
   char mgmtClass[DB2VENDOR_MAX_MGMTCLASS_SZ + 1];
   char oldestLogfile[DB2_LOGFILE_NAME_LEN + 1];
} db2VendorQueryInfo;
```

Parámetros de la API db2VendorGetNextObj

vendorCB
Entrada. Puntero a un espacio asignado por la biblioteca de proveedor.

queryInfo
Salida. Puntero a una estructura db2VendorQueryInfo que rellenará la biblioteca de proveedor.

returnCode
Salida. Código de retorno devuelto por la llamada a la API.

Parámetros de la estructura de datos db2VendorQueryInfo

sizeEstimate
Especifica el tamaño estimado del objeto.

type Especifica el tipo de imagen si el objeto es una imagen de copia de seguridad.

dbPartitionNum
Especifica el número de la partición de base de datos a la que pertenece el objeto.

sequenceNum
Especifica la extensión de archivo para la imagen de copia de seguridad. Sólo es válido si el objeto es una copia de seguridad.

db2Instance
Especifica el nombre de la instancia a la que pertenece el objeto.

dbname
Especifica el nombre de la base de datos a la que pertenece el objeto.

dbalias
Especifica el alias de la base de datos a la que pertenece el objeto.

timestamp
Especifica la indicación de fecha y hora utilizada para identificar la imagen de copia de seguridad. Sólo es válido si el objeto es una imagen de copia de seguridad.

filename
Especifica el nombre del objeto si el objeto es una imagen de copia de carga o un archivo de anotaciones archivado.

owner Especifica el propietario del objeto.

mgmtClass
Especifica la clase de gestión bajo la que se ha almacenado el objeto (utilizado por TSM).

oldestLogfile
Especifica el archivo de anotaciones más antiguo almacenado con una imagen de copia de seguridad.

Notas de uso

No todos los b pertenecerán a cada objeto o a cada proveedor. Los parámetros obligatorios que deben rellenarse son db2Instance, dbname, dbalias, timestamp (para imágenes), filename (para anotaciones e imágenes de copia de carga), owner, sequenceNum (para imágenes) y dbPartitionNum. Los parámetros restantes se dejarán para que los definan los proveedores específicos. Si un parámetro no corresponde, deberá inicializarse como "" para series y 0 para tipos numéricos.

Códigos de retorno

La tabla siguiente muestra los posibles códigos de retorno de esta API.

Tabla 14. Códigos de retorno de la API db2VendorGetNextObj

Número	Código de retorno	Explicación
2	SQLUV_COMM_ERROR	Error de comunicación con dispositivo - Error.

Tabla 14. Códigos de retorno de la API db2VendorGetNextObj (continuación)

Número	Código de retorno	Explicación
4	SQLUV_INV_ACTION	Se ha solicitado una acción no válida o la combinación de los parámetros de entrada da lugar a una operación que no es posible - Error.
5	SQLUV_NO_DEV_AVAIL	No hay ningún dispositivo disponible en este momento - Error.
6	SQLUV_OBJ_NOT_FOUND	No se ha encontrado ningún objeto que suprimir - Error.
12	SQLUV_INV_DEV_HANDLE	Handle de dispositivo no válido - Error.
14	SQLUV_END_OF_DATA	No hay más objetos de consulta que devolver - Correcto.
18	SQLUV_DEV_ERROR	Error de dispositivo - Error.
19	SQLUV_WARNING	Aviso, consulte el código de retorno - Correcto.
21	SQLUV_MORE_DATA	Hay más objetos de consulta que devolver - Correcto.
25	SQLUV_IO_ERROR	Error de E/S - Error.
30	SQLUV_UNEXPECTED_ERROR	Se ha encontrado un error grave - Error.

db2VendorQueryApiVersion - Obtener el nivel soportado de la API de almacenamiento de proveedor

Determina qué nivel de la API de almacenamiento de proveedor está soportado por el conector de almacenamiento de copia de seguridad y restauración. Si el conector de almacenamiento de proveedor especificado no es compatible con DB2, no se utilizará.

Si un conector de almacenamiento de proveedor no tiene implementada esta API para anotaciones cronológicas, no se puede utilizar y DB2 informará de un error. Esto no afectará a las imágenes que actualmente trabajan con bibliotecas de proveedor existentes.

Autorización

Ninguna

Conexión necesaria

Base de datos.

Archivo de inclusión de la API

```
db2VendorApi.h
```

Sintaxis de la API y de las estructuras de datos

```
void db2VendorQueryApiVersion ( db2Uint32  * supportedVersion );
```

Parámetros de la API db2VendorQueryApiVersion

supportedVersion
Salida. Devuelve la versión de la API de almacenamiento de proveedor que la biblioteca de proveedor soporta.

Notas de uso

Se llamará a esta API antes de invocar cualquier otra API de almacenamiento de proveedor.

sqluvdel - Suprimir sesión confirmada

Suprime sesiones confirmadas de un dispositivo de proveedor.

Autorización

Ninguna

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqluvend.h
```

Sintaxis de la API y de las estructuras de datos

```
int sqluvdel ( struct Init_input *in,
               struct Init_output *vendorDevData,
               struct Return_code *return_code);
```

Parámetros de la API sqluvdel

in Entrada. Espacio asignado para Init_input y Return_code.

vendorDevData
Salida. Estructura que contiene los datos de salida devueltos por el dispositivo del proveedor.

return_code
Salida. Código de retorno de la llamada de la API. El objeto apuntado por la estructura Init_input se suprime.

Notas de uso

Si hay varias sesiones abiertas y algunas sesiones se confirman, pero una de ellas falla, se llama a esta API para suprimir las sesiones confirmadas. No se especifica ningún número de orden; sqluvdel es responsable de buscar todos los objetos que se han creado durante una operación de copia de seguridad determinada y de suprimirlos. Se utiliza la información de la estructura Init_input para identificar los datos de salida que deben suprimirse. La llamada a sqluvdel es responsable de establecer las conexiones o sesiones necesarias para suprimir un objeto de copia de seguridad del dispositivo de proveedor. Si el código de retorno devuelto por esta llamada es SQLUV_DELETE_FAILED, DB2 no informa al llamador ya que DB2 devuelve la primera anomalía muy grave y pasa por alto las anomalías subsiguientes. En este caso, para que DB2 hubiese llamado a la API sqluvdel, se debería haber producido un error inicial muy grave.

Códigos de retorno

Tabla 15. Códigos de retorno válidos para sqluvdel y acción de base de datos resultante

Literal en archivo de cabecera	Descripción	Próxima llamada probable
SQLUV_OK	Operación satisfactoria	No se realizan más llamadas.
SQLUV_DELETE_FAILED	La petición de supresión ha fallado.	No se realizan más llamadas.

sqluvend - Desenlazar un dispositivo de proveedor y liberar sus recursos

Desenlaza un dispositivo de proveedor y libera todos sus recursos relacionados. Se deben liberar todos los recursos no utilizados (por ejemplo, espacio asignado y descriptores de contexto de archivo) antes de que la llamada a la API sqluvend vuelva a DB2.

Autorización

Ninguna

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqluvend.h
```

Sintaxis de la API y de las estructuras de datos

```
int sqluvend ( sqlint32            action,
               void *hdle,
               struct Init_output *in_out,
               struct Return_code *return_code);
```

Parámetros de la API sqluvend

action Entrada. Se utiliza para confirmar o cancelar la sesión:

- SQLUV_COMMIT (0 = para confirmar)
- SQLUV_ABORT (1 = para finalizar anormalmente)

hdle Entrada. Puntero a la estructura Init_output.

in_out Salida. Espacio desasignado para Init_output. Los datos se han confirmado en almacenamiento estable para una copia de seguridad si la acción es confirmar. Los datos se eliminan para una copia de seguridad si la acción es cancelar.

return_code
Salida. Código de retorno devuelto por la llamada a la API.

Notas de uso

Esta API se invoca para cada sesión que se ha abierto. Existen dos códigos de acción posibles:

Commit

La salida de datos de esta sesión o la lectura de datos de la sesión ha finalizado.

Para una sesión de escritura (copia de seguridad), si el proveedor vuelve a DB2 con un código de retorno SQLUV_OK, DB2 considera que el producto de proveedor ha guardado los datos de salida debidamente, y se puede acceder a ellos si se hace referencia a los mismos en una llamada posterior a sqluvint.

Para una sesión de lectura (restauración), si el proveedor vuelve a DB2 con un código de retorno SQLUV_OK, los datos no se deben suprimir ya que pueden ser necesarios nuevamente. Si el proveedor devuelve SQLUV_COMMIT_FAILED, DB2 considera que existen problemas con toda la operación de copia de seguridad o restauración. Todas las sesiones activas se terminan mediante llamadas a sqluvend con la acción = SQLUV_ABORT. Para una operación de copia de seguridad, las sesiones confirmadas reciben una secuencia de llamadas sqluvint, sqluvdel y sqluvend.

Abort DB2 ha encontrado un problema y no se leerán ni grabarán más datos en la sesión.

Para una sesión de escritura (copia de seguridad), el proveedor debe suprimir el conjunto de datos de salida parcial y utilizar el código de retorno SQLUV_OK si se suprime la salida parcial. DB2 considera que existen problemas con toda la copia de seguridad. Todas las sesiones activas se terminan mediante llamadas a sqluvend con la acción = SQLUV_ABORT, y las sesiones confirmadas reciben una secuencia de llamadas sqluvint, sqluvdel y sqluvend.

Para una sesión de lectura (restauración), el proveedor no debe suprimir los datos (ya que pueden volver a ser necesarios), pero debe realizar una limpieza y volver a DB2 con un código de retorno SQLUV_OK. DB2 termina todas las sesiones de restauración mediante llamadas a sqluvend con la acción = SQLUV_ABORT. Si el proveedor devuelve SQLUV_ABORT_FAILED a DB2, no se informa al llamador de este error ya que DB2 devuelve la primera anomalía muy grave y pasa por alto las anomalías subsiguientes. En este caso, para que DB2 hubiese llamado a sqluvend con la acción = SQLUV_ABORT, se debería haber producido un error inicial muy grave.

Códigos de retorno

Tabla 16. Códigos de retorno válidos para sqluvend y acción resultado de DB2

Literal en archivo de cabecera	Descripción	Próxima llamada probable	Otros comentarios
SQLUV_OK	Operación satisfactoria	No se realizan más llamadas.	Libere toda la memoria asignada a esta sesión y termine.
SQLUV_COMMIT_FAILED	La petición de confirmación ha fallado.	No se realizan más llamadas.	Libere toda la memoria asignada a esta sesión y termine.

Tabla 16. Códigos de retorno válidos para sqluvend y acción resultado de DB2 (continuación)

Literal en archivo de cabecera	Descripción	Próxima llamada probable	Otros comentarios
SQLUV_ABORT_FAILED	La petición de finalización anómala ha fallado.	No se realizan más llamadas.	

sqluvget - Leer datos de un dispositivo de proveedor

Una vez inicializado el dispositivo de proveedor con la API sqluvint, DB2 llama esta API para leer del dispositivo durante una operación de restauración.

Autorización

Ninguna

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqluvend.h
```

Sintaxis de la API y de las estructuras de datos

```
int sqluvget ( void *               hdle,
               struct Data *data,
               struct Return_code *return_code);
```

Parámetros de la API sqluvget

hdle Entrada. Puntero al espacio asignado para la estructura de Data (incluido el almacenamiento intermedio de datos) y el código de retorno.

data Entrada o salida. Puntero a la estructura de Data.

return_code
Salida. Código de retorno devuelto por la llamada a la API.

Notas de uso

Esta API es utilizada por el programa de utilidad de restauración.

Códigos de retorno

Tabla 17. Códigos de retorno válidos para sqluvget y acción resultante de DB2

Literal en archivo de cabecera	Descripción	Próxima llamada probable	Otros comentarios
SQLUV_OK	Operación satisfactoria.	sqluvget	DB2 procesa los datos
SQLUV_COMM_ERROR	Error de comunicación con dispositivo.	sqluvend, action = SQLU_ABORT (ver nota debajo)	La sesión terminará.

Tabla 17. Códigos de retorno válidos para sqluvget y acción resultante de DB2 (continuación)

Literal en archivo de cabecera	Descripción	Próxima llamada probable	Otros comentarios
SQLUV_INV_ACTION	Se ha solicitado una acción no válida.	sqluvend, action = SQLU_ABORT (ver nota debajo)	La sesión terminará.
SQLUV_INV_DEV_HANDLE	Handle de dispositivo no válido.	sqluvend, action = SQLU_ABORT (ver nota debajo)	La sesión terminará.
SQLUV_INV_BUFF_SIZE	Se ha especificado un tamaño de almacenamiento intermedio no válido.	sqluvend, action = SQLU_ABORT (ver nota debajo)	La sesión terminará.
SQLUV_DEV_ERROR	Error de dispositivo.	sqluvend, action = SQLU_ABORT (ver nota debajo)	La sesión terminará.
SQLUV_WARNING	Aviso. No se debe utilizar para indicar el fin de soporte de almacenamiento a DB2; utilice SQLUV_ENDOFMEDIA o SQLUV_ENDOFMEDIA_NO _DATA para este propósito. Sin embargo, las condiciones de dispositivo no preparado se pueden indicar utilizando este código de retorno.	sqluvget o sqluvend, action= SQLU_ABORT	
SQLUV_LINK_NOT_EXIST	No existe ningún enlace actualmente	sqluvend, action = SQLU_ABORT (ver nota debajo)	La sesión terminará.
SQLUV_MORE_DATA	Operación satisfactoria; más datos disponibles.	sqluvget	
SQLUV_ENDOFMEDIA_NO_ DATA	Fin de soporte de almacenamiento y 0 bytes leídos (por ejemplo, fin de cinta).	sqluvend	
SQLUV_ENDOFMEDIA	Fin de soporte de almacenamiento y >0 bytes leídos (por ejemplo, fin de cinta).	sqluvend	DB2 procesa los datos y, a continuación, maneja la condición de fin de soporte de almacenamiento.
SQLUV_IO_ERROR	Error de E/S.	sqluvend, action = SQLU_ABORT (ver nota debajo)	La sesión terminará.

Nota: Próxima llamada: si la llamada siguiente es sqluvend, action = SQLU_ABORT, terminará esta sesión y todas las demás sesiones activas.

sqluvint - Inicializar y enlazar con un dispositivo de proveedor

Proporciona información para inicializar un dispositivo de proveedor y establecer un enlace lógico entre DB2 y el dispositivo de proveedor.

Autorización

Ninguna

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqluvend.h
```

Sintaxis de la API y de las estructuras de datos

```
int sqluvint ( struct Init_input *in,
               struct Init_output *out,
               struct Return_code *return_code);
```

Parámetros de la API sqluvint

in Entrada. Estructura que contiene información proporcionada por DB2 para establecer un enlace lógico con el dispositivo de proveedor.

out Salida. Estructura que contiene los datos de salida devueltos por el dispositivo del proveedor.

return_code
Salida. Estructura que contiene el código de retorno que se debe pasar a DB2, y una breve explicación del texto.

Notas de uso

Por cada sesión de E/S de soporte de almacenamiento, DB2 llamará a esta API para obtener un descriptor de contexto de dispositivo. Si por cualquier razón, la API de almacenamiento de proveedor encuentra un error durante la inicialización, lo indicará mediante un código de retorno. Si el código de retorno indica un error, DB2 puede decidir terminar la operación llamando a la API sqluvend. Los detalles sobre posibles códigos de retorno y la reacción de DB2 a cada uno de ellos, se encuentra en la tabla de códigos de retorno (vea la tabla siguiente).

La estructura Init_input contiene elementos que pueden ser utilizados por el producto del proveedor para determinar si la copia de seguridad o restauración puede continuar:

size_HI_order y size_LOW_order
Es el tamaño estimado de la copia de seguridad. Se puede utilizar para determinar si los dispositivos del proveedor pueden trabajar con el tamaño de la imagen de copia de seguridad. Se puede utilizar para calcular la cantidad de soporte de almacenamiento extraíble que será necesario para contener la copia de seguridad. Puede ser ventajoso fallar en la primera llamada a sqluvint si se prevén problemas.

req_sessions
Es el número de sesiones solicitadas por el usuario que se pueden utilizar en combinación con el tamaño estimado y el nivel de interacción con el usuario para determinar si es posible la operación de copia de seguridad o restauración.

prompt_lvl
El nivel de interacción con el usuario indica al proveedor si es posible solicitar acciones tales como cambiar el soporte de almacenamiento

extraíble (por ejemplo, colocar otra cinta en la unidad de cintas). Esto puede sugerir que la operación no puede continuar debido a que no habrá una forma de solicitar acciones al usuario. Si el nivel de interacción con el usuario es WITHOUT PROMPTING y la cantidad de soportes de almacenamiento extraíbles es mayor que el número de sesiones solicitadas, DB2 no podrá terminar la operación satisfactoriamente.

DB2 designa la copia de seguridad que se está escribiendo o la restauración que se debe leer a través de campos de la estructura DB2_info. En el caso de una acción = SQLUV_READ, el producto del proveedor debe comprobar la existencia del objeto designado. Si no se puede encontrar el objeto, el código de retorno se debe establecer en SQLUV_OBJ_NOT_FOUND para que DB2 emprenda la acción apropiada.

Una vez finalizada la inicialización satisfactoriamente, DB2 continuará llamando a otras API de transferencia de datos, pero puede terminar la sesión en cualquier momento con una llamada sqluvend.

Códigos de retorno

Tabla 18. Códigos de retorno válidos para sqluvint y acción resultante de DB2

Literal en archivo de cabecera	Descripción	Próxima llamada probable	Otros comentarios
SQLUV_OK	Operación satisfactoria.	sqluvput, sqluvget (ver comentarios)	Si action = SQLUV_WRITE, la próxima llamada será a la API sqluvput (a datos BACKUP). Si action = SQLUV_READ, verifique la existencia del objeto nombrado antes de devolver SQLUV_OK; la siguiente llamada será a la API sqluvget para restaurar los datos.
SQLUV_LINK_EXIST	Sesión activada anteriormente.	No se realizan más llamadas.	La inicialización de la sesión falla. Libere memoria asignada a esta sesión y termine. No se recibirá una llamada a la API sqluvend, ya que la sesión no se ha establecido nunca.
SQLUV_COMM_ERROR	Error de comunicación con dispositivo.	No se realizan más llamadas.	La inicialización de la sesión falla. Libere memoria asignada a esta sesión y termine. No se recibirá una llamada a la API sqluvend, ya que la sesión no se ha establecido nunca.
SQLUV_INV_VERSION	Los productos DB2 y del proveedor son incompatibles.	No se realizan más llamadas.	La inicialización de la sesión falla. Libere memoria asignada a esta sesión y termine. No se recibirá una llamada a la API sqluvend, ya que la sesión no se ha establecido nunca.

Tabla 18. Códigos de retorno válidos para sqluvint y acción resultante de DB2 (continuación)

Literal en archivo de cabecera	Descripción	Próxima llamada probable	Otros comentarios
SQLUV_INV_ACTION	Se ha solicitado una acción no válida. También puede utilizarse para indicar que la combinación de parámetros da como resultado una operación que no es posible.	No se realizan más llamadas.	La inicialización de la sesión falla. Libere memoria asignada a esta sesión y termine. No se recibirá una llamada a la API sqluvend, ya que la sesión no se ha establecido nunca.
SQLUV_NO_DEV_AVAIL	No hay ningún dispositivo disponible en este momento.	No se realizan más llamadas.	La inicialización de la sesión falla. Libere memoria asignada a esta sesión y termine. No se recibirá una llamada a la API sqluvend, ya que la sesión no se ha establecido nunca.
SQLUV_OBJ_NOT_FOUND	El objeto especificado no puede encontrarse. Debe utilizarse cuando la acción de la llamada de sqluvint es ″R″ (lectura) y el objeto solicitado no puede encontrarse en función de los criterios especificados en la estructura DB2_info.	No se realizan más llamadas.	La inicialización de la sesión falla. Libere memoria asignada a esta sesión y termine. No se recibirá una llamada a la API sqluvend, ya que la sesión no se ha establecido nunca.
SQLUV_OBJS_FOUND	Más de 1 objeto coincide con los criterios especificados. Este será el resultado cuando la acción de la llamada de sqluvint es ″R″ (lectura) y más de un objeto coincide con los criterios especificados en la estructura DB2_info.	No se realizan más llamadas.	La inicialización de la sesión falla. Libere memoria asignada a esta sesión y termine. No se recibirá una llamada a la API sqluvend, ya que la sesión no se ha establecido nunca.
SQLUV_INV_USERID	Se ha especificado un ID de usuario no válido.	No se realizan más llamadas.	La inicialización de la sesión falla. Libere memoria asignada a esta sesión y termine. No se recibirá una llamada a la API sqluvend, ya que la sesión no se ha establecido nunca.
SQLUV_INV_PASSWORD	Contraseña incorrecta.	No se realizan más llamadas.	La inicialización de la sesión falla. Libere memoria asignada a esta sesión y termine. No se recibirá una llamada a la API sqluvend, ya que la sesión no se ha establecido nunca.
SQLUV_INV_OPTIONS	Se han encontrado opciones no válidas en el campo de opciones de proveedor.	No se realizan más llamadas.	La inicialización de la sesión falla. Libere memoria asignada a esta sesión y termine. No se recibirá una llamada a la API sqluvend, ya que la sesión no se ha establecido nunca.

Tabla 18. Códigos de retorno válidos para sqluvint y acción resultante de DB2 (continuación)

Literal en archivo de cabecera	Descripción	Próxima llamada probable	Otros comentarios
SQLUV_INIT_FAILED	La inicialización ha fallado y la sesión debe terminar.	No se realizan más llamadas.	La inicialización de la sesión falla. Libere memoria asignada a esta sesión y termine. No se recibirá una llamada a la API sqluvend, ya que la sesión no se ha establecido nunca.
SQLUV_DEV_ERROR	Error de dispositivo.	No se realizan más llamadas.	La inicialización de la sesión falla. Libere memoria asignada a esta sesión y termine. No se recibirá una llamada a la API sqluvend, ya que la sesión no se ha establecido nunca.
SQLUV_MAX_LINK_GRANT	Se ha establecido el número máximo de enlaces.	sqluvput, sqluvget (ver comentarios).	DB2 lo considera como un aviso. El aviso indica a DB2 que no abra sesiones adicionales con el producto del proveedor, ya que se ha llegado al número máximo de sesiones que puede soportar (nota: puede deberse a disponibilidad de dispositivos). Si action = SQLUV_WRITE (BACKUP), la próxima llamada será a la API sqluvput. Si action = SQLUV_READ, verifique la existencia del objeto nombrado antes de devolver SQLUV_MAX_LINK_GRANT; la siguiente llamada se hará a la API sqluvget para restaurar los datos.
SQLUV_IO_ERROR	Error de E/S.	No se realizan más llamadas.	La inicialización de la sesión falla. Libere memoria asignada a esta sesión y termine. No se recibirá una llamada a la API sqluvend, ya que la sesión no se ha establecido nunca.
SQLUV_NOT_ENOUGH_SPACE	No hay espacio suficiente para almacenar toda la imagen de copia de seguridad; el tamaño estimado se suministra como valor de 64 bits, en bytes.	No se realizan más llamadas.	La inicialización de la sesión falla. Libere memoria asignada a esta sesión y termine. No se recibirá una llamada a la API sqluvend, ya que la sesión no se ha establecido nunca.

sqluvpt - Escribir datos en un dispositivo de proveedor

Una vez inicializado un dispositivo de proveedor con la API sqluvint, DB2 llama esta API para grabar en el dispositivo durante una operación de copia de seguridad.

Autorización

Ninguna

Conexión necesaria

Base de datos

Archivo de inclusión de la API

```
sqluvend.h
```

Sintaxis de la API y de las estructuras de datos

```
int sqluvput ( void *               hdle,
               struct Data *data,
               struct Return_code *return_code);
```

Parámetros de la API sqluvput

hdle Entrada. Puntero al espacio asignado para la estructura DATA (incluido el almacenamiento de datos) y Return_code.

data Salida. Almacenamiento de datos llenado con datos que se deben escribir.

return_code
Salida. Código de retorno devuelto por la llamada a la API.

Notas de uso

Esta API es utilizada por el programa de utilidad de copia de seguridad.

Códigos de retorno

Tabla 19. Códigos de retorno válidos para sqluvput y acción resultante de DB2

Literal en archivo de cabecera	Descripción	Próxima llamada probable	Otros comentarios
SQLUV_OK	Operación satisfactoria.	sqluvput o sqluvend, si ha finalizado (por ejemplo, DB2 no tiene más datos)	Informa a otros procesos de una operación satisfactoria.
SQLUV_COMM_ERROR	Error de comunicación con dispositivo.	sqluvend, action = SQLU_ABORT (ver nota abajo).	La sesión terminará.
SQLUV_INV_ACTION	Se ha solicitado una acción no válida.	sqluvend, action = SQLU_ABORT (ver nota abajo).	La sesión terminará.
SQLUV_INV_DEV_HANDLE	Handle de dispositivo no válido.	sqluvend, action = SQLU_ABORT (ver nota abajo).	La sesión terminará.
SQLUV_INV_BUFF_SIZE	Se ha especificado un tamaño de almacenamiento intermedio no válido.	sqluvend, action = SQLU_ABORT (ver nota abajo).	La sesión terminará.
SQLUV_ENDOFMEDIA	Se ha llegado al final del soporte de almacenamiento; por ejemplo, fin de cinta.	sqluvend	

Tabla 19. Códigos de retorno válidos para sqluvput y acción resultante de DB2 (continuación)

Literal en archivo de cabecera	Descripción	Próxima llamada probable	Otros comentarios
SQLUV_DATA_RESEND	El dispositivo ha solicitado que se envíe de nuevo almacenamiento intermedio.	sqluvput	DB2 retransmitirá el último almacenamiento intermedio. Esta operación sólo se realizará una vez.
SQLUV_DEV_ERROR	Error de dispositivo.	sqluvend, action = SQLU_ABORT (ver nota abajo).	La sesión terminará.
SQLUV_WARNING	Aviso. No se debe utilizar para indicar el fin de soporte de almacenamiento a DB2; utilice SQLUV_ENDOFMEDIA para este propósito. Sin embargo, mediante este código de retorno pueden indicarse condiciones de dispositivo no preparado.	sqluvput	
SQLUV_LINK_NOT_EXIST	No existe ningún enlace actualmente.	sqluvend, action = SQLU_ABORT (ver nota abajo).	La sesión terminará.
SQLUV_IO_ERROR	Error de E/S.	sqluvend, action = SQLU_ABORT (ver nota abajo).	La sesión terminará.

Nota: Próxima llamada: si la llamada siguiente es sqluvend, action = SQLU_ABORT, terminará esta sesión y todas las demás sesiones activas. Las sesiones confirmadas se suprimen con una secuencia de llamadas a sqluvint, sqluvdel y sqluvend.

DB2_info

Contiene información sobre el producto DB2 y la base de datos de la que se está haciendo una copia de seguridad o restaurando. Esta estructura se utiliza para identificar DB2 ante el dispositivo de proveedor y para describir una sesión determinada entre DB2 y el dispositivo de proveedor. La estructura se pasa al plugin de copia de seguridad y restauración del almacenamiento de proveedor como parte de la estructura de datos Init_input.

Tabla 20. Campos de la estructura DB2_info

Nombre de campo	Tipo de datos	Descripción
DB2_id	char	Identificador del producto DB2. La longitud máxima de la serie a la que señala es de 8 caracteres.
version	char	Versión actual del producto DB2. La longitud máxima de la serie a la que señala es de 8 caracteres.

Tabla 20. Campos de la estructura DB2_info (continuación)

Nombre de campo	Tipo de datos	Descripción
release	char	Release actual del producto DB2. Se establece en NULL si no es significativa. La longitud máxima de la serie a la que señala es de 8 caracteres.
level	char	Nivel actual del producto DB2. Se establece en NULL si no es significativa. La longitud máxima de la serie a la que señala es de 8 caracteres.
action	char	Especifica la acción que se debe realizar. La longitud máxima de la serie a la que señala es de 1 carácter.
filename	char	Nombre del archivo utilizado para identificar la imagen de copia de seguridad. Si es NULL, los valores server_id, db2instance, dbname y timestamp identificarán de forma exclusiva la imagen de copia de seguridad. La longitud máxima de la serie a la que señala es de 255 caracteres.
server_id	char	Nombre exclusivo que identifica el servidor en el que reside la base de datos. La longitud máxima de la serie a la que señala es de 8 caracteres.
db2instance	char	El ID de db2instance. Es el ID de usuario que invoca el mandato. La longitud máxima de la serie a la que señala es de 8 caracteres.
type	char	Especifica el tipo de copia de seguridad o de restauración que se realiza. Los valores posible son los siguientes: Cuando la acción es SQLUV_WRITE: 0 - copia de seguridad de base de datos completa 3 - copia de seguridad a nivel de espacio de tablas. Cuando la acción es SQLUV_READ: 0 - restauración completa 3 - restauración de espacio de tablas en línea 4 - restauración de espacio de tablas 5 - restauración de archivo histórico

Tabla 20. Campos de la estructura DB2_info (continuación)

Nombre de campo	Tipo de datos	Descripción
dbname	char	Nombre de la base de datos que se debe copiar o restaurar. La longitud máxima de la serie a la que señala es de 8 caracteres.
alias	char	Alias de la base de datos que se debe copiar o restaurar. La longitud máxima de la serie a la que señala es de 8 caracteres.
timestamp	char	Indicación de fecha y hora utilizada para identificar la imagen de copia de seguridad. La longitud máxima de la serie a la que señala es de 26 caracteres.
sequence	char	Especifica la extensión de archivo para la imagen de copia de seguridad. Para operaciones de escritura, el valor de la primera sesión es 1 y, cada vez que se inicia otra sesión con una llamada a sqluvint, el valor se incrementa en 1. Para operaciones de lectura, el valor siempre es cero. La longitud máxima de la serie a la que señala es de 3 caracteres.
obj_list	struct sqlu_gen_list	Reservado para una utilización futura.
max_bytes_per_txn	sqlint32	Especifica al proveedor, en bytes, el tamaño del almacenamiento intermedio de transferencia especificado por el usuario.
image_filename	char	Reservado para una utilización futura.
reserve	void	Reservado para una utilización futura.
nodename	char	Nombre del nodo en el que se ha generado la copia de seguridad.
password	char	Contraseña del nodo en el que se ha generado la copia de seguridad.
owner	char	ID del originador de la copia de seguridad.
mcNameP	char	Clase de gestión.

Tabla 20. Campos de la estructura DB2_info (continuación)

Nombre de campo	Tipo de datos	Descripción
nodeNum	SQL_PDB_NODE_TYPE	Número de nodo. La interfaz de proveedor da soporte a números mayores que 255.

Nota: Todos los campos con el tipo de datos char son series de terminación nula.

Los parámetros filename, o server_id, db2instance, type, dbname y timestamp identifican de forma exclusiva la imagen de copia de seguridad. El número de secuencia, especificado por secuencia, identifica la extensión de archivo. Cuando se debe restaurar una imagen de copia de seguridad, se deben especificar los mismos valores para recuperar la imagen de copia de seguridad. Dependiendo del producto de proveedor, si se utiliza filename, el valor de los demás parámetros se puede establecer en NULL, y viceversa.

Sintaxis de la API y de las estructuras de datos

```
typedef struct DB2_info
{
  char *DB2_id;
  char *version;
  char *release;
  char *level;
  char *action;
  char *filename;
  char *server_id;
  char *db2instance;
  char *type;
  char *dbname;
  char *alias;
  char *timestamp;
  char *sequence;
  struct sqlu_gen_list
                 *obj_list;
  sqlint32 max_bytes_per_txn;
  char *image_filename;
  void *reserve;
  char *nodename;
  char *password;
  char *owner;
  char *mcNameP;
  SQL_PDB_NODE_TYPE nodeNum;
} DB2_info ;
```

Vendor_info

Contiene información, devuelta a DB2 como parte de la estructura Init_output, que identifica el proveedor y la versión del dispositivo de proveedor.

Tabla 21. Campos de la estructura Vendor_info

Nombre de campo	Tipo de datos	Descripción
vendor_id	char	Identificador del proveedor. La longitud máxima de la serie a la que señala es de 64 caracteres.

Tabla 21. Campos de la estructura Vendor_info (continuación)

Nombre de campo	Tipo de datos	Descripción
version	char	Versión actual del producto de proveedor. La longitud máxima de la serie a la que señala es de 8 caracteres.
release	char	Release actual del producto de proveedor. Se establece en NULL si no es significativa. La longitud máxima de la serie a la que señala es de 8 caracteres.
level	char	Nivel actual del producto de proveedor. Se establece en NULL si no es significativa. La longitud máxima de la serie a la que señala es de 8 caracteres.
server_id	char	Nombre exclusivo que identifica el servidor en el que reside la base de datos. La longitud máxima de la serie a la que señala es de 8 caracteres.
max_bytes_per_txn	sqlint32	Tamaño máximo de almacenamiento intermedios de transferencia soportado. Especificado por el proveedor, en bytes. Sólo se utiliza si el código de retorno de la API de inicialización del proveedor es SQLUV_BUFF_SIZE, lo que indica que se ha especificado un tamaño de almacenamiento intermedio no válido.
num_objects_in_backup	sqlint32	Número de sesiones utilizadas para realizar una copia de seguridad completa. Se utiliza para determinar cuándo se han procesado todas las imágenes de copia de seguridad durante una operación de restauración.
reserve	void	Reservado para una utilización futura.

Nota: Todos los campos con el tipo de datos char son series de terminación nula.

Sintaxis de la API y de las estructuras de datos

```
typedef struct Vendor_info
{
  char *vendor_id;
  char *version;
  char *release;
```

```
  char *level;
  char *server_id;
  sqlint32 max_bytes_per_txn;
  sqlint32 num_objects_in_backup;
  void *reserve;
} Vendor_info;
```

Init_input

Contiene información proporcionada por DB2 para configurar y establecer un enlace lógico con un dispositivo de proveedor. Esta estructura de datos es utilizada por DB2 para enviar información al plugin de copia de seguridad y restauración del almacenamiento de proveedor mediante las API sqluvint y sqluvdel.

Tabla 22. Campos de la estructura Init_input.

Nombre de campo	Tipo de datos	Descripción
DB2_session	struct DB2_info	Descripción de la sesión desde la perspectiva de DB2.
size_options	unsigned short	Longitud del campo de opciones. Al utilizar la función de copia de seguridad o de restauración de DB2, los datos de este campo se pasan directamente desde el parámetro VendorOptionsSize.
size_HI_order	sqluint32	32 bits más a la izquierda del cálculo de tamaño de la base de datos en bytes; el tamaño total es 64 bits.
size_LOW_order	sqluint32	32 bits más a la derecha del cálculo de tamaño de la base de datos en bytes; el tamaño total es 64 bits.
options	void	Esta información se pasa desde la aplicación cuando se invoca la función de copia de seguridad o restauración. Esta estructura de datos debe ser plana; en otras palabras, no se da soporte a ningún nivel de indirección. La reversión de bytes no se realiza, y la página de códigos de estos datos no se comprueba. Al utilizar la función de copia de seguridad o de restauración de DB2, los datos de este campo se pasan directamente desde el parámetro pVendorOptions.
reserve	void	Reservado para una utilización futura.

Tabla 22. Campos de la estructura Init_input. (continuación)

Nombre de campo	Tipo de datos	Descripción
prompt_lvl	char	Nivel de interacción solicitado por el usuario cuando se ha invocado una operación de copia de seguridad o restauración. La longitud máxima de la serie a la que señala es de 1 carácter. Este campo es una serie terminada en nulo.
num_sessions	unsigned short	Número de sesiones solicitadas por el usuario cuando se ha invocado una operación de copia de seguridad o restauración.

Sintaxis de la API y de las estructuras de datos

```
typedef struct Init_input
{
   struct DB2_info *DB2_session;
   unsigned short   size_options;
   sqluint32 size_HI_order;
   sqluint32 size_LOW_order;
   void *options;
   void *reserve;
   char *prompt_lvl;
   unsigned short   num_sessions;
} Init_input;
```

Init_output

Contiene un bloque de control para la sesión e información devuelta por el plugin de copia de seguridad y restauración de almacenamiento de proveedor para DB2. Esta estructura de datos es utilizada por las API sqluvint y sqluvdel.

Tabla 23. Campos de la estructura Init- output

Nombre de campo	Tipo de datos	Descripción
vendor_session	struct Vendor_info	Contiene información para identificar el proveedor en DB2.
pVendorCB	void	Bloque de control de proveedor.
reserve	void	Reservado para una utilización futura.

Sintaxis de la API y de las estructuras de datos

```
typedef struct Init_output
{
   struct Vendor_info * vendor_session;
   void               * pVendorCB;
   void               * reserve;
} Init_output ;
```

Data

Contiene datos transferidos entre DB2 y un dispositivo de proveedor. Esta estructura es utilizada por la API sqluvput al escribir datos en el dispositivo de proveedor y por la API sqluvget al leer datos del dispositivo de proveedor.

Tabla 24. Campos de la estructura Data

Nombre de campo	Tipo de datos	Descripción
obj_num	sqlint32	Número de secuencia asignado por DB2 durante una operación de copia de seguridad.
buff_size	sqlint32	Tamaño del almacenamiento intermedio.
actual_buff_size	sqlint32	Número real de bytes enviados o recibidos. No debe sobrepasar el valor de buff_size.
dataptr	void	Puntero al almacenamiento intermedio de datos. DB2 asigna espacio para el almacenamiento intermedio.
reserve	void	Reservado para una utilización futura.

Sintaxis de la API y de las estructuras de datos

```
typedef struct Data
{
   sqlint32 obj_num;
   sqlint32 buff_size;
   sqlint32 actual_buff_size;
   void *dataptr;
   void *reserve;
} Data;
```

Return_code

Contiene el código de retorno y una breve explicación del error devuelto a DB2 por el plugin de copia de seguridad y de restauración del almacenamiento de proveedor. Esta estructura de datos es utilizada por todas las API del plugin de almacenamiento de proveedor.

Tabla 25. Campos de la estructura Return_code

Nombre de campo	Tipo de datos	Descripción
return_code (ver nota más abajo)	sqlint32	Código de retorno de la API de proveedor.
descripción	char	Breve descripción del código de retorno.
reserve	void	Reservado para una utilización futura.

Nota: Este código de retorno es específico del proveedor y no es el mismo que el valor devuelto por diversas API de DB2. Consulte las descripciones de cada API para conocer los códigos de retorno que se aceptan desde productos de proveedor.

Sintaxis de la API y de las estructuras de datos

```
typedef struct Return_code
{
        sqlint32 return_code;
        char description[SQLUV_COMMENT_LEN];
        void *reserve;
} Return_code;
```

Las API de DB2 para utilizar la compresión con operaciones de copia de seguridad y restauración

DB2 proporciona varias API que productos de compresión de terceros pueden utilizar para comprimir y descomprimir imágenes de copia de seguridad. Esta interfaz está diseñada para ampliar o sustituir la biblioteca de compresión que se puede utilizar como componente estándar de DB2. La interfaz del plugin de compresión se puede utilizar con las API de copia de seguridad y restauración de DB2 o con los plugins de copia de seguridad y restauración para dispositivos de almacenamiento de proveedor.

DB2 define un conjunto de prototipos de API que proporcionan una interfaz de uso general para la compresión y descompresión que puede ser utilizada por muchos proveedores. El proveedor debe proporcionar estas API en una biblioteca compartida en los sistemas Linux y UNIX, o en una DLL en el sistema operativo Windows. Cuando DB2 invoca las API, se carga la biblioteca compartida o DLL especificada por la rutina de copia de seguridad o restauración que ha realizado la llamada, y se invocan las API proporcionadas por el proveedor para realizar las tareas solicitadas.

Visión general

Se definen ocho API para interaccionar con DB2 y el producto de proveedor:

- InitCompression - Inicializar la biblioteca de compresión
- GetSavedBlock - Obtener bloque de proveedor para imagen de copia de seguridad
- Compress - Comprimir un bloque de datos
- GetMaxCompressedSize - Calcular el tamaño máximo posible del almacenamiento intermedio
- TermCompression - Terminar la biblioteca de compresión
- InitDecompression - Inicializar la biblioteca de descompresión
- Decompress - Descomprimir un bloque de datos
- TermDecompression - Terminar la biblioteca de descompresión

DB2 proporciona la definición para la estructura COMPR_DB2INFO; el proveedor proporciona las definiciones para cada una de las demás estructuras y API para utilizar la compresión con la copia de seguridad y la restauración. Las estructuras, los prototipos y las constantes están definidos en el archivo sqlucompr.h, que se proporciona con DB2.

Estas API son invocadas por DB2 y deben ser proporcionadas por el producto de proveedor en una biblioteca compartida en sistemas Linux y UNIX, o en una DLL en el sistema operativo Windows.

Nota: El código de la biblioteca compartida o DLL se ejecutará como parte del código del motor de base de datos. Por tanto, el código debe ser reentrante y estar totalmente depurado. Una función de comportamiento irregular podría poner en peligro la integridad de los datos de la base de datos.

Secuencia de llamada de ejemplo

Para la copia de seguridad, DB2 emite la siguiente secuencia de llamadas para cada sesión:

```
InitCompression
```

seguido por 0 a 1

```
GetMaxCompressedSize
Comprimir
```

seguido por 1

```
TermCompress
```

Para la restauración, la secuencia de llamadas para cada sesión es:

```
InitDecompression
```

seguido por 1 a n

```
Decompress
```

seguido por 1

```
TermCompression
```

Códigos de retorno de la interfaz del plugin de compresión

A continuación se muestran los códigos de retorno que las API podrían devolver. Excepto donde se especifique, DB2 termina la copia de seguridad o restauración cuando se devuelve un código de retorno cualquiera distinto de cero.

```
SQLUV_OK
```

0

Operación satisfactoria

```
SQLUV_BUFFER_TOO_SMALL
```

100

El almacenamiento intermedio de destino es demasiado pequeño. Cuando este código de retorno aparezca durante una copia de seguridad, el campo tgtAct indica el tamaño necesario para comprimir el objeto. DB2 reintentará la operación con un almacenamiento intermedio que sea como mínimo tan grande como se especifique. Cuando este código de retorno aparezca durante una restauración, la operación fallará.

```
SQLUV_PARTIAL_BUFFER
```

101

Un almacenamiento intermedio se ha comprimido parcialmente. Cuando este código de retorno aparezca durante una copia de seguridad, el campo srcAct indicará la cantidad real de datos que se han comprimido realmente y el campo tgtAct indicará el tamaño real de los datos comprimidos. Cuando este código de retorno aparezca durante una restauración, la operación fallará.

SQLUV_NO_MEMORY

102

Falta de memoria

SQLUV_EXCEPTION

103

Se ha emitido una señal o excepción en el código.

SQLUV_INTERNAL_ERROR

104

Se ha detectado un error interno.

La diferencia entre SQLUV_BUFFER_TOO_SMALL y SQLUV_PARTIAL_BUFFER es que cuando se devuelve SQLUV_PARTIAL_BUFFER, DB2 considera que los datos del almacenamiento intermedio de salida son válidos.

COMPR_CB

La biblioteca de plugins utiliza internamente esta estructura como bloque de control. Contiene datos que se utilizan internamente en las API de compresión y descompresión. DB2 pasa esta estructura a cada llamada que realiza a la biblioteca de plugins, pero es la biblioteca la que se encarga de todos los aspectos de la estructura, incluyendo la definición de los parámetros y la gestión de memoria de la estructura.

Sintaxis de la API y de las estructuras de datos

```
struct COMPR_CB;
```

COMPR_DB2INFO

Describe el entorno DB2. DB2 asigna y define esta estructura y la pasa como un parámetro a las API InitCompression y InitDecompression. Esta estructura describe la base de datos de la que se está haciendo una copia de seguridad o restaurando y proporciona detalles sobre el entorno DB2 donde se está realizando la operación. Los parámetros dbalias, instance, node, catnode y timestamp se utilizan para designar la imagen de copia de seguridad.

Sintaxis de la API y de las estructuras de datos

```
struct COMPR_DB2INFO {
   char tag[16];
   db2Uint32 version;
   db2Uint32 size;
   char dbalias[SQLU_ALIAS_SZ+1];
   char instance[SQL_INSTNAME_SZ+1];
   SQL_PDB_NODE_TYPE node;
   SQL_PDB_NODE_TYPE catnode;
   char timestamp[SQLU_TIME_STAMP_LEN+1];
```

```
   db2Uint32 bufferSize;
   db2Uint32 options;
   db2Uint32 bkOptions;
   db2Uint32 db2Version;
   db2Uint32 platform;
   db2int32 comprOptionsByteOrder;
   db2Uint32 comprOptionsSize;
   void *comprOptions;
   db2Uint32 savedBlockSize;
   void *savedBlock;
};
```

Parámetros de la estructura de datos COMPR_DB2INFO

tag Se utiliza como anuncio para la estructura. Su valor es siempre la serie "COMPR_DB2INFO \0".

version
Indica qué versión de la estructura se está utilizando para que las API puedan indicar la presencia de campos adicionales. Actualmente la versión es la 1. En el futuro, es posible que se añadan más parámetros a esta estructura.

size Especifica el tamaño de la estructura COMPR_DB2INFO, en bytes.

dbalias
Alias de la base de datos. Para las operaciones de restauración, dbalias hace referencia al alias de la base de datos fuente.

instance
Nombre de la instancia.

nodo Número de nodo.

catnode
Número de nodo de catálogo.

timestamp
Indicación de fecha y hora de la imagen que se está copiando o restaurando.

bufferSize
Especifica el tamaño de un almacenamiento intermedio de transferencia (expresado en páginas de 4K).

options
Es el parámetro iOptions especificado en la API db2Backup o la API db2Restore.

bkOptions
Para las operaciones de restauración, especifica el parámetro iOptions que se utilizó en la API db2Backup cuando se creó la copia de seguridad. Para las operaciones de copia de seguridad, su valor se establece en cero.

db2Version
Especifica la versión del motor de DB2.

platform
Especifica la plataforma en la que el motor de DB2 se ejecuta. El valor será uno de los listados en sqlmon.h (ubicado en el directorio de inclusión).

comprOptionsByteOrder
Especifica el orden de bytes utilizado en el cliente donde se está ejecutando la API. DB2 no hará ninguna interpretación ni conversión de los datos pasados como comprOptions, por lo que este campo se debe utilizar para

determinar si se debe invertir el orden de bytes de los datos antes de utilizarlos. Cualquier conversión debe ser realizada por la propia biblioteca de plugins.

comprOptionsSize
Especifica el valor del parámetro piComprOptionsSize en las API db2Backup y db2Restore.

comprOptions
Especifica el valor del parámetro piComprOptions en las API db2Backup y db2Restore.

savedBlockSize
Tamaño de savedBlock, en bytes.

savedBlock
DB2 permite a la biblioteca de plugins guardar un bloque arbitrario de datos en la imagen de copia de seguridad. Si ese bloque de datos se ha guardado con una copia de seguridad determinada, el bloque de datos se devolverá en estos campos en la operación de restauración. Para operaciones de copia de seguridad, estos campos están establecidos en cero.

COMPR_PIINFO

La biblioteca de plugins utiliza esta estructura para describirse a sí misma ante DB2. DB2 asigna e inicializa esta estructura, y la biblioteca de plugins define los campos esenciales en la llamada a la API InitCompression.

Sintaxis de la API y de las estructuras de datos

```
struct COMPR_PIINFO {
   char tag[16];
   db2Uint32 version;
   db2Uint32 size;
   db2Uint32 useCRC;
   db2Uint32 useGran;
   db2Uint32 useAllBlocks;
   db2Uint32 savedBlockSize;
};
```

Parámetros de la estructura de datos COMPR_PIINFO

tag Se utiliza como anuncio para la estructura. (Lo establece DB2.) Su valor es siempre la serie `"COMPR_PIINFO  \0"`.

version
Indica qué versión de la estructura se está utilizando para que las API puedan indicar la presencia de campos adicionales. Actualmente, la versión es 1.

(Lo establece DB2.) En el futuro, es posible que se añadan más campos a esta estructura.

size Indica el tamaño de la estructura COMPR_PIINFO (en bytes). (Lo establece DB2.)

useCRC
DB2 permite que los plugins de compresión utilicen un valor de comprobación de redundancia cíclica (CRC) de 32 bits o de suma de comprobación para validad la integridad de los datos sujetos a compresión y descompresión.

La biblioteca, si utiliza una comprobación de esta clase, asigna el valor 1 a este campo; de lo contrario, asigna el valor 0.

useGran
Si la rutina de compresión es capaz de comprimir los datos según incrementos de tamaño arbitrario, la biblioteca establece este campo en 1. Si la rutina de compresión solamente comprime datos según incrementos de bytes, la biblioteca establece este campo en 0. Vea la descripción del parámetro srcGran de la API Compress para conocer detalles sobre los efectos de definir este indicador.

En el caso de las operaciones de restauración, este parámetro no se tiene en cuenta.

useAllBlocks
Especifica si DB2 debe realizar una copia de seguridad de un bloque de datos comprimido que sea mayor que el bloque original sin comprimir. Por omisión, DB2 almacena los datos sin comprimir si la versión comprimida es mayor, pero en algunos casos a la biblioteca de plugins le interesará que se haga una copia de seguridad de los datos comprimidos. Si DB2 debe guardar la versión comprimida de los datos para todos los bloques, la biblioteca establecerá este valor en 1. Si DB2 debe guardar la versión comprimida de los datos solamente cuando sea menor que los datos originales, la biblioteca establecerá este valor en 0. Para las operaciones de restauración, este campo no se tiene en cuenta.

savedBlockSize
DB2 permite a la biblioteca de plugins guardar un bloque arbitrario de datos en la imagen de copia de seguridad. Si ese bloque de datos se debe guardar con una determinada copia de seguridad, la biblioteca establecerá el valor este parámetro en el tamaño del bloque que se debe asignar para estos datos. (Los datos reales se transferirán a DB2 en una llamada a la API posterior.) Si no se van a guardar datos, la biblioteca de plugin establecerá el valor de este parámetro en cero. En el caso de las operaciones de restauración, este parámetro no se tiene en cuenta.

Compress - Comprimir un bloque de datos

Comprime un bloque de datos. El parámetro src señala hacia un bloque de datos cuyo tamaño en bytes es srcLen. El parámetro tgt señala hacia un almacenamiento intermedio cuyo tamaño en bytes es tgtSize. La biblioteca de plugins comprime los datos situados en la dirección src y escribe los datos comprimidos en la dirección tgt del almacenamiento intermedio. La cantidad real de datos sin comprimir que se comprimieron se almacena en srcAct. El tamaño real de los datos comprimidos se devuelve en tgtAct.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

`sqlucompr.h`

Sintaxis de la API y de las estructuras de datos

```
int Compress(
      struct COMPR_CB *pCB,
      const char *src,
      db2int32 srcSize,
      db2Uint32 srcGran,
      char *tgt,
      db2int32 tgtSize,
      db2int32 *srcAct,
      db2int32 *tgtAct,
      db2Uint32 *tgtCRC);
```

Parámetros de la API Compress

pCB Entrada. Es el bloque de control devuelto por la llamada a la API InitCompression.

src Entrada. Puntero al bloque de datos que se debe comprimir.

srcLen Entrada. Tamaño en bytes del bloque de datos que se debe comprimir.

srcGran
Entrada. Si la biblioteca devolvió un valor de 1 para piInfo->useGran, srcGran especifica el valor log2 del tamaño de página de los datos. (Por ejemplo, si el tamaño de página de los datos es de 4096 bytes, srcGran es de 12.) La biblioteca garantiza que la cantidad de datos comprimidos realmente (srcAct) sea un múltiplo exacto de este tamaño de página. Si la biblioteca establece el distintivo useGran, DB2 es capaz de utilizar un algoritmo más eficaz para ajustar los datos comprimidos a la imagen de copia de seguridad. Esto significa que el rendimiento del plugin será mejor y que la imagen de copia de seguridad comprimida será más pequeña. Si la biblioteca devuelta un valor de 0 para piInfo->srcGran, la granularidad es de 1 byte.

tgt Entrada y salida. Almacenamiento intermedio de destino para datos comprimidos. DB2 suministrará este almacenamiento intermedio de destino y el plugin comprimirá los datos situados en src y escribirá aquí los datos comprimidos.

tgtSize
Entrada. Tamaño, en bytes, del almacenamiento intermedio de destino.

srcAct Salida. Cantidad real, en bytes, de datos sin comprimir procedentes de src que se han comprimido.

tgtAct Salida. Cantidad real de bytes de datos comprimidos almacenados en tgt.

tgtCRC
Salida. Si la biblioteca ha devuelto un valor de 1 para piInfo->useCRC, el valor CRC del bloque no comprimido se devuelve como tgtCRC. Si la biblioteca ha devuelto un valor de 0 para piInfo->useCRC, tgtCRC será un puntero nulo.

Decompress - Descomprimir un bloque de datos

Descomprime un bloque de datos. El parámetro src señala hacia un bloque de datos cuya longitud en bytes es srcLen. El parámetro tgt señala hacia un almacenamiento intermedio cuyo tamaño en bytes es tgtSize. La biblioteca de plugins descomprime los datos situados en la dirección src y escribe los datos descomprimidos en el almacenamiento intermedio situado en la dirección tgt. El tamaño real de los datos descomprimidos se devuelve en tgtLen. Si la biblioteca

devolvió el valor 1 para piInfo->useCRC, el valor CRC del bloque sin comprimir se devuelve como tgtCRC. Si la biblioteca devolvió el valor 0 para piInfo->useCRC, tgtLen será un puntero nulo.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlucompr.h
```

Sintaxis de la API y de las estructuras de datos

```
int Decompress(
      struct COMPR_CB *pCB,
      const char *src,
      db2int32 srcSize,
      char *tgt,
      db2int32 tgtSize,
      db2int32 *tgtLen,
      db2Uint32 *tgtCRC);
```

Parámetros de la API Decompress

pCB Entrada. Es el bloque de control devuelto por la llamada a la API InitDecompression.

src Entrada. Puntero al bloque de datos que se debe descomprimir.

srcLen Entrada. Tamaño, en bytes, del bloque de datos que se debe descomprimir.

tgt Entrada y salida. Almacenamiento intermedio de destino para datos descomprimidos. DB2 suministrará este almacenamiento intermedio de destino y el plugin descomprimirá los datos situados en src y escribirá aquí los datos descomprimidos.

tgtSize
Entrada. Tamaño, en bytes, del almacenamiento intermedio de destino.

tgtLen Salida. Cantidad real de bytes de datos descomprimidos almacenados en tgt.

tgtCRC
Salida. Si la biblioteca ha devuelto un valor de 1 para piInfo->useCRC, el valor CRC del bloque no comprimido se devuelve como tgtCRC. Si la biblioteca ha devuelto un valor de 0 para piInfo->useCRC, tgtCRC será un puntero nulo.

GetMaxCompressedSize - Calcular el tamaño máximo posible del almacenamiento intermedio

Calcula el tamaño máximo posible del almacenamiento intermedio necesario para comprimir un bloque de datos. srcLen indica el tamaño de un bloque de datos que se va a comprimir. La biblioteca devuelve en tgtLen el tamaño máximo teórico del almacenamiento intermedio después de la compresión.

DB2 utilizará el valor devuelto como tgtLen para optimizar la utilización interna que DB2 hace de la memoria. La consecuencia negativa de no calcular un valor o de calcular un valor incorrecto es que DB2 tendrá que llamar a la API Compress más de una vez para un mismo bloque de datos, o bien que pueda desaprovechar memoria de la pila de programa de utilidad. DB2 todavía creará correctamente la copia de seguridad, sean cuales sean los valores devueltos.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlucompr.h
```

Sintaxis de la API y de las estructuras de datos

```
int GetMaxCompressedSize(
      struct COMPR_CB *pCB,
      db2Uint32 srcLen);
```

Parámetros de la API GetMaxCompressedSize

pCB Entrada. Es el bloque de control devuelto por la llamada a la API InitCompression.

srcLen Entrada. Tamaño, en bytes, de un bloque de datos que está a punto de comprimirse.

GetSavedBlock - Obtener el proveedor del bloque de datos para la imagen de copia de seguridad

Obtiene el bloque de datos, específico del proveedor, que se debe guardar con la imagen de copia de seguridad. Si la biblioteca ha devuelto un valor distinto de cero para piInfo->savedBlockSize, DB2 llamará a GetSavedBlock utilizando ese valor para blockSize. La biblioteca de plugins escribe datos del tamaño indicado en la memoria referenciada por los datos. Se llama a esta API durante el proceso de datos inicial realizado por el primer proceso db2bm, solo para copia de seguridad. Aunque se especifique un paralelismo > 1 en la API db2Backup, solo se llama a esta API una vez por cada copia de seguridad.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlucompr.h
```

Sintaxis de la API y de las estructuras de datos

```
int GetSavedBlock(
      struct COMPR_CB *pCB,
      db2Uint32 blockSize,
      void *data);
```

Parámetros de la API GetSavedBlock

pCB Entrada. Es el bloque de control devuelto por la llamada a la API InitCompression.

blocksize
Entrada. Es el tamaño del bloque devuelto en piInfo->savedBlockSize por la llamada a la API InitCompression.

data Salida. Es el bloque de datos, específico del proveedor, que se debe guardar con la imagen de copia de seguridad.

InitCompression - Inicializar la biblioteca de compresión

Inicializa la biblioteca de compresión. DB2 pasará las estructuras db2Info y piInfo. La biblioteca rellena los parámetros apropiados de piInfo, asigna el bloque control (pCB) y devuelve un puntero a la memoria asignada.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlucompr.h
```

Sintaxis de la API y de las estructuras de datos

```
int InitCompression(
      const struct COMPR_DB2INFO
                       *db2Info,
      struct COMPR_PIINFO
                       *piInfo,
      struct COMPR_CB  **pCB);
```

Parámetros de la API InitCompression

db2Info
Entrada. Describe la base de datos de la que se está haciendo una copia de seguridad y proporciona detalles sobre el entorno DB2 donde se está realizando la operación.

piInfo Salida. La biblioteca de plugins utiliza esta estructura para describirse a sí misma ante DB2. DB2 asigna e inicializa esta estructura y la biblioteca de plugins rellena los parámetros clave.

pCB Salida. Es el bloque de control utilizado por la biblioteca de compresión. La biblioteca de plugins es responsable de la gestión de memoria de la estructura.

InitDecompression - Inicializar la biblioteca de descompresión

Inicializa la biblioteca de descompresión. DB2 pasará la estructura db2Info. La biblioteca asigna el bloque de control (pCB) y devuelve un puntero a la memoria asignada.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlucompr.h
```

Sintaxis de la API y de las estructuras de datos

```
int InitDecompression(
      const struct COMPR_DB2INFO
                       *db2Info,
      struct COMPR_CB  **pCB);
```

Parámetros de la API InitDecompression

db2Info
Entrada. Describe la base de datos de la que se está haciendo una copia de seguridad y proporciona detalles sobre el entorno DB2 donde se está realizando la operación.

pCB Salida. Es el bloque de control utilizado por la biblioteca de descompresión. La biblioteca de plugins es responsable de la gestión de memoria de la estructura.

TermCompression - Detener la biblioteca de compresión

Termina la biblioteca de compresión. La biblioteca libera la memoria utilizada para pCB.

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlucompr.h
```

Sintaxis de la API y de las estructuras de datos

```
int TermCompression(
      struct COMPR_CB *pCB);
```

Parámetros de la API TermCompression

pCB Entrada. Es el bloque de control devuelto por la llamada a la API InitCompression.

TermDecompression - Detener la biblioteca de descompresión

Termina la biblioteca de descompresión. La biblioteca libera la memoria utilizada para pCB. La biblioteca gestionará toda la memoria utilizada internamente por las API de compresión. La biblioteca de plugins también gestionará la memoria utilizada por la estructura COMPR_CB. DB2 gestionará la memoria utilizada para los almacenamientos intermedios de datos (parámetros src y tgt de las API de descompresión).

Autorización

Ninguna

Conexión necesaria

Ninguna

Archivo de inclusión de la API

```
sqlucompr.h
```

Sintaxis de la API y de las estructuras de datos

```
int TermDecompression(
      struct COMPR_CB *pCB);
```

Parámetros de la API TermDecompression

pCB Entrada. Es el bloque de control devuelto por la llamada a la API InitDecompression.

Capítulo 9. Estructuras de datos que utilizan las API

db2HistoryData

Esta estructura se utiliza para devolver información después de una llamada a la API db2HistoryGetEntry.

Tabla 26. Campos de la estructura db2HistoryData

Nombre de campo	Tipo de datos	Descripción
ioHistDataID	char(8)	Identificador de estructura de 8 bytes y ″captador de atención″ de vuelcos de almacenamiento. El único valor válido es SQLUHINF. No existe ninguna definición simbólica para esta serie.
oObjectPart	db2Char	Los 14 primeros caracteres son una indicación de fecha y hora con el formato *aaaammddhhmmss*, que indica cuándo se ha iniciado la operación. Los 3 caracteres siguientes son un número de secuencia. Cada operación de copia de seguridad puede dar como resultado varias entradas en este archivo cuando la imagen de copia de seguridad se guarda en varios archivos o en varias cintas. El número de secuencia permite especificar varias ubicaciones. Las operaciones de restauración y carga tienen una sola entrada en este archivo, que corresponde al número de secuencia ″001″ de la copia de seguridad correspondiente. La indicación de fecha y hora, combinada con el número de secuencia, debe ser exclusiva.
oEndTime	db2Char	Indicación de fecha y hora con el formato *aaaammddhhmmss*, que indica cuándo se ha completado la operación.
oFirstLog	db2Char	ID del archivo de anotaciones cronológicas más antiguo (de S0000000 a S9999999): • Necesario para aplicar la recuperación en avance de una copia de seguridad en línea • Necesario para aplicar la recuperación en avance de una copia de seguridad fuera de línea • Aplicado tras la restauración de una copia de seguridad en el nivel del espacio de tablas o la base de datos completa que era actual cuando se inició la carga.

Tabla 26. Campos de la estructura db2HistoryData (continuación)

Nombre de campo	Tipo de datos	Descripción
oLastLog	db2Char	ID del archivo de anotaciones cronológicas más reciente (de S0000000 a S9999999): • Necesario para aplicar la recuperación en avance de una copia de seguridad en línea • Necesario para aplicar la recuperación en avance hasta el punto en el tiempo actual de una copia de seguridad fuera de línea • Aplicado tras la restauración de una copia de seguridad completa en el nivel del espacio de tablas o la base de datos que era actual cuando finalizó la operación de carga (será el mismo que **oFirstLog** si no se aplica la recuperación en avance).
oID	db2Char	Identificador de copia de seguridad o tabla exclusivo.
oTableQualifier	db2Char	Calificador de tabla.
oTableName	db2Char	Nombre de tabla.
oLocation	db2Char	Para copias de seguridad y copias de carga, este campo indica dónde se han guardado los datos. Para operaciones que requieren varias entradas en el archivo, el número de secuencia definido por el parámetro **oObjectPart** identifica qué parte de la copia de seguridad se encuentra en la ubicación especificada. Para operaciones de restauración y carga, la ubicación identifica siempre dónde se ha guardado el primer componente de los datos restaurados o cargados (correspondientes a la secuencia ″001″ para copias de seguridad multicomponente). Los datos de **oLocation** se interpretan de forma distinta, dependiendo del parámetro **oDeviceType**: • Para discos o disquetes (D o K), un nombre de archivo totalmente calificado. • Para cintas (T), una etiqueta de volumen. • Para TSM (A y F), el nombre/vía de acceso de biblioteca del proveedor que ejecutó la copia de seguridad. • Para salidas de usuario u otros (U u O), texto en formato libre.
oComment	db2Char	Comentario de texto en formato libre.
oCommandText	db2Char	Texto del mandato o DDL.
oLastLSN	SQLU_LSN	Último número de secuencia de anotaciones cronológicas.
oEID	Estructura	Identificador de entrada exclusivo.
poEventSQLCA	Estructura	sqlca resultante del evento de registrado.
poTablespace	db2Char	Lista de nombres de espacio de tablas.
iNumTablespaces	db2Uint32	Número de entradas de la lista **poTablespace** que están disponibles para ser utilizadas por la API db2HistoryGetEntry.

Tabla 26. Campos de la estructura db2HistoryData (continuación)

Nombre de campo	Tipo de datos	Descripción
oNumTablespaces	db2Uint32	Número de entradas de la lista **poTablespace** utilizadas por la API db2HistoryGetEntry. Cada copia de seguridad de espacio de tablas contiene uno o varios espacios de tablas. Cada operación de restauración de espacio de tablas sustituye uno o varios espacios de tablas. Si este campo no es cero (lo que indica una copia de seguridad o restauración a nivel de espacio de tablas), las líneas siguientes de este archivo contendrán el nombre del espacio de tablas copiado o restaurado, representado por una serie de 18 caracteres. En cada línea aparece un nombre de espacio de tablas.
oOperation	char	Consulte la Tabla 27.
oObject	char	Granularidad de la operación: D para base de datos completa, P para espacio de tablas, T para tabla, R para partición de rango e I para índice.
oOptype	char	Consulte la Tabla 28 en la página 522.
oStatus	char	Estado de entrada: A para activo; I para inactivo; E para caducado; D para suprimido; y X para no suprimir.
oDeviceType	char	Tipo de dispositivo. Este campo determina cómo se interpreta el campo **oLocation**: A para TSM, C para cliente, D para disco, F para copia de seguridad selectiva, K para disquete, L para local, O para otro (para soporte de dispositivo de otro proveedor), P para conexión, Q para cursor, S para servidor, T para cinta y U para salida de usuario.

Tabla 27. Valores de **oOperation** *válidos en la estructura db2HistoryData*

Valor	Descripción	Definición C	Definición COBOL/FORTRAN
A	añadir espacio de tablas	DB2HISTORY_OP_ADD_ TABLESPACE	DB2HIST_OP_ADD_ TABLESPACE
B	copia de seguridad	DB2HISTORY_OP_BACKUP	DB2HIST_OP_BACKUP
C	copia de carga	DB2HISTORY_OP_LOAD_COPY	DB2HIST_OP_LOAD_COPY
D	tabla descartada	DB2HISTORY_OP_DROPPED_ TABLE	DB2HIST_OP_DROPPED_ TABLE
F	avance	DB2HISTORY_OP_ROLLFWD	DB2HIST_OP_ROLLFWD
G	reorganizar tabla	DB2HISTORY_OP_REORG	DB2HIST_OP_REORG
L	load	DB2HISTORY_OP_LOAD	DB2HIST_OP_LOAD
N	redenominar espacio de tablas	DB2HISTORY_OP_REN_ TABLESPACE	DB2HIST_OP_REN_ TABLESPACE
O	descartar espacio de tablas	DB2HISTORY_OP_DROP_ TABLESPACE	DB2HIST_OP_DROP_ TABLESPACE
Q	inmovilizar	DB2HISTORY_OP_QUIESCE	DB2HIST_OP_QUIESCE
R	restaurar	DB2HISTORY_OP_RESTORE	DB2HIST_OP_RESTORE
T	alterar espacio de tablas	DB2HISTORY_OP_ALT_ TABLESPACE	DB2HIST_OP_ALT_TBS

Tabla 27. Valores de **oOperation** *válidos en la estructura db2HistoryData (continuación)*

Valor	Descripción	Definición C	Definición COBOL/FORTRAN
U	descargar	DB2HISTORY_OP_UNLOAD	DB2HIST_OP_UNLOAD
X	archivador de anotaciones	DB2HISTORY_OP_ARCHIVE_ LOG	DB2HIST_OP_ARCHIVE_LOG

Tabla 28. Valores de **oOptype** *válidos en la estructura db2HistData*

oOperation	oOptype	Descripción	Definición C/COBOL/FORTRAN
B	F N I O D E	fuera de línea, en línea, fuera de línea incremental, en línea incremental, fuera de línea delta, en línea delta	DB2HISTORY_OPTYPE_OFFLINE, DB2HISTORY_OPTYPE_ONLINE, DB2HISTORY_OPTYPE_INCR_OFFLINE, DB2HISTORY_OPTYPE_INCR_ONLINE, DB2HISTORY_OPTYPE_DELTA_OFFLINE, DB2HISTORY_OPTYPE_DELTA_ONLINE
F	E P	fin de anotaciones, punto en el tiempo	DB2HISTORY_OPTYPE_EOL, DB2HISTORY_OPTYPE_PIT
G	F N	fuera de línea, en línea	DB2HISTORY_OPTYPE_OFFLINE, DB2HISTORY_OPTYPE_ONLINE
L	I R	insertar, sustituir	DB2HISTORY_OPTYPE_INSERT, DB2HISTORY_OPTYPE_REPLACE
Q	S U X Z	inmovilizar compartimiento, inmovilizar actualización, inmovilizar exclusividad, inmovilizar restablecimiento	DB2HISTORY_OPTYPE_SHARE, DB2HISTORY_OPTYPE_UPDATE, DB2HISTORY_OPTYPE_EXCL, DB2HISTORY_OPTYPE_RESET
R	F N I O R	fuera de línea, en línea, fuera de línea incremental, en línea incremental, reconstruir	DB2HISTORY_OPTYPE_OFFLINE, DB2HISTORY_OPTYPE_ONLINE, DB2HISTORY_OPTYPE_INCR_OFFLINE, DB2HISTORY_OPTYPE_INCR_ONLINE, DB2HISTORY_OPTYPE_REBUILD
T	C R	añadir contenedores, reequilibrar	DB2HISTORY_OPTYPE_ADD_CONT, DB2HISTORY_OPTYPE_REB
X	N P M F 1 2	mandato de anotaciones de archivador, vía de acceso de anotaciones primarias, vía de acceso de anotaciones de duplicación, vía de acceso de anomalía de archivador, método 1 de archivador de anotaciones, método 2 de archivador de anotaciones	DB2HISTORY_OPTYPE_ARCHIVE_CMD, DB2HISTORY_OPTYPE_PRIMARY, DB2HISTORY_OPTYPE_MIRROR, DB2HISTORY_OPTYPE_ARCHFAIL, DB2HISTORY_OPTYPE_ARCH1, DB2HISTORY_OPTYPE_ARCH2

Tabla 29. Campos de la estructura db2HistoryEID

Nombre de campo	Tipo de datos	Descripción
ioNode ioHID	SQL_PDB_NODE_TYPE db2Uint32	Número de nodo. ID de entrada del archivo histórico local.

Sintaxis de la API y de las estructuras de datos

```
typedef SQL_STRUCTURE db2HistoryData
{
   char ioHistDataID[8];
   db2Char oObjectPart;
   db2Char oEndTime;
   db2Char oFirstLog;
   db2Char oLastLog;
   db2Char oID;
   db2Char oTableQualifier;
   db2Char oTableName;
   db2Char oLocation;
   db2Char oComment;
   db2Char oCommandText;
   SQLU_LSN oLastLSN;
   db2HistoryEID oEID;
   struct sqlca *poEventSQLCA;
   struct db2Char *poTablespace;
   db2Uint32 iNumTablespaces;
   db2Uint32 oNumTablespaces;
   char oOperation;
   char oObject;
   char oOptype;
   char oStatus;
   char oDeviceType;
} db2HistoryData;

typedef SQL_STRUCTURE db2Char
{
   char *pioData;
   db2Uint32 iLength;
   db2Uint32 oLength;
} db2Char;

typedef SQL_STRUCTURE db2HistoryEID
{
   SQL_PDB_NODE_TYPE ioNode;
   db2Uint32 ioHID;
} db2HistoryEID;
```

Parámetros de la estructura de datos db2Char

pioData
Puntero a un almacenamiento intermedio de datos de caracteres. Si el valor es NULL, no se devolverán datos.

iLength
Entrada. Tamaño del almacenamiento intermedio **pioData**.

oLength
Salida. Número de caracteres válidos de datos contenidos en el almacenamiento intermedio **pioData**.

Parámetros de la estructura de datos db2HistoryEID

ioNode
Este parámetro se puede utilizar como parámetro de entrada o salida. Indica el número de nodo.

ioHID Este parámetro se puede utilizar como parámetro de entrada o salida. Indica el ID de entrada del archivo histórico local.

sql_authorizations

Nota: Esta estructura está en desuso porque sólo se utiliza para la compatibilidad con versiones anteriores para la API sqluadau().

Esta estructura se utiliza para devolver información después de una llamada a la API sqluadau. El tipo de datos de todos los campos es SMALLINT. La primera mitad de la tabla siguiente contiene las autorizaciones otorgadas directamente al usuario. La segunda mitad de la tabla contiene las autorizaciones otorgadas a los grupos a los que pertenece el usuario.

Tabla 30. Campos de la estructura SQL_AUTHORIZATIONS

Nombre de campo	Descripción
SQL_AUTHORIZATIONS_LEN	Tamaño de la estructura.
SQL_SYSADM_AUTH	Autorización SYSADM.
SQL_SYSCTRL_AUTH	Autorización SYSCTRL.
SQL_SYSMAINT_AUTH	Autorización SYSMAINT.
SQL_DBADM_AUTH	Autorización DBADM.
SQL_CREATETAB_AUTH	Autorización CREATETAB.
SQL_CREATET_NOT_FENC_AUTH	Autorización CREATE_NOT_FENCED.
SQL_BINDADD_AUTH	Autorización BINDADD.
SQL_CONNECT_AUTH	Autorización CONNECT.
SQL_IMPLICIT_SCHEMA_AUTH	Autorización IMPLICIT_SCHEMA.
SQL_LOAD_AUTH	Autorización LOAD.
SQL_SYSADM_GRP_AUTH	El usuario pertenece a un grupo que tiene autorización SYSADM.
SQL_SYSCTRL_GRP_AUTH	El usuario pertenece a un grupo que tiene autorización SYSCTRL.
SQL_SYSMAINT_GRP_AUTH	El usuario pertenece a un grupo que tiene autorización SYSMAINT.
SQL_DBADM_GRP_AUTH	El usuario pertenece a un grupo que tiene autorización DBADM.
SQL_CREATETAB_GRP_AUTH	El usuario pertenece a un grupo que tiene autorización CREATETAB.
SQL_CREATE_NON_FENC_GRP_AUTH	El usuario pertenece a un grupo que tiene autorización CREATE_NOT_FENCED.
SQL_BINDADD_GRP_AUTH	El usuario pertenece a un grupo que tiene autorización BINDADD.
SQL_CONNECT_GRP_AUTH	El usuario pertenece a un grupo que tiene autorización CONNECT.

Tabla 30. Campos de la estructura SQL_AUTHORIZATIONS (continuación)

Nombre de campo	Descripción
SQL_IMPLICIT_SCHEMA_GRP_AUTH	El usuario pertenece a un grupo que tiene autorización IMPLICIT_SCHEMA.
SQL_LOAD_GRP_AUTH	El usuario pertenece a un grupo que tiene autorización LOAD.
SQL_CREATE_EXT_RTN_AUTH	Autorización CREATE_EXTERNAL_ROUTINE.
SQL_CREATE_EXT_RTN_GRP_AUTH	El usuario pertenece a un grupo que tiene autorización CREATE_EXTERNAL_ROUTINE.
SQL_QUIESCE_CONNECT_AUTH	Autorización QUIESCE CONNECT.
SQL_QUIESCE_CONNECT_GRP_AUTH	El usuario pertenece a un grupo que tiene autorización QUIESCE CONNECT.
SQL_SECURITY_ADMIN_AUTH	Autorización SECADM.
SQL_SECURITY_ADMIN_GRP_AUTH	El usuario pertenece a un grupo que tiene autorización SECADM.
SQL_SYSMON_AUTH	Autorización SYSMON.
SQL_SYSMON_GRP_AUTH	El usuario pertenece a un grupo que tiene autorización SYSMON.

Nota: SYSADM, SYSMAINT, SYSMON y SYSCTRL son sólo autorizaciones indirectas y no se pueden otorgar directamente al usuario. Solamente están disponibles a través de los grupos a los que pertenece el usuario.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sql_authorizations
{
   short sql_authorizations_len;
   short sql_sysadm_auth;
   short sql_dbadm_auth;
   short sql_createtab_auth;
   short sql_bindadd_auth;
   short sql_connect_auth;
   short sql_sysadm_grp_auth;
   short sql_dbadm_grp_auth;
   short sql_createtab_grp_auth;
   short sql_bindadd_grp_auth;
   short sql_connect_grp_auth;
   short sql_sysctrl_auth;
   short sql_sysctrl_grp_auth;
   short sql_sysmaint_auth;
   short sql_sysmaint_grp_auth;
   short sql_create_not_fenc_auth;
   short sql_create_not_fenc_grp_auth;
   short sql_implicit_schema_auth;
   short sql_implicit_schema_grp_auth;
   short sql_load_auth;
   short sql_load_grp_auth;
   short sql_create_ext_rtn_auth;
   short sql_create_ext_rtn_grp_auth;
   short sql_quiesce_connect_auth;
   short sql_quiesce_connect_grp_auth;
   short sql_security_admin_auth;
   short sql_security_admin_grp_auth;
   short sql_library_admin_auth;
```

```
   short sql_library_admin_grp_auth;
   short sql_sysmon_auth;
   short sql_sysmon_grp_auth;
};
```

Estructura en COBOL

```
* Archivo: sqlutil.cbl
01 SQL-AUTHORIZATIONS.
    05 SQL-AUTHORIZATIONS-LEN PIC S9(4) COMP-5.
    05 SQL-SYSADM-AUTH        PIC S9(4) COMP-5.
    05 SQL-DBADM-AUTH         PIC S9(4) COMP-5.
    05 SQL-CREATETAB-AUTH     PIC S9(4) COMP-5.
    05 SQL-BINDADD-AUTH       PIC S9(4) COMP-5.
    05 SQL-CONNECT-AUTH       PIC S9(4) COMP-5.
    05 SQL-SYSADM-GRP-AUTH    PIC S9(4) COMP-5.
    05 SQL-DBADM-GRP-AUTH     PIC S9(4) COMP-5.
    05 SQL-CREATETAB-GRP-AUTH PIC S9(4) COMP-5.
    05 SQL-BINDADD-GRP-AUTH   PIC S9(4) COMP-5.
    05 SQL-CONNECT-GRP-AUTH   PIC S9(4) COMP-5.
    05 SQL-SYSCTRL-AUTH       PIC S9(4) COMP-5.
    05 SQL-SYSCTRL-GRP-AUTH   PIC S9(4) COMP-5.
    05 SQL-SYSMAINT-AUTH      PIC S9(4) COMP-5.
    05 SQL-SYSMAINT-GRP-AUTH  PIC S9(4) COMP-5.
    05 SQL-CREATE-NOT-FENC-AUTH PIC S9(4) COMP-5.
    05 SQL-CREATE-NOT-FENC-GRP-AUTH PIC S9(4) COMP-5.
    05 SQL-IMPLICIT-SCHEMA-AUTH PIC S9(4) COMP-5.
    05 SQL-IMPLICIT-SCHEMA-GRP-AUTH PIC S9(4) COMP-5.
    05 SQL-LOAD-AUTH PIC S9(4) COMP-5.
    05 SQL-LOAD-GRP-AUTH PIC S9(4) COMP-5.
*
```

sql_dir_entry

Esta estructura es utilizada por las API del directorio de DCS.

Tabla 31. Campos de la estructura SQL-DIR-ENTRY

Nombre de campo	Tipo de datos	Descripción
STRUCT_ID RELEASE CODEPAGE COMMENT LDB TDB AR PARM	SMALLINT SMALLINT SMALLINT CHAR(30) CHAR(8) CHAR(18) CHAR(32) CHAR(512)	Identificador de la estructura. Establecido en SQL_DCS_STR_ID (definido en sqlenv). Versión de release (asignado por la API). Página de códigos del comentario. Descripción opcional de la base de datos. Nombre local de la base de datos; debe coincidir con el alias de base de datos contenido en el directorio de bases de datos del sistema. Nombre real de la base de datos. Nombre del cliente de aplicaciones. Contiene el prefijo del programa de transacciones, el nombre del programa de transacciones, el nombre del archivo de correlación de SQLCODE y la opción de desconexión y seguridad.

Nota: Los campos de caracteres pasados en esta estructura deben tener una terminación nula o estar rellenados con blancos hasta completar la longitud del campo.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sql_dir_entry
{
        unsigned short         struct_id;
        unsigned short         release;
        unsigned short         codepage;
        _SQLOLDCHAR comment[SQL_CMT_SZ + 1];
        _SQLOLDCHAR ldb[SQL_DBNAME_SZ + 1];
        _SQLOLDCHAR tdb[SQL_LONG_NAME_SZ + 1];
        _SQLOLDCHAR ar[SQL_AR_SZ + 1];
        _SQLOLDCHAR parm[SQL_PARAMETER_SZ + 1];
};
```

Estructura en COBOL

```
* Archivo: sqlenv.cbl
01 SQL-DIR-ENTRY.
    05 STRUCT-ID             PIC 9(4) COMP-5.
    05 RELEASE-LVL           PIC 9(4) COMP-5.
    05 CODEPAGE              PIC 9(4) COMP-5.
    05 COMMENT               PIC X(30).
    05 FILLER                PIC X.
    05 LDB                   PIC X(8).
    05 FILLER                PIC X.
    05 TDB                   PIC X(18).
    05 FILLER                PIC X.
    05 AR                    PIC X(32).
    05 FILLER                PIC X.
    05 PARM                  PIC X(512).
    05 FILLER                PIC X.
    05 FILLER                PIC X(1).
*
```

SQLB_TBS_STATS

Esta estructura se utiliza para devolver estadísticas adicionales sobre espacios de tablas a un programa de aplicación.

Tabla 32. Campos de la estructura SQLB-TBS-STATS

Nombre de campo	Tipo de datos	Descripción
TOTALPAGES	INTEGER	Espacio total del sistema operativo ocupado por el espacio de tablas (en páginas de 4 KB). Para DMS, es la suma de los tamaños de contenedor (incluida la actividad general). Para SMS, es la suma de todo el espacio de archivo utilizado para las tablas almacenadas en este espacio de tablas. Es el único componente de información devuelto para espacios de tablas SMS; los demás campos se establecen en este valor o en cero.
USEABLEPAGES	INTEGER	Para DMS, es igual a TOTALPAGES menos (actividad general más extensiones parciales). Para SMS, es igual a TOTALPAGES.
USEDPAGES	INTEGER	Para DMS, es el número total de páginas que se están utilizando. Para SMS, es igual a TOTALPAGES.

Tabla 32. Campos de la estructura SQLB-TBS-STATS (continuación)

Nombre de campo	Tipo de datos	Descripción
FREEPAGES	INTEGER	Para DMS, es igual a USEABLEPAGES menos USEDPAGES. Para SMS, no es aplicable.
HIGHWATERMARK	INTEGER	Para DMS, la marca de límite superior es el ″final″ actual del espacio de direcciones de espacio de tablas. En otras palabras, el número de página de la primera extensión libre situada a continuación de la última extensión asignada de un espacio de tablas.

Nota: Esto no representa una marca de límite superior, sino una marca de límite actual, pues el valor puede disminuir. Para SMS, no es aplicable.

Durante un reequilibrado del espacio de tablas, el número de páginas utilizables incluirá páginas para el contenedor recién añadido, pero estas nuevas páginas no se reflejarán en el número de páginas libres hasta que finalice el reequilibrado. Cuando no hay un reequilibrado de espacio de tablas en proceso, el número de páginas utilizadas más el número de páginas libres es igual al número de páginas utilizables.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE SQLB_TBS_STATS
{
   sqluint32 totalPages;
   sqluint32 useablePages;
   sqluint32 usedPages;
   sqluint32 freePages;
   sqluint32 highWaterMark;
};
```

Estructura en COBOL

```
* Archivo: sqlutil.cbl
01 SQLB-TBS-STATS.
    05 SQL-TOTAL-PAGES        PIC 9(9) COMP-5.
    05 SQL-USEABLE-PAGES      PIC 9(9) COMP-5.
    05 SQL-USED-PAGES         PIC 9(9) COMP-5.
    05 SQL-FREE-PAGES         PIC 9(9) COMP-5.
    05 SQL-HIGH-WATER-MARK    PIC 9(9) COMP-5.
*
```

SQLB_TBSCONTQRY_DATA

Esta estructura se utiliza para devolver datos sobre contenedores a un programa de aplicación.

Tabla 33. Campos de la estructura SQLB-TBSCONTQRY-DATA

Nombre de campo	Tipo de datos	Descripción
ID	INTEGER	Identificador del contenedor.
NTBS	INTEGER	Siempre 1.
TBSID	INTEGER	Identificador de espacio de tablas.
NAMELEN	INTEGER	Longitud del nombre de contenedor (para lenguajes que no sean C).
NAME	CHAR(256)	Nombre del contenedor.

Tabla 33. Campos de la estructura SQLB-TBSCONTQRY-DATA (continuación)

Nombre de campo	Tipo de datos	Descripción
UNDERDBDIR	INTEGER	Su valor es 1 (contenedor situado en el directorio de bases de datos) o 0 (contenedor no situado en el directorio de bases de datos)
CONTTYPE	INTEGER	Tipo del contenedor.
TOTALPAGES	INTEGER	Número total de páginas ocupadas por el contenedor de espacio de tablas.
USEABLEPAGES	INTEGER	Para DMS, TOTALPAGES menos actividad general. Para SMS, es igual a TOTALPAGES.
OK	INTEGER	Su valor es 1 (contenedor accesible) o 0 (contenedor no accesible). Cero indica una situación anómala que generalmente requiere la atención del administrador de la base de datos.

Los valores posibles para CONTTYPE (definido en sqlutil) son:

SQLB_CONT_PATH
Especifica una vía de acceso de directorios (solamente para SMS).

SQLB_CONT_DISK
Especifica un dispositivo en bruto (solamente para DMS).

SQLB_CONT_FILE
Especifica un archivo (solamente para DMS).

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE SQLB_TBSCONTQRY_DATA
{
   sqluint32 id;
   sqluint32 nTbs;
   sqluint32 tbsID;
   sqluint32 nameLen;
   char name[SQLB_MAX_CONTAIN_NAME_SZ];
   sqluint32 underDBDir;
   sqluint32 contType;
   sqluint32 totalPages;
   sqluint32 useablePages;
   sqluint32 ok;
};
```

Estructura en COBOL

```
* Archivo: sqlutbcq.cbl
01 SQLB-TBSCONTQRY-DATA.
    05 SQL-ID                 PIC 9(9) COMP-5.
    05 SQL-N-TBS              PIC 9(9) COMP-5.
    05 SQL-TBS-ID             PIC 9(9) COMP-5.
    05 SQL-NAME-LEN           PIC 9(9) COMP-5.
    05 SQL-NAME               PIC X(256).
    05 SQL-UNDER-DBDIR        PIC 9(9) COMP-5.
    05 SQL-CONT-TYPE          PIC 9(9) COMP-5.
    05 SQL-TOTAL-PAGES        PIC 9(9) COMP-5.
    05 SQL-USEABLE-PAGES      PIC 9(9) COMP-5.
    05 SQL-OK                 PIC 9(9) COMP-5.
*
```

SQLB_TBSPQRY_DATA

Esta estructura se utiliza para devolver datos sobre espacios de tablas a un programa de aplicación.

Tabla 34. Campos de la estructura SQLB-TBSPQRY-DATA

Nombre de campo	Tipo de datos	Descripción
TBSPQVER	CHAR(8)	Identificador de versión de la estructura.
ID	INTEGER	Identificador interno del espacio de tablas.
NAMELEN	INTEGER	Longitud del nombre del espacio de tablas.
NAME	CHAR(128)	Nombre terminado en nulo del espacio de tablas.
TOTALPAGES	INTEGER	Número de páginas especificado por CREATE TABLESPACE (sólo DMS).
USEABLEPAGES	INTEGER	TOTALPAGES menos actividad general (sólo DMS). Este valor se redondea por defecto en el próximo múltiplo de 4 KB.
FLAGS	INTEGER	Atributos de bit del espacio de tablas.
PAGESIZE	INTEGER	Tamaño de página (en bytes) del espacio de tablas. Fijado actualmente en 4 KB.
EXTSIZE	INTEGER	Tamaño ampliado (en páginas) del espacio de tablas.
PREFETCHSIZE	INTEGER	Tamaño de captación previa.
NCONTAINERS	INTEGER	Número de contenedores del espacio de tablas.
TBSSTATE	INTEGER	Estados del espacio de tablas.
LIFELSN	CHAR(6)	Indicación de fecha y hora que identifica el origen del espacio de tablas.
FLAGS2	INTEGER	Atributos de bit del espacio de tablas.
MINIMUMRECTIME	CHAR(27)	Punto del tiempo más temprano que una operación de avance de espacio de tablas puntual en el tiempo puede especificar.

Tabla 34. Campos de la estructura SQLB-TBSPQRY-DATA (continuación)

Nombre de campo	Tipo de datos	Descripción
STATECHNGOBJ	INTEGER	Si TBSSTATE es SQLB_LOAD_PENDING o SQLB_DELETE_PENDING, el ID de objeto del espacio de tablas STATECHANGEID que ha provocado el establecimiento del estado del espacio de tablas. De lo contrario, es cero.
STATECHNGID	INTEGER	Si TBSSTATE es SQLB_LOAD_PENDING o SQLB_DELETE_PENDING, el ID de objeto STATECHANGEOBJ que ha provocado el establecimiento del estado del espacio de tablas. De lo contrario, es cero.
NQUIESCERS	INTEGER	Si TBSSTATE es SQLB_QUIESCED_SHARE, UPDATE o EXCLUSIVE, el número de inmovilizadores del espacio de tablas y el número de entradas de QUIESCERS.
QUIESCER	Matriz de las estructuras SQLB_QUIESCER_ DATA	Cada entrada de la matriz consta de los datos de inmovilización de un objeto inmovilizado.
FSCACHING	UNSIGNED CHAR	Política de almacenamiento en antememoria del sistema de archivos para dar soporte a la E/S directa. Es un campo de 31 bits.
RESERVED	CHAR(31)	Reservado para una utilización futura.

Los valores posibles para FLAGS (definido en sqlutil) son:

SQLB_TBS_SMS
Espacio gestionado por el sistema

SQLB_TBS_DMS
Espacio gestionado por la base de datos

SQLB_TBS_ANY
Todos los tipos de datos permanentes. Espacio de tablas regular.

SQLB_TBS_LONG
Todos los tipos de datos permanentes. Espacio de tablas grande.

SQLB_TBS_SYSTMP
Datos temporales del sistema.

SQLB_TBS_USRTMP
Datos temporales del usuario.

Los valores posibles para TBSSTATE (definido en sqlutil) son:

SQLB_NORMAL
Normal

SQLB_QUIESCED_SHARE
Inmovilizado: SHARE

SQLB_QUIESCED_UPDATE
Inmovilizado: UPDATE

SQLB_QUIESCED_EXCLUSIVE
Inmovilizado: EXCLUSIVE

SQLB_LOAD_PENDING
Carga pendiente

SQLB_DELETE_PENDING
Supresión pendiente

SQLB_BACKUP_PENDING
Copia de seguridad pendiente

SQLB_ROLLFORWARD_IN_PROGRESS
Avance en proceso

SQLB_ROLLFORWARD_PENDING
Avance pendiente

SQLB_RESTORE_PENDING
Restauración pendiente

SQLB_DISABLE_PENDING
Inhabilitación pendiente

SQLB_REORG_IN_PROGRESS
Reorganización en curso

SQLB_BACKUP_IN_PROGRESS
Copia de seguridad en curso

SQLB_STORDEF_PENDING
Se debe definir el almacenamiento

SQLB_RESTORE_IN_PROGRESS
Restauración en curso

SQLB_STORDEF_ALLOWED
Se puede definir el almacenamiento

SQLB_STORDEF_FINAL_VERSION
El estado de la definición del almacenamiento es 'final'

SQLB_STORDEF_CHANGED
Se cambió la definición del almacenamiento antes del avance

SQLB_REBAL_IN_PROGRESS
El reequilibrador DMS está activo

SQLB_PSTAT_DELETION
La supresión del espacio de tablas está en curso

SQLB_PSTAT_CREATION
La creación del espacio de tablas está en curso.

Los valores posibles para FLAGS2 (definido en sqlutil) son:

SQLB_STATE_SET
Solamente para su utilización por el servicio técnico.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE SQLB_TBSPQRY_DATA
{
   char tbspqver[SQLB_SVERSION_SIZE];
   sqluint32 id;
   sqluint32 nameLen;
   char name[SQLB_MAX_TBS_NAME_SZ];
   sqluint32 totalPages;
   sqluint32 useablePages;
   sqluint32 flags;
   sqluint32 pageSize;
   sqluint32 extSize;
   sqluint32 prefetchSize;
   sqluint32 nContainers;
   sqluint32 tbsState;
   char lifeLSN[6];
   char pad[2];
   sqluint32 flags2;
   char minimumRecTime[SQL_STAMP_STRLEN+1];
   char pad1[1];
   sqluint32 StateChngObj;
   sqluint32 StateChngID;
   sqluint32 nQuiescers;
   struct SQLB_QUIESCER_DATA quiescer[SQLB_MAX_QUIESCERS];
   unsigned char   fsCaching;
   char reserved[31];
};

SQL_STRUCTURE SQLB_QUIESCER_DATA
{
   sqluint32 quiesceId;
   sqluint32 quiesceObject;
};
```

Parámetros de la estructura de datos SQLB_QUIESCER_DATA

pad Reservado. Se utiliza para la alineación de estructuras y no debe llenarse con datos del usuario.

pad1 Reservado. Se utiliza para la alineación de estructuras y no debe llenarse con datos del usuario.

quiesceId
Entrada. ID del espacio de tablas donde se ha creado el objeto inmovilizado.

quiesceObject
Entrada. ID de objeto del objeto inmovilizado.

Estructura en COBOL

```
* Archivo: sqlutbsp.cbl
01 SQLB-TBSPQRY-DATA.
    05 SQL-TBSPQVER          PIC X(8).
    05 SQL-ID                PIC 9(9) COMP-5.
    05 SQL-NAME-LEN          PIC 9(9) COMP-5.
    05 SQL-NAME              PIC X(128).
    05 SQL-TOTAL-PAGES       PIC 9(9) COMP-5.
    05 SQL-USEABLE-PAGES     PIC 9(9) COMP-5.
    05 SQL-FLAGS             PIC 9(9) COMP-5.
    05 SQL-PAGE-SIZE         PIC 9(9) COMP-5.
    05 SQL-EXT-SIZE          PIC 9(9) COMP-5.
    05 SQL-PREFETCH-SIZE     PIC 9(9) COMP-5.
```

```
    05 SQL-N-CONTAINERS       PIC 9(9) COMP-5.
    05 SQL-TBS-STATE          PIC 9(9) COMP-5.
    05 SQL-LIFE-LSN           PIC X(6).
    05 SQL-PAD                PIC X(2).
    05 SQL-FLAGS2             PIC 9(9) COMP-5.
    05 SQL-MINIMUM-REC-TIME   PIC X(26).
    05 FILLER                 PIC X.
    05 SQL-PAD1               PIC X(1).
    05 SQL-STATE-CHNG-OBJ     PIC 9(9) COMP-5.
    05 SQL-STATE-CHNG-ID      PIC 9(9) COMP-5.
    05 SQL-N-QUIESCERS        PIC 9(9) COMP-5.
    05 SQL-QUIESCER OCCURS 5 TIMES.
        10 SQL-QUIESCE-ID     PIC 9(9) COMP-5.
        10 SQL-QUIESCE-OBJECT PIC 9(9) COMP-5.
    05 SQL-FSCACHING          PIC X(1).
    05 SQL-RESERVED           PIC X(31).
*
```

SQLCA

La estructura del área de comunicaciones de SQL (SQLCA) es utilizada por el gestor de bases de datos para devolver información sobre errores a un programa de aplicación. Esta estructura se actualiza después de cada llamada a una API y sentencia de SQL emitidas.

Sintaxis del lenguaje

Estructura en C

```
/* Archivo: sqlca.h */
/* Estructura: SQLCA */
/* ... */
SQL_STRUCTURE  sqlca
{
  _SQLOLDCHAR    sqlcaid[8];
  sqlint32       sqlcabc;
  #ifdef DB2_SQL92E
  sqlint32       sqlcade;
  #else
  sqlint32       sqlcode;
  #endif
  short          sqlerrml;
  _SQLOLDCHAR    sqlerrmc[70];
  _SQLOLDCHAR    sqlerrp[8];
  sqlint32       sqlerrd[6];
  _SQLOLDCHAR    sqlwarn[11];
  #ifdef DB2_SQL92E
  _SQLOLDCHAR    sqlstat[5];
  #else
  _SQLOLDCHAR    sqlstate[5];
  #endif
};
/* ... */
```

Estructura en COBOL

```
* Archivo: sqlca.cbl
01 SQLCA SYNC.
    05 SQLCAID PIC X(8) VALUE "SQLCA   ".
    05 SQLCABC PIC S9(9) COMP-5 VALUE 136.
    05 SQLCODE PIC S9(9) COMP-5.
    05 SQLERRM.
    05 SQLERRP PIC X(8).
    05 SQLERRD OCCURS 6 TIMES PIC S9(9) COMP-5.
    05 SQLWARN.
```

```
        10 SQLWARN0 PIC X.
        10 SQLWARN1 PIC X.
        10 SQLWARN2 PIC X.
        10 SQLWARN3 PIC X.
        10 SQLWARN4 PIC X.
        10 SQLWARN5 PIC X.
        10 SQLWARN6 PIC X.
        10 SQLWARN7 PIC X.
        10 SQLWARN8 PIC X.
        10 SQLWARN9 PIC X.
        10 SQLWARNA PIC X.
    05 SQLSTATE PIC X(5).
*
```

sqlchar

Esta estructura se utiliza para pasar datos de longitud variable al gestor de bases de datos.

Tabla 35. Campos de la estructura SQLCHAR

Nombre de campo	Tipo de datos	Descripción
LENGTH	SMALLINT	Longitud de la serie a la que señala DATA.
DATA	CHAR(n)	Matriz de caracteres de longitud LENGTH.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqlchar
{
        short length;
        _SQLOLDCHAR data[1];
};
```

Estructura en COBOL

Esto no está definido en ningún archivo de cabecera. El ejemplo siguiente muestra cómo definir la estructura en COBOL:

```
* Sustituir maxlen por el valor apropiado:
01 SQLCHAR.
49 SQLCHAR-LEN  PIC S9(4) COMP-5.
49 SQLCHAR-DATA PIC X(maxlen).
```

SQLDA

La estructura del área de descriptores de SQL (SQLDA) es un conjunto de variables que es necesario para la ejecución de la sentencia DESCRIBE de SQL. Las variables de SQLDA son opciones que se pueden utilizar con las sentencias PREPARE, OPEN, FETCH, EXECUTE y CALL.

La SQLDA se comunica mediante SQL dinámico; se puede utilizar en una sentencia DESCRIBE, modificar con las direcciones de variables de lenguaje principal y luego reutilizar en una sentencia FETCH.

Las SQLDA se pueden utilizar con todos los lenguajes, pero solamente se proporcionan declaraciones predefinidas para C, REXX, FORTRAN y COBOL.

El significado de la información contenida en una SQLDA depende de la utilización de la SQLDA. En las sentencias PREPARE y DESCRIBE, una SQLDA proporciona información a un programa de aplicación sobre una sentencia preparada. En las sentencias OPEN, EXECUTE, FETCH y CALL, una SQLDA describe variables de lenguaje principal.

Sintaxis del lenguaje

Estructura en C

```
/* Archivo: sqlda.h */
/* Estructura: SQLDA */
/* ... */
SQL_STRUCTURE  sqlda
{
  _SQLOLDCHAR    sqldaid[8];
  long           sqldabc;
  short          sqln;
  short          sqld;
  struct sqlvar  sqlvar[1];
};
/* ... */
```

```
/* Archivo: sqlda.h */
/* Estructura: SQLVAR */
/* ... */
SQL_STRUCTURE  sqlvar
{
  short          sqltype;
  short          sqllen;
  _SQLOLDCHAR   *SQL_POINTER sqldata;
  short         *SQL_POINTER sqlind;
  struct sqlname sqlname;
};
/* ... */
```

```
/* Archivo: sqlda.h */
/* Estructura: SQLNAME */
/* ... */
SQL_STRUCTURE  sqlname
{
  short length;
  _SQLOLDCHAR    data[30];
};
/* ... */
```

```
/* Archivo: sqlda.h */
/* Estructura: SQLVAR2 */
/* ... */
SQL_STRUCTURE  sqlvar2
{
  union sql8bytelen len;
  char *SQL_POINTER sqldatalen;
  struct sqldistinct_type sqldatatype_name;
};
/* ... */
```

```
/* Archivo: sqlda.h */
/* Estructura: SQL8BYTELEN */
/* ... */
union sql8bytelen
{
  long        reserve1[2];
  long        sqllonglen;
};
/* ... */
```

```
/* Archivo: sqlda.h */
/* Estructura: SQLDISTINCT-TYPE */
/* ... */
SQL_STRUCTURE  sqldistinct_type
{
  short length;
  char           data[27];
  char           reserved1[3];
};
/* ... */
```

Estructura en COBOL

```
* Archivo: sqlda.cbl
01 SQLDA SYNC.
    05 SQLDAID PIC X(8) VALUE "SQLDA  ".
    05 SQLDABC PIC S9(9) COMP-5.
    05 SQLN PIC S9(4) COMP-5.
    05 SQLD PIC S9(4) COMP-5.
    05 SQLVAR-ENTRIES OCCURS 0 TO 1489 TIMES
        10 SQLVAR.
        10 SQLVAR2 REDEFINES SQLVAR.
*
```

sqldcol

Esta estructura se utiliza para pasar información variable sobre columnas a las API db2Export, db2Import y db2Load.

Tabla 36. Campos de la estructura SQLDCOL

Nombre de campo	Tipo de datos	Descripción
DCOLMETH	SMALLINT	Carácter que indica el método que se debe usar para seleccionar columnas dentro del archivo de datos.
DCOLNUM	SMALLINT	Número de columnas especificadas en la matriz DCOLNAME.
DCOLNAME	Matriz	Matriz de estructuras sqldcoln DCOLNUM.

Los valores válidos para DCOLMETH (definido en sqlutil) son:

SQL_METH_N
: Nombres. Al importar o cargar, utilice los nombres de columna proporcionadas por esta estructura para identificar los datos que se deben importar o cargar desde el archivo externo. Las mayúsculas y minúsculas de estos nombres de columna deben coincidir con las mayúsculas y minúsculas de los nombres correspondientes en los catálogos del sistema. Al exportar, utilice los nombres de columna proporcionadas por esta estructura como nombres de columna del archivo de salida.

  El puntero dcolnptr de cada elemento de la matriz dcolname apunta a una matriz de caracteres, cuya longitud en bytes es la indicada por dcolnlen, que forman el nombre de una columna que se debe importar o cargar. El campo dcolnum, que debe ser un valor positivo, indica el número de elementos de la matriz dcolname.

Este método no es válido si el archivo externo no contiene nombres de columna (por ejemplo, archivo de formato DEL o ASC).

SQL_METH_P
Posiciones. Al importar o cargar, utilice las posiciones de columna proporcionadas por esta estructura para identificar los datos que se deben importar o cargar desde el archivo externo. Este método no es válido cuando se exportan datos.

El puntero dcolnptr de cada elemento de la matriz dcolname no se tiene en cuenta, mientras que el campo dcolnlen contiene una posición de columna en el archivo externo. El campo dcolnum, que debe ser un valor positivo, indica el número de elementos de la matriz dcolname.

El valor de posición de columna válido más bajo es 1 (que indica la primera columna), y el valor válido más alto depende del tipo de archivo externo. La selección de posición no es válida para importar archivos ASC.

SQL_METH_L
Ubicaciones. Al importar o cargar, utilice las posiciones de columna inicial y final proporcionadas por esta estructura para identificar los datos que se deben importar o cargar desde el archivo externo. Este método no es válido cuando se exportan datos.

El campo dcolnptr del primer elemento de la matriz dcolname apunta a una estructura sqlloctab, que consta de una matriz de estructuras sqllocpair. El número de elementos de esta matriz está determinado por el campo dcolnum de la estructura sqldcol, que debe ser un valor positivo. Cada elemento de la matriz es un par de enteros, de 2 bytes, que indican donde comienza y finaliza la columna. El primer elemento de cada par de valores de ubicación es el byte dentro del archivo donde comienza la columna, y el segundo elemento es el byte donde finaliza la columna. La primera posición de byte dentro de una fila del archivo se considera la posición de byte 1. Las columnas pueden solaparse.

SQL_METH_D
Valor por omisión. Al importar o cargar archivos DEL e IXF, la primera columna del archivo se carga o importa en la primera columna de la tabla, y así sucesivamente. Al exportar datos, se utilizan los nombres por omisión para las columnas del archivo externo.

Los campos dcolnum y dcolname de la estructura sqldcol no se tienen en cuenta, y las columnas del archivo externo se toman en su orden natural.

Una columna del archivo externo se puede utilizar en la matriz más de una vez. No es necesario utilizar cada columna del archivo externo.

Tabla 37. Campos de la estructura SQLDCOLN

Nombre de campo	Tipo de datos	Descripción
DCOLNLEN	SMALLINT	Longitud de los datos a los que señala DCOLNPTR.
DCOLNPTR	Puntero	Puntero a un elemento de datos determinado por DCOLMETH.

Nota: Los campos DCOLNLEN y DCOLNPTR se repiten para cada columna especificada.

Tabla 38. Campos de la estructura SQLLOCTAB

Nombre de campo	Tipo de datos	Descripción
LOCPAIR	Matriz	Matriz de estructuras sqllocpair.

Tabla 39. Campos de la estructura SQLLOCPAIR

Nombre de campo	Tipo de datos	Descripción
BEGIN_LOC	SMALLINT	Posición inicial de los datos de columna en el archivo externo.
END_LOC	SMALLINT	Posición final de los datos de columna en el archivo externo.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqldcol
{
   short dcolmeth;
   short dcolnum;
   struct sqldcoln dcolname[1];
};

SQL_STRUCTURE sqldcoln
{
   short dcolnlen;
   char *dcolnptr;
};

SQL_STRUCTURE sqlloctab
{
   struct sqllocpair locpair[1];
};

SQL_STRUCTURE sqllocpair
{
   short begin_loc;
   short end_loc;
};
```

Estructura en COBOL

```
* Archivo: sqlutil.cbl
01 SQL-DCOLDATA.
    05 SQL-DCOLMETH           PIC S9(4) COMP-5.
    05 SQL-DCOLNUM            PIC S9(4) COMP-5.
    05 SQLDCOLN OCCURS 0 TO 255 TIMES DEPENDING ON SQL-DCOLNUM.
        10 SQL-DCOLNLEN       PIC S9(4) COMP-5.
        10 FILLER             PIC X(2).
        10 SQL-DCOLN-PTR      USAGE IS POINTER.
*


* Archivo: sqlutil.cbl
01 SQL-LOCTAB.
    05 SQL-LOC-PAIR OCCURS 1 TIMES.
        10 SQL-BEGIN-LOC      PIC S9(4) COMP-5.
        10 SQL-END-LOC        PIC S9(4) COMP-5.
*
```

sqle_addn_options

Esta estructura se utiliza para pasar información a la API sqleaddn.

Tabla 40. Campos de la estructura SQLE-ADDN-OPTIONS

Nombre de campo	Tipo de datos	Descripción
SQLADDID	CHAR	Valor de reclamo que se debe establecer en SQLE_ADDOPTID_V51.
TBLSPACE_TYPE	sqluint32	Especifica el tipo de definiciones de espacios de tablas temporales del sistema que se debe utilizar para el nodo que se está añadiendo. Consulte más abajo para conocer los valores. Nota: esta opción no se tiene en cuenta para los espacios de tablas temporales del sistema que se definen para utilizar almacenamiento automático (es decir, espacios de tablas temporales del sistema que se han creado con la cláusula MANAGED BY AUTOMATIC STORAGE de la sentencia CREATE TABLESPACE o sin especificar la cláusula MANAGED BY CLAUSE). Para estos espacios de tabla, no se puede aplazar la creación de contenedores ni optar por crear un conjunto de contenedores tal como están definidos en otra partición. El gestor de bases de datos asignará automáticamente contenedores basándose en las vías de acceso de almacenamiento que están asociadas a la base de datos.
TBLSPACE_NODE	SQL_PDB_NODE_TYPE	Especifica el número de nodo del que deben obtenerse definiciones de espacios de tablas temporales del sistema. El número de nodo debe existir en el archivo db2nodes.cfg y solamente se utiliza si el campo tblspace_type tiene el valor SQLE_TABLESPACES_LIKE_NODE.

Los valores válidos para TBLSPACE_TYPE (definido en sqlenv) son:

SQLE_TABLESPACES_NONE
No crear ningún espacio de tablas temporales del sistema.

SQLE_TABLESPACES_LIKE_NODE
Los contenedores para los espacios de tablas temporales del sistema deben ser los mismos que los utilizados para el nodo especificado.

SQLE_TABLESPACES_LIKE_CATALOG
Los contenedores para los espacios de tablas temporales del sistema deben ser los mismos que los utilizados para el nodo de catálogo de cada base de datos.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqle_addn_options
{
        char sqladdid[8];
        sqluint32 tblspace_type;
        SQL_PDB_NODE_TYPE tblspace_node;
};
```

Estructura en COBOL

```
* Archivo: sqlenv.cbl
01 SQLE-ADDN-OPTIONS.
    05 SQLADDID               PIC X(8).
    05 SQL-TBLSPACE-TYPE      PIC 9(9) COMP-5.
    05 SQL-TBLSPACE-NODE      PIC S9(4) COMP-5.
    05 FILLER                 PIC X(2).
*
```

sqle_client_info

Esta estructura se utiliza para pasar información a las API sqleseti y sqleqryi. Esta estructura especifica:

- El tipo de información que se está definiendo o consultando
- La longitud de los datos que se están definiendo o consultando
- Un puntero a uno de estos elementos:
  - Un área que contendrá los datos que se están definiendo
  - Un área de longitud suficiente para contener los datos que se están consultando

Las aplicaciones pueden especificar los tipos de información siguientes:

- ID de usuario del cliente que se está definiendo o consultando. Se puede definir un máximo de 255 caracteres, pero los servidores pueden truncar esta información de acuerdo con un valor específico de la plataforma.

  Nota: Este ID de usuario es solamente para fines de identificación, y no se utiliza para ninguna autorización.
- Nombre de la estación de trabajo del cliente que se está definiendo o consultando. Se puede definir un máximo de 255 caracteres, pero los servidores pueden truncar esta información de acuerdo con un valor específico de la plataforma.
- Nombre de aplicación del cliente que se está definiendo o consultando. Se puede definir un máximo de 255 caracteres, pero los servidores pueden truncar esta información de acuerdo con un valor específico de la plataforma.
- Vía de acceso del paquete actual del cliente que se está definiendo o consultando. Se puede definir un máximo de 255 caracteres, pero los servidores pueden truncar esta información de acuerdo con un valor específico de la plataforma.
- ID del programa cliente que se está definiendo o consultando. Se puede definir un máximo de 80 caracteres, pero los servidores pueden truncar esta información de acuerdo con un valor específico de la plataforma.
- Serie de contabilidad de cliente que se está definiendo o consultando. Se puede definir un máximo de 200 caracteres, pero los servidores pueden truncar esta información de acuerdo con un valor específico de la plataforma.

Nota: La información se puede definir mediante la API sqlesact. Pero sqlesact no permite cambiar la información contable una vez establecida una conexión, mientras que sqleseti permite cambiar la información contable para las conexiones actuales y futuras.

Tabla 41. Campos de la estructura SQLE-CLIENT-INFO

Nombre de campo	Tipo de datos	Descripción
TYPE	sqlint32	Tipo de valor.
LENGTH	sqlint32	Longitud del valor. En las llamadas a sqleseti, la longitud puede estar comprendida entre 0 y la longitud máxima definida para el tipo. Una longitud 0 indica un valor nulo. En las llamadas a sqleqryi, se devuelve la longitud, pero el área apuntada por pValue debe ser lo suficiente grande para contener la longitud máxima para el tipo. Una longitud 0 indica un valor nulo.
PVALUE	Puntero	Puntero a un almacenamiento intermedio, asignado por la aplicación, que contiene el valor especificado. El tipo de datos de este valor depende del campo TYPE.

A continuación se muestran las entradas válidas para el elemento SQLE-CLIENT-INFO TYPE y las descripciones correspondientes a cada entrada:

Tabla 42. Valores de conexión

Tipo	Tipo de datos	Descripción
SQLE_CLIENT_INFO_USERID	CHAR(255)	ID de usuario del cliente. Algunos servidores pueden truncar este valor. Por ejemplo, los servidores DB2 para z/OS permiten una longitud máxima de 16 caracteres. Este ID de usuario es solamente para fines de identificación, y no se utiliza para ninguna autorización.
SQLE_CLIENT_INFO_WRKSTNNAME	CHAR(255)	Nombre de la estación de trabajo del cliente. Algunos servidores pueden truncar este valor. Por ejemplo, los servidores DB2 para z/OS permiten una longitud máxima de 18 caracteres.
SQLE_CLIENT_INFO_APPLNAME	CHAR(255)	Nombre de aplicación del cliente. Algunos servidores pueden truncar este valor. Por ejemplo, los servidores DB2 para z/OS permiten una longitud máxima de 32 caracteres.
SQLE_CLIENT_INFO_PROGRAMID	CHAR(80)	Identificador de programa del cliente. Una vez definido este elemento, DB2 Universal Database para z/OS Versión 8 asocia este identificador a cualquier sentencia insertada en la antememoria de sentencias de SQL dinámico. Este elemento sólo se puede utilizar para aplicaciones que acceden a DB2 UDB para z/OS Versión 8.
SQLE_CLIENT_INFO_ACCTSTR	CHAR(200)	Serie de contabilidad del cliente. Algunos servidores pueden truncar este valor. Por ejemplo, los servidores DB2 para z/OS permiten una longitud máxima de 200 caracteres.
SQLE_CLIENT_INFO_AUTOCOMMIT	CHAR(1)	Valor de confirmación automática del cliente. Puede ser igual a SQLE_CLIENT_AUTOCOMMIT_ON o SQLE_CLIENT_AUTOCOMMIT_OFF.

Nota: Estos nombres de campo están definidos para el lenguaje de programación C. Existen nombres similares para FORTRAN y COBOL, que tienen la misma semántica.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqle_client_info
{
        unsigned short         type;
        unsigned short         length;
        char *pValue;
};
```

Estructura en COBOL

```
* Archivo: sqlenv.cbl
01 SQLE-CLIENT-INFO.
    05 SQLE-CLIENT-INFO-ITEM OCCURS 4 TIMES.
        10 SQLE-CLIENT-INFO-TYPE   PIC S9(4) COMP-5.
        10 SQLE-CLIENT-INFO-LENGTH PIC S9(4) COMP-5.
        10 SQLE-CLIENT-INFO-VALUE  USAGE IS POINTER.
*
```

sqle_conn_setting

Esta estructura se utiliza para especificar tipos de valores de conexión y valores para las API sqleqryc y sqlesetc.

Tabla 43. Campos de la estructura SQLE-CONN-SETTING

Nombre de campo	Tipo de datos	Descripción
TYPE VALUE	SMALLINT SMALLINT	Tipo de valor. Valor del valor.

Las entradas válidas para el elemento TYPE de SQLE-CONN-SETTING y las descripciones correspondientes a cada entrada aparecen listadas a continuación (se definen en sqlenv y sql):

Tabla 44. Valores de conexión

Tipo	Valor	Descripción
SQL_CONNECT_TYPE	SQL_CONNECT_1 SQL_CONNECT_2	CONNECT de tipo 1 impone la semántica de base de datos única por unidad de trabajo de los releases anteriores, también conocida como normas de unidad de trabajo remota (RUOW). CONNECT de tipo 2 da soporte la semántica de varias bases de datos por unidad de trabajo de DUOW.
SQL_RULES	SQL_RULES_DB2 SQL_RULES_STD	Permite a la sentencia CONNECT de SQL conmutar la conexión actual por una conexión establecida (latente). Sólo permite el establecimiento de una conexión nueva por medio de la sentencia SQL CONNECT. Se deberá utilizar la sentencia SET CONNECTION de SQL para conmutar la conexión actual por una conexión latente.

Tabla 44. Valores de conexión (continuación)

Tipo	Valor	Descripción
SQL_DISCONNECT	SQL_DISCONNECT_EXPL SQL_DISCONNECT_COND SQL_DISCONNECT_AUTO	Elimina las conexiones que la sentencia SQL RELEASE ha marcado explícitamente para liberar durante la confirmación. Interrumpe las conexiones que no tiene cursores WITH HOLD abiertos durante la confirmación, y las conexiones que la sentencia SQL RELEASE ha marcado explícitamente para liberar. Interrumpe todas las conexiones durante la confirmación.
SQL_DEFERRED_PREPARE	SQL_DEFERRED_PREPARE_ NO SQL_DEFERRED_PREPARE_ YES SQL_DEFERRED_PREPARE_ ALL	La sentencia PREPARE se ejecutará en el momento de emitirse. La ejecución de la sentencia PREPARE se diferirá hasta que se emita la sentencia OPEN, DESCRIBE o EXECUTE correspondiente. La sentencia PREPARE no se diferirá si utiliza la cláusula INTO, que requiere que se devuelva un SQLDA inmediatamente. Sin embargo, si se emite la sentencia PREPARE INTO para un cursor que no utiliza ningún marcador de parámetro, el proceso se optimizará al ejecutar previamente OPEN en el cursor cuando se ejecute PREPARE. Igual que YES, excepto que también se difiere una sentencia PREPARE INTO que contiene marcadores de parámetro. Si una sentencia PREPARE INTO no contiene marcadores de parámetro, también se realizará la reapertura del cursor. Si la sentencia PREPARE utiliza la cláusula INTO para devolver un SQLDA, la aplicación no debe hacer referencia al contenido de este SQLDA hasta que se emita y se devuelva la sentencia OPEN, DESCRIBE o EXECUTE.
SQL_CONNECT_NODE	Entre 0 y 999, o la palabra clave SQL_CONN_CATALOG_ NODE.	Especifica el nodo con el que debe establecerse una conexión. Altera temporalmente el valor de la variable de entorno DB2NODE. Por ejemplo, si se han definido los nodos 1, 2 y 3, el cliente sólo necesita poder acceder a uno de estos nodos. Si sólo se ha catalogado el nodo 1 que contiene bases de datos y se establece este parámetro en 3, el próximo intento de conexión producirá una conexión en el nodo 3, después de una conexión inicial al nodo 1.

Tabla 44. Valores de conexión (continuación)

Tipo	Valor	Descripción
SQL_ATTACH_NODE	Entre 0 y 999.	Especifica el nodo con el que debe establecerse una conexión. Altera temporalmente el valor de la variable de entorno DB2NODE. Por ejemplo, si se han definido los nodos 1, 2 y 3, el cliente sólo necesita poder acceder a uno de estos nodos. Si sólo se ha catalogado el nodo 1 que contiene bases de datos y se establece este parámetro en 3, el próximo intento de conexión producirá una conexión en el nodo 3, después de una conexión inicial al nodo 1.

Nota: Estos nombres de campo están definidos para el lenguaje de programación C. Existen nombres similares para FORTRAN y COBOL, que tienen la misma semántica.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqle_conn_setting
{
        unsigned short         type;
        unsigned short         value;
};
```

Estructura en COBOL

```
* Archivo: sqlenv.cbl
01 SQLE-CONN-SETTING.
    05 SQLE-CONN-SETTING-ITEM OCCURS 7 TIMES.
        10 SQLE-CONN-TYPE  PIC S9(4) COMP-5.
        10 SQLE-CONN-VALUE PIC S9(4) COMP-5.
*
```

sqle_node_local

Esta estructura se utiliza para catalogar nodos locales para la API sqlectnd.

Tabla 45. Campos de la estructura SQLE-NODE-LOCAL

Nombre de campo	Tipo de datos	Descripción
INSTANCE_NAME	CHAR(8)	Nombre de una instancia.

Nota: Los campos de caracteres pasados en esta estructura deben tener una terminación nula o estar rellenados con blancos hasta completar la longitud del campo.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqle_node_local
{
        char instance_name[SQL_INSTNAME_SZ+1];
};
```

Estructura en COBOL

```
* Archivo: sqlenv.cbl
01 SQL-NODE-LOCAL.
    05 SQL-INSTANCE-NAME      PIC X(8).
    05 FILLER                 PIC X.
*
```

sqle_node_npipe

Esta estructura se utiliza para catalogar nodos de conexión con nombre para la API sqlectnd.

Tabla 46. Campos de la estructura SQLE-NODE-NPIPE

Nombre de campo	Tipo de datos	Descripción
COMPUTERNAME	CHAR(15)	Nombre del sistema.
INSTANCE_NAME	CHAR(8)	Nombre de una instancia.

Nota: Los campos de caracteres pasados en esta estructura deben tener una terminación nula o estar rellenados con blancos hasta completar la longitud del campo.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqle_node_npipe
{
        char computername[SQL_COMPUTERNAME_SZ+1];
        char instance_name[SQL_INSTNAME_SZ+1];
};
```

Estructura en COBOL

```
* Archivo: sqlenv.cbl
01 SQL-NODE-NPIPE.
    05 COMPUTERNAME           PIC X(15).
    05 FILLER                 PIC X.
    05 INSTANCE-NAME          PIC X(8).
    05 FILLER                 PIC X.
*
```

sqle_node_struct

Esta estructura se utiliza para catalogar nodos para la API sqlectnd.

Nota: NetBIOS ya no está soportado. SNA, incluyendo sus API APPC, APPN y CPI-C, tampoco está soportado. Si utiliza estos protocolos, debe volver a catalogar los nodos y bases de datos utilizando un protocolo soportado como, por ejemplo, TCP/IP. Las referencias a estos protocolos se deben pasar por alto.

Tabla 47. Campos de la estructura SQLE-NODE-STRUCT

Nombre de campo	Tipo de datos	Descripción
STRUCT_ID	SMALLINT	Identificador de la estructura.
CODEPAGE	SMALLINT	Página de códigos del comentario.
COMMENT	CHAR(30)	Descripción opcional del nodo.

Tabla 47. Campos de la estructura SQLE-NODE-STRUCT (continuación)

Nombre de campo	Tipo de datos	Descripción
NODENAME	CHAR(8)	Nombre local del nodo donde reside la base de datos.
PROTOCOL	CHAR(1)	Tipo de protocolo de comunicaciones.

Nota: Los campos de caracteres pasados en esta estructura deben tener una terminación nula o estar rellenados con blancos hasta completar la longitud del campo.

Los valores válidos para PROTOCOL (definido en sqlenv) son:

- SQL_PROTOCOL_APPC
- SQL_PROTOCOL_APPN
- SQL_PROTOCOL_CPIC
- SQL_PROTOCOL_LOCAL
- SQL_PROTOCOL_NETB
- SQL_PROTOCOL_NPIPE
- SQL_PROTOCOL_SOCKS
- SQL_PROTOCOL_TCPIP

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqle_node_struct
{
        unsigned short         struct_id;
        unsigned short         codepage;
        _SQLOLDCHAR comment[SQL_CMT_SZ + 1];
        _SQLOLDCHAR nodename[SQL_NNAME_SZ + 1];
        unsigned char  protocol;
};
```

Estructura en COBOL

```
* Archivo: sqlenv.cbl
01 SQL-NODE-STRUCT.
    05 STRUCT-ID             PIC 9(4) COMP-5.
    05 CODEPAGE              PIC 9(4) COMP-5.
    05 COMMENT               PIC X(30).
    05 FILLER                PIC X.
    05 NODENAME              PIC X(8).
    05 FILLER                PIC X.
    05 PROTOCOL              PIC X.
    05 FILLER                PIC X(1).
*
```

sqle_node_tcpip

Esta estructura se utiliza para catalogar nodos TCP/IP para la API sqlectnd.

Nota: Para catalogar un nodo TCP/IP, TCP/IPv4 o TCP/IPv6, en la estructura del directorio de nodos, establezca el tipo de protocolo en SQL_PROTOCOL_TCPIP, SQL_PROTOCOL_TCPIP4 o SQL_PROTOCOL_TCPIP6 respectivamente en la estructura SQLE-NODE-STRUCT antes de invocar la API sqlectnd. Para catalogar un nodo SOCKS de TCP/IP o TCP/IPv4, en la estructura del directorio de nodos,

establezca el tipo de protocolo en SQL_PROTOCOL_SOCKS o SQL_PROTOCOL_SOCKS4 respectivamente en la estructura SQLE-NODE-STRUCT antes de invocar la API sqlectnd. SOCKS no está soportado en IPv6. Por ejemplo, SQL_PROTOCOL_SOCKS no se puede utilizar con una dirección IPv6.

Tabla 48. Campos de la estructura SQLE-NODE-TCPIP

Nombre de campo	Tipo de datos	Descripción
HOSTNAME	CHAR(255)	Nombre de sistema principal o dirección IP en el que reside la instancia de servidor DB2. El tipo de dirección IP aceptada depende del protocolo seleccionado.
SERVICE_NAME	CHAR(14)	Nombre de servicio TCP/IP o número de puerto asociado de la instancia de servidor DB2.

Nota: Los campos de caracteres pasados en esta estructura deben tener una terminación nula o estar rellenados con blancos hasta completar la longitud del campo.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqle_node_tcpip
{
        _SQLOLDCHAR hostname[SQL_HOSTNAME_SZ+1];
        _SQLOLDCHAR service_name[SQL_SERVICE_NAME_SZ+1];
};
```

Estructura en COBOL

```
* Archivo: sqlenv.cbl
01 SQL-NODE-TCPIP.
    05 HOSTNAME              PIC X(255).
    05 FILLER                PIC X.
    05 SERVICE-NAME          PIC X(14).
    05 FILLER                PIC X.
*
```

sqledbdesc

La estructura de Bloque de descripción de base de datos (SQLEDBDESC) puede utilizarse durante una llamada a la API sqlecrea para especificar valores permanentes para atributos de base de datos. Estos atributos incluyen el comentario de base de datos, los órdenes de clasificación y las definiciones de espacio de tablas.

Tabla 49. Campos de la estructura SQLEDBDESC

Nombre de campo	Tipo de datos	Descripción
SQLDBDID	CHAR(8)	Identificador de estructura y ″captador de atención″ de vuelcos de almacenamiento. Es una serie de ocho bytes que debe inicializarse con el valor de SQLE_DBDESC_2 (definido en sqlenv). El contenido de este campo se valida para el control de versiones.
SQLDBCCP	INTEGER	Página de códigos del comentario de base de datos. El gestor de bases de datos ya no utiliza este valor.
SQLDBCSS	INTEGER	Valor que indica el origen del orden de clasificación de la base de datos. Consulte más abajo para conocer los valores. Nota: especifique SQL_CS_NONE para indicar que el orden de clasificación de la base de datos es IDENTITY (que implementa un orden de clasificación binario). SQL_CS_NONE es el valor por omisión.
SQLDBUDC	CHAR(256)	Si SQLDBCSS se establece en SQL_CS_USER, el byte número *n* contiene el peso de clasificación del punto de código cuya representación decimal subyacente es el carácter n de la página de códigos de la base de datos. Si SQLDBCSS se establece en SQL_CS_UNICODE, este campo contiene el nombre de clasificación que tiene en cuenta el idioma o basada en UCA sensible al entorno local (una serie terminada en NULL de hasta 128 bytes de longitud). Si SQLDBCSS no es igual a SQL_CS_USER o SQL_CS_UNICODE, este campo se pasa por alto.
SQLDBCMT	CHAR(30)	El comentario de la base de datos.
SQLDBSGP	INTEGER	Campo reservado. Ya no se utiliza.

Tabla 49. Campos de la estructura SQLEDBDESC (continuación)

Nombre de campo	Tipo de datos	Descripción
SQLDBNSG	SHORT	Valor que indica el número de segmentos de archivo que deben crearse en la base de datos. El valor mínimo de este campo es 1 y el valor máximo es 256. Si se especifica un valor -1, este campo tomará por omisión el valor 1. Nota: SQLDBNSG establecido en cero produce un valor por omisión para la compatibilidad con la Versión 1.
SQLTSEXT	INTEGER	Valor, en página de 4 KB, que indica el tamaño de extensión por omisión de cada espacio de tablas de la base de datos. El valor mínimo de este campo es 2 y el valor máximo es 256. Si se especifica un valor -1, este campo tomará por omisión el valor 32.
SQLCATTS	Puntero	Puntero a un bloque de control de descripción de espacio de tablas, SQLETSDESC, que define el espacio de tablas de catálogo. Si es nulo, se creará un espacio de tablas de catálogo por omisión basado en los valores de SQLTSEXT y SQLDBNSG.
SQLUSRTS	Puntero	Puntero a un bloque de control de descripción de espacio de tablas, SQLETSDESC, que define el espacio de tablas de usuario. Si es nulo, se creará un espacio de tablas de usuario por omisión basado en los valores de SQLTSEXT y SQLDBNSG.
SQLTMPTS	Puntero	Puntero a un bloque de control de descripción de espacio de tablas, SQLETSDESC, que define el espacio de tablas temporal del sistema. Si es nulo, se creará un espacio de tablas temporal del sistema por omisión basado en los valores de SQLTSEXT y SQLDBNSG.

La estructura de bloque de descripción de espacio de tablas (SQLETSDESC) se utiliza para especificar los atributos de cualquiera de los tres espacios de tablas iniciales.

Tabla 50. Campos de la estructura SQLETSDESC

Nombre de campo	Tipo de datos	Descripción
SQLTSDID	CHAR(8)	Identificador de estructura y ″captador de atención″ de vuelcos de almacenamiento. Es una serie de ocho bytes que debe inicializarse con el valor de SQLE_DBTSDESC_1 (definido en sqlenv). El contenido de este campo se valida para el control de versiones.
SQLEXTNT	INTEGER	Tamaño de extensión del espacio de tablas, en páginas de 4 KB. Si se especifica un valor -1, este campo tomará por omisión el valor actual del parámetro de configuración dft_extent_sz.
SQLPRFTC	INTEGER	Tamaño de captación previa del espacio de tablas, en páginas de 4 KB. Si se especifica un valor -1, este campo tomará por omisión el valor actual del parámetro de configuración dft_prefetch_sz.
SQLFSCACHING	UNSIGNED CHAR	Antememoria del sistema de archivos. Si se especifica un valor 1, la antememoria del sistema de archivos estará desactivada (OFF) para el espacio de tablas actual. Si se especifica un valor 0, la antememoria del sistema de archivos estará activada (ON) para el espacio de tablas actual. Especifique 2 para indicar el valor por omisión. EN este caso, la antememoria del sistema de archivos estará desactivada (OFF) en AIX, Linux, Solaris y Windows excepto en AIX JFS, Linux en System z, archivos del espacio de tablas temporal de Solaris no-VxFS para SMS y para archivos de objetos grandes SMS o archivos grandes. La antememoria del sistema de archivos estará activada (ON) para todas las demás plataformas.

Tabla 50. Campos de la estructura SQLETSDESC (continuación)

Nombre de campo	Tipo de datos	Descripción
SQLPOVHD	DOUBLE	Actividad general de E/S de espacio de tablas, en milisegundos. Si se especifica un valor -1, este campo tomará por omisión un valor interno del gestor de bases de datos (actualmente 24.1 ms) que puede cambiar en releases futuros.
SQLTRFRT	DOUBLE	Velocidad de transferencia de E/S de espacio de tablas, en milisegundos. Si se especifica un valor -1, este campo tomará por omisión un valor interno del gestor de bases de datos (actualmente 0.9 ms) que puede cambiar en releases futuros.
SQLTSTYP	CHAR(1)	Indica si el espacio de tablas es gestionado por el sistema o gestionado por la base de datos. Consulte más abajo para conocer los valores.
SQLCCNT	SMALLINT	Número de contenedores asignados al espacio de tablas. Indica cuántos valores SQLCTYPE/SQLCSIZE/ SQLCLEN/SQLCONTR siguen.
CONTAINR	Matriz	Matriz de estructuras sqlccnt SQLETSCDESC.

Tabla 51. Campos de la estructura SQLETSCDESC

Nombre de campo	Tipo de datos	Descripción
SQLCTYPE	CHAR(1)	Identifica el tipo de este contenedor. Consulte más abajo para conocer los valores.
SQLCSIZE	INTEGER	Tamaño del contenedor identificado en SQLCONTR, especificado en páginas de 4 KB. Sólo es válido cuando SQLTSTYP está establecido en SQL_TBS_TYP_DMS.
SQLCLEN	SMALLINT	Longitud del valor SQLCONTR siguiente.
SQLCONTR	CHAR(256)	Serie del contenedor.

Los valores válidos para SQLDBCSS (definido en sqlenv) son:

SQL_CS_SYSTEM
Para bases de datos no Unicode, es la opción por omisión, con la secuencia

de clasificación basada en el territorio de la base de datos. Para bases de datos Unicode, esta opción es equivalente a la opción IDENTITY. Si se pasa un puntero NULO, se utiliza la secuencia de clasificación del sistema operativo (basada en el código de entorno local actual y en la página de códigos). Es igual que especifica un SQLDBCSS igual a SQL_CS_SYSTEM (0).

SQL_CS_USER
El orden de clasificación se especifica mediante la tabla de pesos de 256 bytes suministrada por el usuario. Cada peso de la tabla tiene un byte de longitud.

SQL_CS_NONE
Secuencia de clasificación de identidad, en la que se comparan las series byte por byte. Este es el valor por omisión para bases de datos Unicode.

SQL_CS_COMPATABILITY
Utilizar orden de clasificación pre-Versión.

SQL_CS_SYSTEM_NLSCHAR
Orden de clasificación del sistema que utiliza la versión NLS de rutinas de comparación para tipos de caracteres. Este valor sólo se puede especificar al crear una base de datos Thai TIS620-1.

SQL_CS_USER_NLSCHAR
El orden de clasificación se especifica mediante la tabla de pesos de 256 bytes suministrada por el usuario. Cada peso de la tabla tiene un byte de longitud. Este valor sólo se puede especificar al crear una base de datos Thai TIS620-1.

SQL_CS_IDENTITY_16BIT
Orden de clasificación CESU-8 (Esquema de codificación de compatibilidad para UTF-16: 8-Bits) tal como está especificado en Unicode Technical Report #26, que se encuentra disponible en el sitio web de Unicode Consortium (www.unicode.org). Este valor sólo se puede especificar al crear una base de datos Unicode.

SQL_CS_UCA400_NO
La secuencia de clasificación de UCA (Unicode Collation Algorithm) basada en la versión 4.0.0 del estándar Unicode con la normalización implícitamente activada (establecida en 'on'). Encontrará detalles sobre UCA en el documento Unicode Technical Standard #10, que se encuentra disponible en el sitio web de Unicode Consortium (www.unicode.org). Este valor sólo se puede especificar al crear una base de datos Unicode.

SQL_CS_UCA400_LSK
La secuencia de clasificación de UCA (Unicode Collation Algorithm) basada en la versión 4.0.0 del estándar Unicode, pero ordenará los caracteres eslovacos en el orden correcto. Encontrará detalles sobre UCA en el documento Unicode Technical Standard #10, que se encuentra disponible en el sitio Web de Unicode Consortium (www.unicode.org). Este valor sólo se puede especificar al crear una base de datos Unicode.

SQL_CS_UCA400_LTH
La secuencia de clasificación de UCA (Unicode Collation Algorithm) basada en la versión 4.0.0 del estándar Unicode, pero clasificará todos los caracteres tailandeses según el orden del Diccionario Real de tailandés. Encontrará detalles sobre UCA en el documento Unicode Technical

Standard #10, que se encuentra disponible en el sitio web de Unicode Consortium (www.unicode.org). Este valor sólo se puede especificar al crear una base de datos Unicode.

SQL_CS_UNICODE

La secuencia de clasificación se basa en el idioma para una base de datos basada en Unicode. El nombre de la clasificación específica se especifica en el campo SQLDBUDC y debe terminar en un byte 0x00. El nombre de clasificación puede identificar cualquier clasificación consciente del idioma definida en ″Clasificaciones de datos Unicode que tienen en cuenta el idioma″ o cualquier clasificación basada en UCA sensible al entorno local identificada en ″Clasificaciones basadas en el algoritmo de clasificación Unicode″.

Por ejemplo, para utilizar la clasificación equivalente al inglés de EE.UU. de la página de códigos 819, establezca SQLDBCSS en SQL_CS_UNICODE y SQLDBUDC en SYSTEM_819_US.

Nota: Cuando se ejecuta CREATE DATABASE en un servidor anterior a la versión 9.5, no se puede utilizar esta opción. Por omisión, una base de datos Unicode en este tipo de servidor se creará con la clasificación SYSTEM.

Los valores válidos para SQLTSTYP (definido en sqlenv) son:

SQL_TBS_TYP_SMS
Gestionado por el sistema

SQL_TBS_TYP_DMS
Gestionado por la base de datos

Los valores válidos para SQLCTYPE (definido en sqlenv) son:

SQL_TBSC_TYP_DEV
Dispositivo. Sólo es válido cuando SQLTSTYP = SQL_TBS_TYP_DMS.

SQL_TBSC_TYP_FILE
Archivo. Sólo es válido cuando SQLTSTYP = SQL_TBS_TYP_DMS.

SQL_TBSC_TYP_PATH
Vía de acceso (directorio). Sólo es válido cuando SQLTSTYP = SQL_TBS_TYP_SMS.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqledbdesc
{
        _SQLOLDCHAR sqldbdid[8];
        sqlint32 sqldbccp;
        sqlint32 sqldbcss;
        unsigned char  sqldbudc[SQL_CS_SZ];
        _SQLOLDCHAR sqldbcmt[SQL_CMT_SZ+1];
        _SQLOLDCHAR pad[1];
        sqluint32 sqldbsgp;
        short sqldbnsg;
        char pad2[2];
        sqlint32 sqltsext;
        struct SQLETSDESC *sqlcatts;
        struct SQLETSDESC *sqlusrts;
        struct SQLETSDESC *sqltmpts;
};

SQL_STRUCTURE SQLETSDESC
```

```
{
        char sqltsdid[8];
        sqlint32 sqlextnt;
        sqlint32 sqlprftc;
        double sqlpovhd;
        double sqltrfrt;
        char sqltstyp;
        unsigned char  sqlfscaching;
        short sqlccnt;
        struct SQLETSCDESC containr[1];
};

SQL_STRUCTURE SQLETSCDESC
{
        char sqlctype;
        char pad1[3];
        sqlint32 sqlcsize;
        short sqlclen;
        char sqlcontr[SQLB_MAX_CONTAIN_NAME_SZ];
        char pad2[2];
};
```

Parámetros de la estructura sqledbdesc

pad1 Reservado. Se utiliza para la alineación de estructuras y no debe llenarse con datos del usuario.

pad2 Reservado. Se utiliza para la alineación de estructuras y no debe llenarse con datos del usuario.

Parámetros de la estructura SQLETSCDESC

pad1 Reservado. Se utiliza para la alineación de estructuras y no debe llenarse con datos del usuario.

pad2 Reservado. Se utiliza para la alineación de estructuras y no debe llenarse con datos del usuario.

Estructura en COBOL

```
* Archivo: sqlenv.cbl
01 SQLEDBDESC.
    05 SQLDBDID              PIC X(8).
    05 SQLDBCCP              PIC S9(9) COMP-5.
    05 SQLDBCSS              PIC S9(9) COMP-5.
    05 SQLDBUDC              PIC X(256).
    05 SQLDBCMT              PIC X(30).
    05 FILLER                PIC X.
    05 SQL-PAD               PIC X(1).
    05 SQLDBSGP              PIC 9(9) COMP-5.
    05 SQLDBNSG              PIC S9(4) COMP-5.
    05 SQL-PAD2              PIC X(2).
    05 SQLTSEXT              PIC S9(9) COMP-5.
    05 SQLCATTS              USAGE IS POINTER.
    05 SQLUSRTS              USAGE IS POINTER.
    05 SQLTMPTS              USAGE IS POINTER.
*


* Archivo: sqletsd.cbl
01 SQLETSDESC.
    05 SQLTSDID              PIC X(8).
    05 SQLEXTNT              PIC S9(9) COMP-5.
    05 SQLPRFTC              PIC S9(9) COMP-5.
    05 SQLPOVHD              USAGE COMP-2.
    05 SQLTRFRT              USAGE COMP-2.
    05 SQLTSTYP              PIC X.
```

```
    05 SQL-PAD1              PIC X.
    05 SQLCCNT               PIC S9(4) COMP-5.
    05 SQL-CONTAINR OCCURS 001 TIMES.
        10 SQLCTYPE          PIC X.
        10 SQL-PAD1          PIC X(3).
        10 SQLCSIZE          PIC S9(9) COMP-5.
        10 SQLCLEN           PIC S9(4) COMP-5.
        10 SQLCONTR          PIC X(256).
        10 SQL-PAD2          PIC X(2).
*


* Archivo: sqlenv.cbl
01 SQLETSCDESC.
    05 SQLCTYPE              PIC X.
    05 SQL-PAD1              PIC X(3).
    05 SQLCSIZE              PIC S9(9) COMP-5.
    05 SQLCLEN               PIC S9(4) COMP-5.
    05 SQLCONTR              PIC X(256).
    05 SQL-PAD2              PIC X(2).
*
```

sqledbdescext

La estructura de bloque de descripción de base de datos ampliado (sqledbdescext) se emplea durante una llama a la API sqlecrea para especificar valores permanentes para los atributos de la base de datos. El bloque de descripción de base de datos ampliado permite el almacenamiento automático de una base de datos, elige un tamaño de página por omisión para la base de datos o especifica valores para los nuevos atributos de espacio de tablas que se han introducido. Esta estructura se utiliza además de la estructura de bloque de descripción de base de datos (sqledbdesc), no en lugar de ella.

Si esta estructura no se pasa a la API sqlecrea, se utiliza el siguiente comportamiento:

- El almacenamiento automático está habilitado para la base de datos
- El tamaño de página por omisión de la base de datos es de 4096 bytes (4 KB)
- Si procede, los sistemas de base de datos DB2 determinan automáticamente el valor de los atributos de espacio de tablas ampliado

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqledbdescext
{
        sqluint32 sqlPageSize;
        struct sqleAutoStorageCfg *sqlAutoStorage;
        struct SQLETSDESCEXT *sqlcattsext;
        struct SQLETSDESCEXT *sqlusrtsext;
        struct SQLETSDESCEXT *sqltmptsext;
        void *reserved;
};

SQL_STRUCTURE sqleAutoStorageCfg
{
        char sqlEnableAutoStorage;
        char pad[3];
        sqluint32 sqlNumStoragePaths;
        char           **sqlStoragePaths;
};

SQL_STRUCTURE SQLETSDESCEXT
{
```

```
        sqlint64 sqlInitSize;
        sqlint64 sqlIncreaseSize;
        sqlint64 sqlMaximumSize;
        char sqlAutoResize;
        char sqlInitSizeUnit;
        char sqlIncreaseSizeUnit;
        char sqlMaximumSizeUnit;
};

SQL_STRUCTURE sqledboptions
{
        void *piAutoConfigInterface;
        sqlint32 restrictive;
        void *reserved;
};
```

Parámetros de la estructura de datos sqledbdescext

Tabla 52. Campos de la estructura sqledbdescext

Nombre de campo	Tipo de datos	Descripción
SQLPAGESIZE	sqluint32	Especifica el tamaño de página de la agrupación de almacenamientos intermedios por omisión, así como los espacios de tablas iniciales (SYSCATSPACE, TEMPSPACE1, USERSPACE1) cuando se crea la base de datos. El valor proporcionado también representa el tamaño de página por omisión para todas las sentencias CREATE BUFFERPOOL y CREATE TABLESPACE futuras. Los valores se indican en la información que figura después de esta tabla.
SQLAUTOSTORAGE	Puntero	Puntero a una estructura de configuración de almacenamiento automático. El puntero habilita o inhabilita el almacenamiento automático de la base de datos. Si se proporciona un puntero, el almacenamiento automático puede habilitarse o inhabilitarse. Si el valor es NULL, se habilita el almacenamiento automático y se presupone una sola vía de acceso con un valor determinado por el parámetro `dbpath` transferido o el parámetro de configuración del gestor de bases de datos, `dftdbpath`.
SQLCATTSEXT	Puntero	Puntero a un bloque de control de descripción de espacio de tablas ampliado (SQLETSDESCEXT) para el espacio de tablas de catálogos del sistema, que define atributos adicionales a los que se encuentran en SQLETSDESC. Si el valor es NULL, el gestor de bases de datos determina automáticamente el valor de estos e atributos (si procede).
SQLUSRTSEXT	Puntero	Puntero a un bloque de control de descripción de espacio de tablas ampliado (SQLETSDESCEXT) para el espacio de tablas de usuarios, que define atributos adicionales a los que se encuentran en SQLETSDESC. Si el valor es NULL, el gestor de bases de datos determina automáticamente el valor de estos e atributos (si procede).

Tabla 52. Campos de la estructura sqledbdescext (continuación)

Nombre de campo	Tipo de datos	Descripción
SQLTMPTSEXT	Puntero	Puntero a un bloque de control de descripción de espacio de tablas ampliado (SQLETSDESCEXT) para el espacio de tablas temporal del sistema, que define atributos adicionales a los que se encuentran en SQLETSDESC. Si el valor es NULL, el gestor de bases de datos determina automáticamente el valor de estos e atributos (si procede).
RESERVED	Puntero	Puntero a un bloque de control de opciones de base de datos (sqledboptions).

Los valores válidos de SQLPAGESIZE (definido en sqlenv) son:

SQL_PAGESIZE_4K
Tamaño de página por omisión de la base de datos igual a 4096 bytes.

SQL_PAGESIZE_8K
Tamaño de página por omisión de la base de datos igual a 8.192 bytes.

SQL_PAGESIZE_16K
Tamaño de página por omisión de la base de datos igual a 16.384 bytes.

SQL_PAGESIZE_32K
Tamaño de página por omisión de la base de datos igual a 32.768 bytes.

Parámetros de la estructura de datos de configuración de almacenamiento automático (sqleAutoStorageCfg)

La estructura de configuración de almacenamiento automático (sqleAutoStorageCfg) se puede usar durante una llamada a la API sqlecrea. Es un elemento de la estructura sqledbdescext y especifica si el almacenamiento automático está habilitado o no para la base de datos.

Tabla 53. Campos de la estructura sqleAutoStorageCfg

Nombre de campo	Tipo de datos	Descripción
SQLENABLEAUTOSTORAGE	CHAR(1)	Especifica si el almacenamiento automático se habilita o no para la base de datos. Los valores se indican en la información que figura después de esta tabla.
SQLNUMSTORAGEPATHS	sqluint32	Valor que indica el número de vías de acceso de almacenamiento hacia las que señala la matriz SQLSTORAGEPATHS. Si el valor es 0, el puntero SQLSTORAGEPATHS debe ser NULL. El número máximo de vías de acceso de almacenamiento es 128 (SQL_MAX_STORAGE_PATHS).

Tabla 53. Campos de la estructura sqleAutoStorageCfg (continuación)

Nombre de campo	Tipo de datos	Descripción
SQLSTORAGEPATHS	Puntero	Matriz de punteros de tipo serie que apunta a las vías de acceso de almacenamiento. El número de punteros de la matriz se refleja en SQLNUMSTORAGEPATHS. El valor de SQLSTORAGEPATHS debe ser NULL si no se proporcionan vías de acceso de almacenamiento (en cuyo caso el valor de SQLNUMSTORAGEPATHS debe ser 0). La longitud máxima de una vía de acceso es de 175 caracteres.

Los valores válidos de SQLENABLEAUTOSTORAGE (definido en sqlenv) son:

SQL_AUTOMATIC_STORAGE_NO
El almacenamiento automático está inhabilitado para la base de datos. Cuando se utiliza este valor, hay que establecer que SQLNUMSTORAGEPATHS sea igual a 0 y que SQLSTORAGEPATHS sea igual a NULL.

SQL_AUTOMATIC_STORAGE_YES
El almacenamiento automático está habilitado para la base de datos. Las vías de acceso de almacenamiento empleadas para el almacenamiento automático se especifican con el puntero SQLSTORAGEPATHS. Si este puntero es NULL, se presupone una sola vía de acceso de almacenamiento con un valor que viene determinado por el parámetro dftdbpath de configuración del gestor de bases de datos.

SQL_AUTOMATIC_STORAGE_DFT
El gestor de bases de datos determina si el almacenamiento automático se habilita o no. Actualmente, la elección se realiza en función del valor del puntero SQLSTORAGEPATHS. Si el puntero es NULL, el almacenamiento automático no se habilita; de lo contrario, se habilita. El valor por omisión equivale a SQL_AUTOMATIC_STORAGE_YES.

Parámetros de la estructura de bloque de descripción de espacio de tablas ampliado (SQLETSDESCEXT)

La estructura de bloque de descripción de espacio de tablas ampliado (SQLETSDESCEXT) sirve para especificar los atributos de los tres espacios de tablas iniciales. Esta estructura se utiliza además de la estructura de bloque de descripción de espacio de tablas (SQLETSDESC), no en lugar de ella.

Tabla 54. Campos de la estructura SQLETSDESCEXT

Nombre de campo	Tipo de datos	Descripción
SQLINITSIZE	sqlint64	Define el tamaño inicial de cada espacio de tablas que utiliza el almacenamiento automático. Este campo solo es pertinente para los espacios de tablas de almacenamiento automático grandes o regulares. Utilice el valor SQL_TBS_AUTOMATIC_INITSIZE para los otros tipos de espacios de tablas o si la intención es que DB2 determine automáticamente un tamaño inicial. Nota: el valor real que utiliza el gestor de bases de datos puede ser ligeramente mayor o menor que el especificado. Se toma esta medida para mantener la coherencia de los tamaños en todos los contenedores del espacio de tablas, teniendo en cuenta que el valor proporcionado podría no mantener la coherencia.
SQLINCREASESIZE	sqlint64	Define el incremento de tamaño que el gestor de bases de datos proporciona automáticamente al espacio de tablas cuando este queda lleno. Este campo solo es pertinente para los espacios de tablas que tengan habilitado el redimensionamiento automático. Utilice el valor SQL_TBS_AUTOMATIC_INCSIZE si el redimensionamiento automático está inhabilitado o si la intención es que el gestor de bases de datos determine automáticamente el incremento de tamaño. Nota: el valor real que utiliza el gestor de bases de datos puede ser ligeramente mayor o menor que el especificado. Se toma esta medida para mantener la coherencia de los tamaños en todos los contenedores del espacio de tablas, teniendo en cuenta que el valor proporcionado podría no mantener la coherencia.

Tabla 54. Campos de la estructura SQLETSDESCEXT (continuación)

Nombre de campo	Tipo de datos	Descripción
SQLMAXIMUMSIZE	sqlint64	Define el tamaño máximo que el gestor de bases de datos proporciona automáticamente al espacios de tablas. Por otro lado, se puede usar el valor SQL_TBS_NO_MAXSIZE para especificar que el tamaño máximo es ″ilimitado″, en cuyo caso el espacio de tablas puede aumentar hasta el límite que permita la arquitectura del espacio de tablas o hasta que se llegue a una situación de ″sistema de archivos lleno″. Este campo solo es pertinente para los espacios de tablas que tengan habilitado el redimensionamiento automático. Utilice el valor SQL_TBS_AUTOMATIC_MAXSIZE si el redimensionamiento automático está inhabilitado o si la intención es que el gestor de bases de datos determine automáticamente el tamaño máximo. Nota: el valor real que utiliza el gestor de bases de datos puede ser ligeramente mayor o menor que el especificado. Se toma esta medida para mantener la coherencia de los tamaños en todos los contenedores del espacio de tablas, teniendo en cuenta que el valor proporcionado podría no mantener la coherencia.
SQLAUTORESIZE	CHAR(1)	Especifica si se habilita o no el redimensionamiento automático para el espacio de tablas. Los valores se indican en la información que figura después de esta tabla.
SQLINITSIZEUNIT	CHAR(1)	Si procede, indica si el valor SQLINITSIZE se expresa en bytes, kilobytes, megabytes o gigabytes. Los valores se indican en la información que figura después de esta tabla.
SQLINCREASESIZEUNIT	CHAR(1)	Si procede, indica si el valor SQLINCREASESIZE se expresa en bytes, kilobytes, megabytes, gigabytes o como porcentaje. Los valores se indican en la información que figura después de esta tabla.
SQLMAXIMUMSIZEUNIT	CHAR(1)	Si procede, indica si el valor SQLMAXIMUMSIZE se expresa en bytes, kilobytes, megabytes o gigabytes. Los valores se indican en la información que figura después de esta tabla.

Los valores válidos de SQLAUTORESIZE (definido en sqlenv) son:

SQL_TBS_AUTORESIZE_NO
El redimensionamiento automático está inhabilitado para el espacio de tablas. Este valor solo se puede especificar para los espacios de tablas gestionados por base de datos (DMS) o para los espacios de tablas de almacenamiento automático.

SQL_TBS_AUTORESIZE_YES
El redimensionamiento automático está habilitado para el espacio de

tablas. Este valor solo se puede especificar para los espacios de tablas gestionados por base de datos (DMS) o para los espacios de tablas de almacenamiento automático.

SQL_TBS_AUTORESIZE_DFT
El gestor de bases de datos determina si el redimensionamiento automático se habilita o no en función del tipo de espacio de tablas: el redimensionamiento automático se desactiva para los espacios de tablas gestionados por base de datos (DMS) y se activa para los espacios de tablas de almacenamiento automático. Utilice este valor para los espacios de tablas gestionados por el sistema (SMS), ya que el redimensionamiento automático no es aplicable para ese tipo de espacio de tablas.

Los valores válidos de SQLINITSIZEUNIT, SQLINCREASESIZEUNIT y SQLMAXIMUMSIZEUNIT (definidos en sqlenv) son:

SQL_TBS_STORAGE_UNIT_BYTES
El valor especificado en el correspondiente campo de tamaño se expresa en bytes.

SQL_TBS_STORAGE_UNIT_KILOBYTES
El valor especificado en el correspondiente campo de tamaño se expresa en kilobytes (1 kilobyte = 1024 bytes).

SQL_TBS_STORAGE_UNIT_MEGABYTES
El valor especificado en el correspondiente campo de tamaño se expresa en megabytes (1 megabyte = 1.048.576 de bytes)

SQL_TBS_STORAGE_UNIT_GIGABYTES
El valor especificado en el correspondiente campo de tamaño se expresa en gigabytes (1 gigabyte = 1.073.741.824 de bytes)

SQL_TBS_STORAGE_UNIT_PERCENT
El valor especificado en el correspondiente campo de tamaño se expresa como porcentaje (valores comprendidos entre 1 y 100). Este valor solo es válido para SQLINCREASESIZEUNIT.

Parámetros de la estructura de datos sqledboptions

piAutoConfigInterface
Entrada. Puntero a la estructura db2AutoConfigInterface con información que sirve a modo de datos de entrada para el asesor de configuración.

restrictive
El valor del campo restrictive se almacena en el parámetro RESTRICT_ACCESS de configuración de base de datos y afectará a todas las migraciones futuras de esta base de datos. Es decir, cuando una base de datos se migra a un release ulterior de DB2, el programa de utilidad de migración comprueba el valor del parámetro RESTRICT_ACCESS de configuración de base de datos para determinar si el conjunto restrictivo de acciones por omisión se tiene que aplicar a los nuevos objetos (por ejemplo, nuevas tablas de catálogos del sistema) que se hayan introducido en el nuevo release de DB2.

Los valores válidos (definidos en el archivo de cabecera sqlenv, ubicado en el directorio de inclusión) de este parámetro son:

SQL_DB_RESTRICT_ACCESS_NO o
SQL_DB_RESTRICT_ACCESS_DFT
Indica que la base de datos se tiene que crear sin usar el conjunto

restrictivo de acciones por omisión. Este valor hará que se otorguen los siguientes privilegios a PUBLIC:

- Privilegio CREATETAB
- Privilegio BINDADD
- Privilegio CONNECT
- Privilegio IMPLSCHEMA
- Privilegio EXECUTE con GRANT sobre todos los procedimientos del esquema SQLJ
- Privilegio EXECUTE con GRANT sobre todas las funciones y procedimientos del esquema SYSPROC
- Privilegio BIND sobre todos los paquetes creados en el esquema NULLID
- Privilegio EXECUTE sobre todos los paquetes creados en el esquema NULLID
- Privilegio CREATEIN sobre el esquema SQLJ
- Privilegio CREATEIN sobre el esquema NULLID
- Privilegio USE en el espacio de tablas USERSPACE1
- Privilegio SELECT sobre las tablas de catálogos SYSIBM
- Privilegio SELECT sobre las vistas de catálogos SYSCAT
- Privilegio SELECT sobre las vistas de catálogos SYSSTAT
- Privilegio UPDATE sobre las vistas de catálogos SYSSTAT

SQL_DB_RESTRICT_ACCESS_YES
Indica que la base de datos se tiene que crear usando el conjunto restrictivo de acciones por omisión. Esto quiere decir que no se llevan a cabo las acciones de otorgar que figuran en la lista anterior bajo SQL_DB_RESTRICT_ACCESS_NO.

reserved
Reservado para una utilización futura.

sqledbterritoryinfo

Esta estructura se utiliza para proporcionar opciones de juego de códigos y de territorio a la API sqlecrea.

Tabla 55. Campos de la estructura SQLEDBTERRITORYINFO

Nombre de campo	Tipo de datos	Descripción
SQLDBCODESET	CHAR(9)	Conjunto de códigos de base de datos.
SQLDBLOCALE	CHAR(5)	Territorio de la base de datos.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqledbcountryinfo
{
        char sqldbcodeset[SQL_CODESET_LEN + 1];
        char sqldblocale[SQL_LOCALE_LEN + 1];
};
typedef SQL_STRUCTURE sqledbcountryinfo SQLEDBTERRITORYINFO;
```

Estructura en COBOL

```
* Archivo: sqlenv.cbl
01 SQLEDBTERRITORYINFO.
    05 SQLDBCODESET          PIC X(9).
    05 FILLER                PIC X.
    05 SQLDBLOCALE           PIC X(5).
    05 FILLER                PIC X.
*
```

sqleninfo

Esta estructura devuelve información después de una llamada a la API slqlengne.

Nota: NetBIOS ya no está soportado. SNA, incluyendo sus API APPC, APPN y CPI-C, tampoco está soportado. Si utiliza estos protocolos, debe volver a catalogar los nodos y bases de datos utilizando un protocolo soportado como, por ejemplo, TCP/IP. Las referencias a estos protocolos se deben pasar por alto.

Tabla 56. Campos de la estructura SQLENINFO

Nombre de campo	Tipo de datos	Descripción
NODENAME	CHAR(8)	Se utiliza para el protocolo NetBIOS; es el nombre del nodo donde reside la base de datos (solamente es válido en el directorio del sistema)
LOCAL_LU	CHAR(8)	Se utiliza para el protocolo APPN; unidad lógica local.
PARTNER_LU	CHAR(8)	Se utiliza para el protocolo APPN; unidad lógica asociada.
MODE	CHAR(8)	Se utiliza para el protocolo APPN; modalidad de servicio de transmisión.
COMMENT	CHAR(30)	El comentario asociado con el nodo.
COM_CODEPAGE	SMALLINT	Página de códigos del comentario. El gestor de bases de datos ya no utiliza este campo.
ADAPTER	SMALLINT	Se utiliza para el protocolo NetBIOS; es el adaptador de red local.
NETWORKID	CHAR(8)	Se utiliza para el protocolo APPN; ID de red.
PROTOCOL	CHAR(1)	Protocolo de comunicaciones.
SYM_DEST_NAME	CHAR(8)	Se utiliza para el protocolo APPC; nombre de destino simbólico.
SECURITY_TYPE	SMALLINT	Se utiliza para el protocolo APPC; el tipo de seguridad. Consulte más abajo para conocer los valores.

Tabla 56. Campos de la estructura SQLENINFO (continuación)

Nombre de campo	Tipo de datos	Descripción
HOSTNAME	CHAR(255)	Se utiliza para el protocolo TCP/IP; nombre del sistema principal TCP/IP o dirección IPv4 o IPv6 donde reside la instancia del servidor DB2.
SERVICE_NAME	CHAR(14)	Se utiliza para el protocolo TCP/IP; nombre de servicio TCP/IP o número de puerto asociado de la instancia del servidor DB2.
FILESERVER	CHAR(48)	Se utiliza para el protocolo IPX/SPX; nombre del servidor de archivos NetWare donde está registrada la instancia del servidor DB2.
OBJECTNAME	CHAR(48)	La instancia del servidor del gestor de bases de datos se representa como objeto, nombre_objeto, en el servidor de archivos NetWare. La dirección interred IPX/SPX del servidor se almacena y recupera de este objeto.
INSTANCE_NAME	CHAR(8)	Se utiliza para los protocolos local y NPIPE; es el nombre de la instancia de servidor.
COMPUTERNAME	CHAR(15)	Utilizado por el protocolo NPIPE; nombre de sistema del nodo de servidor.
SYSTEM_NAME	CHAR(21)	Nombre del sistema DB2 del servidor remoto.
REMOTE_INSTNAME	CHAR(8)	Nombre de la instancia del servidor DB2.
CATALOG_NODE_TYPE	CHAR	Tipo de nodo de catálogo.
OS_TYPE	UNSIGNED SHORT	Identifica el sistema operativo del servidor.

Nota: Los campos de caracteres devueltos se rellenan con blancos hasta completar la longitud del campo.

Los valores válidos para SECURITY_TYPE (definido en sqlenv) son:

- SQL_CPIC_SECURITY_NONE
- SQL_CPIC_SECURITY_SAME
- SQL_CPIC_SECURITY_PROGRAM

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqleninfo
{
        _SQLOLDCHAR nodename[SQL_NNAME_SZ];
```

```
        _SQLOLDCHAR local_lu[SQL_LOCLU_SZ];
        _SQLOLDCHAR partner_lu[SQL_RMTLU_SZ];
        _SQLOLDCHAR mode[SQL_MODE_SZ];
        _SQLOLDCHAR comment[SQL_CMT_SZ];
        unsigned short com_codepage;
        unsigned short adapter;
        _SQLOLDCHAR networkid[SQL_NETID_SZ];
        _SQLOLDCHAR protocol;
        _SQLOLDCHAR sym_dest_name[SQL_SYM_DEST_NAME_SZ];
        unsigned short security_type;
        _SQLOLDCHAR hostname[SQL_HOSTNAME_SZ];
        _SQLOLDCHAR service_name[SQL_SERVICE_NAME_SZ];
        char fileserver[SQL_FILESERVER_SZ];
        char objectname[SQL_OBJECTNAME_SZ];
        char instance_name[SQL_INSTNAME_SZ];
        char computername[SQL_COMPUTERNAME_SZ];
        char system_name[SQL_SYSTEM_NAME_SZ];
        char remote_instname[SQL_REMOTE_INSTNAME_SZ];
        _SQLOLDCHAR catalog_node_type;
        unsigned short os_type;
        _SQLOLDCHAR chgpwd_lu[SQL_RMTLU_SZ];
        _SQLOLDCHAR transpn[SQL_TPNAME_SZ];
        _SQLOLDCHAR lanaddr[SQL_LANADDRESS_SZ];
};
```

Estructura en COBOL

```
* Archivo: sqlenv.cbl
01 SQLENINFO.
    05 SQL-NODE-NAME          PIC X(8).
    05 SQL-LOCAL-LU           PIC X(8).
    05 SQL-PARTNER-LU         PIC X(8).
    05 SQL-MODE               PIC X(8).
    05 SQL-COMMENT            PIC X(30).
    05 SQL-COM-CODEPAGE       PIC 9(4) COMP-5.
    05 SQL-ADAPTER            PIC 9(4) COMP-5.
    05 SQL-NETWORKID          PIC X(8).
    05 SQL-PROTOCOL           PIC X.
    05 SQL-SYM-DEST-NAME      PIC X(8).
    05 FILLER                 PIC X(1).
    05 SQL-SECURITY-TYPE      PIC 9(4) COMP-5.
    05 SQL-HOSTNAME           PIC X(255).
    05 SQL-SERVICE-NAME       PIC X(14).
    05 SQL-FILESERVER         PIC X(48).
    05 SQL-OBJECTNAME         PIC X(48).
    05 SQL-INSTANCE-NAME      PIC X(8).
    05 SQL-COMPUTERNAME       PIC X(15).
    05 SQL-SYSTEM-NAME        PIC X(21).
    05 SQL-REMOTE-INSTNAME    PIC X(8).
    05 SQL-CATALOG-NODE-TYPE  PIC X.
    05 SQL-OS-TYPE            PIC 9(4) COMP-5.
*
```

sqlfupd

Esta estructura pasa información sobre los archivos de configuración de base de datos y el archivo de configuración del gestor de bases de datos.

Tabla 57. Campos de la estructura SQLFUPD

Nombre de campo	Tipo de datos	Descripción
TOKEN	UINT16	Especifica el valor de configuración que se debe devolver o actualizar.

Tabla 57. Campos de la estructura SQLFUPD (continuación)

Nombre de campo	Tipo de datos	Descripción
PTRVALUE	Puntero	Puntero a un almacenamiento intermedio asignado de aplicaciones que contiene los datos especificados por TOKEN.

Los tipos válidos de datos para el elemento de señal son:

Uint16
Número entero sin signo de 2 bytes

Sint16
Número entero, sin signo, de 2 bytes

Uint32
Número entero, sin signo, de 4 bytes

Sint32
Número entero, sin signo, de 4 bytes

Uint64
Número entero, sin signo, de 8 bytes

float Decimal de coma flotante, de 4 bytes

char(n)
Serie de longitud n (sin incluir terminación con nulos).

Las entradas válidas para el elemento de señal SQLFUPD se indican en la lista siguiente:

Tabla 58. Parámetros actualizables de configuración de base de datos

Nombre de parámetro	Señal	Valor de señal	Tipo de datos
alt_collate	SQLF_DBTN_ALT_COLLATE	809	Uint32
app_ctl_heap_sz	SQLF_DBTN_APP_CTL_HEAP_SZ	500	Uint16
appgroup_mem_sz	SQLF_DBTN_APPGROUP_MEM_SZ	800	Uint32
applheapsz	SQLF_DBTN_APPLHEAPSZ	51	Uint16
archretrydelay	SQLF_DBTN_ARCHRETRYDELAY	828	Uint16
• auto_maint • auto_db_backup • auto_tbl_maint • auto_runstats • auto_stats_prof • auto_prof_upd • auto_reorg	• SQLF_DBTN_AUTO_MAINT • SQLF_DBTN_AUTO_DB_BACKUP • SQLF_DBTN_AUTO_TBL_MAINT • SQLF_DBTN_AUTO_RUNSTATS • SQLF_DBTN_AUTO_STATS_PROF • SQLF_DBTN_AUTO_PROF_UPD • SQLF_DBTN_AUTO_REORG	• 831 • 833 • 835 • 837 • 839 • 844 • 841	Uint16
autorestart	SQLF_DBTN_AUTO_RESTART	25	Uint16
avg_appls	SQLF_DBTN_AVG_APPLS	47	Uint16
blk_log_dsk_ful	SQLF_DBTN_BLK_LOG_DSK_FUL	804	Uint16
catalogcache_sz	SQLF_DBTN_CATALOGCACHE_SZ	56	Sint32
chngpgs_thresh	SQLF_DBTN_CHNGPGS_THRESH	38	Uint16

Tabla 58. Parámetros actualizables de configuración de base de datos (continuación)

Nombre de parámetro	Señal	Valor de señal	Tipo de datos
database_memory	SQLF_DBTN_DATABASE_MEMORY	803	Uint64
dbheap	SQLF_DBTN_DB_HEAP	58	Uint64
db_mem_thresh	SQLF_DBTN_DB_MEM_THRESH	849	Uint16
dft_degree	SQLF_DBTN_DFT_DEGREE	301	Sint32
dft_extent_sz	SQLF_DBTN_DFT_EXTENT_SZ	54	Uint32
dft_loadrec_ses	SQLF_DBTN_DFT_LOADREC_SES	42	Sint16
dft_mttb_types	SQLF_DBTN_DFT_MTTB_TYPES	843	Uint32
dft_prefetch_sz	SQLF_DBTN_DFT_PREFETCH_SZ	40	Sint16
dft_queryopt	SQLF_DBTN_DFT_QUERYOPT	57	Sint32
dft_refresh_age	SQLF_DBTN_DFT_REFRESH_AGE	702	char(22)
dft_sqlmathwarn	SQLF_DBTN_DFT_SQLMATHWARN	309	Sint16
discover	SQLF_DBTN_DISCOVER	308	Uint16
dlchktime	SQLF_DBTN_DLCHKTIME	9	Uint32
dyn_query_mgmt	SQLF_DBTN_DYN_QUERY_MGMT	604	Uint16
failarchpath	SQLF_DBTN_FAILARCHPATH	826	char(243)
groupheap_ratio	SQLF_DBTN_GROUPHEAP_RATIO	801	Uint16
hadr_local_host	SQLF_DBTN_HADR_LOCAL_HOST	811	char(256)
hadr_local_svc	SQLF_DBTN_HADR_LOCAL_SVC	812	char(41)
hadr_remote_host	SQLF_DBTN_HADR_REMOTE_HOST	813	char(256)
hadr_remote_inst	SQLF_DBTN_HADR_REMOTE_INST	815	char(9)
hadr_remote_svc	SQLF_DBTN_HADR_REMOTE_SVC	814	char(41)
hadr_syncmode	SQLF_DBTN_HADR_SYNCMODE	817	Uint32
hadr_timeout	SQLF_DBTN_HADR_TIMEOUT	816	Sint32
indexrec	SQLF_DBTN_INDEXREC	30	Uint16
locklist	SQLF_DBTN_LOCK_LIST	704	Uint64
locktimeout	SQLF_DBTN_LOCKTIMEOUT	34	Sint16
logarchmeth1	SQLF_DBTN_LOGARCHMETH1	822	Uint16
logarchmeth2	SQLF_DBTN_LOGARCHMETH2	823	Uint16
logarchopt1	SQLF_DBTN_LOGARCHOPT1	824	char(243)
logarchopt2	SQLF_DBTN_LOGARCHOPT2	825	char(243)
logbufsz	SQLF_DBTN_LOGBUFSZ	33	Uint16
logfilsiz	SQLF_DBTN_LOGFIL_SIZ	92	Uint32
logindexbuild	SQLF_DBTN_LOGINDEXBUILD	818	Uint32
logprimary	SQLF_DBTN_LOGPRIMARY	16	Uint16
logretain	SQLF_DBTN_LOG_RETAIN	23	Uint16
logsecond	SQLF_DBTN_LOGSECOND	17	Uint16
max_log	SQLF_DBTN_MAX_LOG	807	Uint16
maxappls	SQLF_DBTN_MAXAPPLS	6	Uint16
maxfilop	SQLF_DBTN_MAXFILOP	3	Uint16

Tabla 58. Parámetros actualizables de configuración de base de datos (continuación)

Nombre de parámetro	Señal	Valor de señal	Tipo de datos
maxlocks	SQLF_DBTN_MAXLOCKS	15	Uint16
max_log	SQLF_DBTN_MAX_LOG	807	Uint16
mincommit	SQLF_DBTN_MINCOMMIT	32	Uint16
mirrorlogpath	SQLF_DBTN_MIRRORLOGPATH	806	char(242)
newlogpath	SQLF_DBTN_NEWLOGPATH	20	char(242)
num_db_backups	SQLF_DBTN_NUM_DB_BACKUPS	601	Uint16
num_freqvalues	SQLF_DBTN_NUM_FREQVALUES	36	Uint16
num_iocleaners	SQLF_DBTN_NUM_IOCLEANERS	37	Uint16
num_ioservers	SQLF_DBTN_NUM_IOSERVERS	39	Uint16
num_log_span	SQLF_DBTN_NUM_LOG_SPAN	808	Uint16
num_quantiles	SQLF_DBTN_NUM_QUANTILES	48	Uint16
numarchretry	SQLF_DBTN_NUMARCHRETRY	827	Uint16
overflowlogpath	SQLF_DBTN_OVERFLOWLOGPATH	805	char(242)
pckcachesz	SQLF_DBTN_PCKCACHE_SZ	505	Uint32
rec_his_retentn	SQLF_DBTN_REC_HIS_RETENTN	43	Sint16
self_tuning_mem	SQLF_DBTN_SELF_TUNING_MEM	848	Uint16
seqdetect	SQLF_DBTN_SEQDETECT	41	Uint16
sheapthres_shr	SQLF_DBTN_SHEAPTHRES_SHR	802	Uint32
softmax	SQLF_DBTN_SOFTMAX	5	Uint16
sortheap	SQLF_DBTN_SORT_HEAP	52	Uint32
stat_heap_sz	SQLF_DBTN_STAT_HEAP_SZ	45	Uint32
stmtheap	SQLF_DBTN_STMTHEAP	53	Uint16
trackmod	SQLF_DBTN_TRACKMOD	703	Uint16
tsm_mgmtclass	SQLF_DBTN_TSM_MGMTCLASS	307	char(30)
tsm_nodename	SQLF_DBTN_TSM_NODENAME	306	char(64)
tsm_owner	SQLF_DBTN_TSM_OWNER	305	char(64)
tsm_password	SQLF_DBTN_TSM_PASSWORD	501	char(64)
userexit	SQLF_DBTN_USER_EXIT	24	Uint16
util_heap_sz	SQLF_DBTN_UTIL_HEAP_SZ	55	Uint32
vendoropt	SQLF_DBTN_VENDOROPT	829	char(242)

Los bits de SQLF_DBTN_AUTONOMIC_SWITCHES indican los valores por omisión para un número de parámetros de configuración del mantenimiento automático. Los bits individuales que forman este parámetro compuesto son los siguientes:

```
Valor por omisión=> Bit 1 activado  (xxxx xxxx xxxx xxx1): auto_maint
          Bit 2 desactivado (xxxx xxxx xxxx xx0x): auto_db_backup
          Bit 3 activado    (xxxx xxxx xxxx x1xx): auto_tbl_maint
          Bit 4 activado    (xxxx xxxx xxxx 1xxx): auto_runstats
          Bit 5 desactivado (xxxx xxxx xxx0 xxxx): auto_stats_prof
          Bit 6 desactivado (xxxx xxxx xx0x xxxx): auto_prof_upd
          Bit 7 desactivado (xxxx xxxx x0xx xxxx): auto_reorg
          Bit 8 desactivado (xxxx xxxx 0xxx xxxx): auto_storage
```

```
          Bit 9 desactivado (xxxx xxx0 xxxx xxxx): auto_stmt_stats
                      0    0    0    D

Máximo =>Bit 1 activado   (xxxx xxxx xxxx xxx1): auto_maint
          Bit 2 activado    (xxxx xxxx xxxx xx1x): auto_db_backup
          Bit 3 activado    (xxxx xxxx xxxx x1xx): auto_tbl_maint
          Bit 4 activado    (xxxx xxxx xxxx 1xxx): auto_runstats
          Bit 5 desactivado (xxxx xxxx xxx1 xxxx): auto_stats_prof
          Bit 6 desactivado (xxxx xxxx xx1x xxxx): auto_prof_upd
          Bit 7 desactivado (xxxx xxxx x1xx xxxx): auto_reorg
          Bit 8 desactivado (xxxx xxxx 1xxx xxxx): auto_storage
          Bit 9 desactivado (xxxx xxx1 xxxx xxxx): auto_stmt_stats
                      0    1    F    F
```

Los valores válidos para indexrec (definidos en sqlutil.h) son:

- SQLF_INX_REC_SYSTEM (0)
- SQLF_INX_REC_REFERENCE (1)
- SQLF_INX_REC_RESTART (2)

Los valores válidos para logretain (definidos en sqlutil.h) son:

- SQLF_LOGRETAIN_NO (0)
- SQLF_LOGRETAIN_RECOVERY (1)
- SQLF_LOGRETAIN_CAPTURE (2)

Tabla 59. Parámetros no actualizables de configuración de base de datos

Nombre de parámetro	Señal	Valor de señal	Tipo de datos
backup_pending	SQLF_DBTN_BACKUP_PENDING	112	Uint16
codepage	SQLF_DBTN_CODEPAGE	101	Uint16
codeset	SQLF_DBTN_CODESET	120	char(9) (ver nota 1 más abajo)
collate_info	SQLF_DBTN_COLLATE_INFO	44	char(260)
country/region	SQLF_DBTN_COUNTRY	100	Uint16
database_consistent	SQLF_DBTN_CONSISTENT	111	Uint16
database_level	SQLF_DBTN_DATABASE_LEVEL	124	Uint16
log_retain_status	SQLF_DBTN_LOG_RETAIN_STATUS	114	Uint16
loghead	SQLF_DBTN_LOGHEAD	105	char(12)
logpath	SQLF_DBTN_LOGPATH	103	char(242)
multipage_alloc	SQLF_DBTN_MULTIPAGE_ALLOC	506	Uint16
numsegs	SQLF_DBTN_NUMSEGS	122	Uint16
release	SQLF_DBTN_RELEASE	102	Uint16
restore_pending	SQLF_DBTN_RESTORE_PENDING	503	Uint16
rollfwd_pending	SQLF_DBTN_ROLLFWD_PENDING	113	Uint16
territory	SQLF_DBTN_TERRITORY	121	char(5) (ver nota 2 más abajo)
user_exit_status	SQLF_DBTN_USER_EXIT_STATUS	115	Uint16

Nota:

1. char(17) en los sistemas operativos HP-UX, Solaris y Linux.
2. char(33) en los sistemas operativos HP-UX, Solaris y Linux.

Las entradas válidas para el elemento de señal SQLFUPD se indican en la lista siguiente:

Tabla 60. Parámetros actualizables de configuración del gestor de bases de datos

Nombre de parámetro	Señal	Valor de señal	Tipo de datos
agent_stack_sz	SQLF_KTN_AGENT_STACK_SZ	61	Uint16
agentpri	SQLF_KTN_AGENTPRI	26	Sint16
aslheapsz	SQLF_KTN_ASLHEAPSZ	15	Uint32
audit_buf_sz	SQLF_KTN_AUDIT_BUF_SZ	312	Sint32
authentication	SQLF_KTN_AUTHENTICATION	78	Uint16
catalog_noauth	SQLF_KTN_CATALOG_NOAUTH	314	Uint16
clnt_krb_plugin	SQLF_KTN_CLNT_KRB_PLUGIN	812	char(33)
clnt_pw_plugin	SQLF_KTN_CLNT_PW_PLUGIN	811	char(33)
comm_bandwidth	SQLF_KTN_COMM_BANDWIDTH	307	float
conn_elapse	SQLF_KTN_CONN_ELAPSE	508	Uint16
cpuspeed	SQLF_KTN_CPUSPEED	42	float
dft_account_str	SQLF_KTN_DFT_ACCOUNT_STR	28	char(25)
dft_monswitches	SQLF_KTN_DFT_MONSWITCHES	29	Uint16
dft_mon_bufpool	SQLF_KTN_DFT_MON_BUFPOOL	33	Uint16
dft_mon_lock	SQLF_KTN_DFT_MON_LOCK	34	Uint16
dft_mon_sort	SQLF_KTN_DFT_MON_SORT	35	Uint16
dft_mon_stmt	SQLF_KTN_DFT_MON_STMT	31	Uint16
dft_mon_table	SQLF_KTN_DFT_MON_TABLE	32	Uint16
dft_mon_timestamp	SQLF_KTN_DFT_MON_ TIMESTAMP	36	Uint16
dft_mon_uow	SQLF_KTN_DFT_MON_UOW	30	Uint16
dftdbpath	SQLF_KTN_DFTDBPATH	27	char(215)
diaglevel	SQLF_KTN_DIAGLEVEL	64	Uint16
diagpath	SQLF_KTN_DIAGPATH	65	char(215)
dir_cache	SQLF_KTN_DIR_CACHE	40	Uint16
discover	SQLF_KTN_DISCOVER	304	Uint16
discover_inst	SQLF_KTN_DISCOVER_INST	308	Uint16
fcm_num_buffers	SQLF_KTN_FCM_NUM_BUFFERS	503	Uint32
fcm_num_channels	SQLF_KTN_FCM_NUM_CHANNELS	902	Uint32
fed_noauth	SQLF_KTN_FED_NOAUTH	806	Uint16
federated	SQLF_KTN_FEDERATED	604	Sint16
federated_async	SQLF_KTN_FEDERATED_ASYNC	849	Sint32
fenced_pool	SQLF_KTN_FENCED_POOL	80	Sint32
group_plugin	SQLF_KTN_GROUP_PLUGIN	810	char(33)
health_mon	SQLF_KTN_HEALTH_MON	804	Uint16
indexrec	SQLF_KTN_INDEXREC	20	Uint16

Tabla 60. Parámetros actualizables de configuración del gestor de bases de datos (continuación)

Nombre de parámetro	Señal	Valor de señal	Tipo de datos
instance_memory	SQLF_KTN_INSTANCE_MEMORY	803	Uint64
intra_parallel	SQLF_KTN_INTRA_PARALLEL	306	Sint16
java_heap_sz	SQLF_KTN_JAVA_HEAP_SZ	310	Sint32
jdk_path	SQLF_KTN_JDK_PATH	311	char(255)
keepfenced	SQLF_KTN_KEEPFENCED	81	Uint16
local_gssplugin	SQLF_KTN_LOCAL_GSSPLUGIN	816	char(33)
max_connections	SQLF_DBTN_MAX_CONNECTIONS	802	Sint32
max_connretries	SQLF_KTN_MAX_CONNRETRIES	509	Uint16
max_coordagents	SQLF_KTN_MAX_COORDAGENTS	501	Sint32
max_querydegree	SQLF_KTN_MAX_QUERYDEGREE	303	Sint32
max_time_diff	SQLF_KTN_MAX_TIME_DIFF	510	Uint16
mon_heap_sz	SQLF_KTN_MON_HEAP_SZ	79	Uint16
notifylevel	SQLF_KTN_NOTIFYLEVEL	605	Sint16
num_initagents	SQLF_KTN_NUM_INITAGENTS	500	Uint32
num_initfenced	SQLF_KTN_NUM_INITFENCED	601	Sint32
num_poolagents	SQLF_KTN_NUM_POOLAGENTS	502	Sint32
numdb	SQLF_KTN_NUMDB	6	Uint16
query_heap_sz	SQLF_KTN_QUERY_HEAP_SZ	49	Sint32
resync_interval	SQLF_KTN_RESYNC_INTERVAL	68	Uint16
rqrioblk	SQLF_KTN_RQRIOBLK	1	Uint16
sheapthres	SQLF_KTN_SHEAPTHRES	21	Uint32
spm_log_file_sz	SQLF_KTN_SPM_LOG_FILE_SZ	90	Sint32
spm_log_path	SQLF_KTN_SPM_LOG_PATH	313	char(226)
spm_max_resync	SQLF_KTN_SPM_MAX_RESYNC	91	Sint32
spm_name	SQLF_KTN_SPM_NAME	92	char(8)
srvcon_auth	SQLF_KTN_SRVCON_AUTH	815	Uint16
srvcon_gssplugin_list	SQLF_KTN_SRVCON_GSSPLUGIN_ LIST	814	char(256)
srv_plugin_mode	SQLF_KTN_SRV_PLUGIN_MODE	809	Uint16
srvcon_pw_plugin	SQLF_KTN_SRVCON_PW_PLUGIN	813	char(33)
start_stop_time	SQLF_KTN_START_STOP_TIME	511	Uint16
nombservicio	SQLF_KTN_SVCENAME	24	char(14)
sysadm_group	SQLF_KTN_SYSADM_GROUP	39	char(16)
sysctrl_group	SQLF_KTN_SYSCTRL_GROUP	63	char(16)
sysmaint_group	SQLF_KTN_SYSMAINT_GROUP	62	char(16)
sysmon_group	SQLF_KTN_SYSMON_GROUP	808	char(30)
tm_database	SQLF_KTN_TM_DATABASE	67	char(8)
tp_mon_name	SQLF_KTN_TP_MON_NAME	66	char(19)
trust_allclnts	SQLF_KTN_TRUST_ALLCLNTS	301	Uint16

Tabla 60. Parámetros actualizables de configuración del gestor de bases de datos (continuación)

Nombre de parámetro	Señal	Valor de señal	Tipo de datos
trust_clntauth	SQLF_KTN_TRUST_CLNTAUTH	302	Uint16
util_impact_lim	SQLF_KTN_UTIL_IMPACT_LIM	807	Uint32

Nota: Los parámetros de configuración `maxagents` y `maxcagents` han quedado obsoletos. En un futuro release, es posible que estos parámetros de configuración se eliminen por completo.

Los valores válidos para authentication (definidos en sqlenv.h) son:
- SQL_AUTHENTICATION_SERVER (0)
- SQL_AUTHENTICATION_CLIENT (1)
- SQL_AUTHENTICATION_DCS (2)
- SQL_AUTHENTICATION_DCE (3)
- SQL_AUTHENTICATION_SVR_ENCRYPT (4)
- SQL_AUTHENTICATION_DCS_ENCRYPT (5)
- SQL_AUTHENTICATION_DCE_SVR_ENC (6)
- SQL_AUTHENTICATION_KERBEROS (7)
- SQL_AUTHENTICATION_KRB_SVR_ENC (8)
- SQL_AUTHENTICATION_GSSPLUGIN (9)
- SQL_AUTHENTICATION_GSS_SVR_ENC (10)
- SQL_AUTHENTICATION_DATAENC (11)
- SQL_AUTHENTICATION_DATAENC_CMP (12)
- SQL_AUTHENTICATION_NOT_SPEC (255)

SQLF_KTN_DFT_MONSWITCHES es un parámetro Uint16, cuyos bits indican los valores del conmutador de supervisor por omisión. Esto permite realizar la especificación de un número de parámetros al mismo tiempo. Los bits individuales que forman este parámetro compuesto son los siguientes:
- Bit 1 (xxxx xxx1): dft_mon_uow
- Bit 2 (xxxx xx1x): dft_mon_stmt
- Bit 3 (xxxx x1xx): dft_mon_table
- Bit 4 (xxxx 1xxx): dft_mon_buffpool
- Bit 5 (xxx1 xxxx): dft_mon_lock
- Bit 6 (xx1x xxxx): dft_mon_sort
- Bit 7 (x1xx xxxx): dft_mon_timestamp

Los valores válidos para discover (definido en sqlutil.h) son:
- SQLF_DSCVR_KNOWN (1)
- SQLF_DSCVR_SEARCH (2)

Los valores válidos para indexrec (definidos en sqlutil.h) son:
- SQLF_INX_REC_SYSTEM (0)
- SQLF_INX_REC_REFERENCE (1)
- SQLF_INX_REC_RESTART (2)

Los valores válidos para trust_allclnts (definido en sqlutil.h) son:

- SQLF_TRUST_ALLCLNTS_NO (0)
- SQLF_TRUST_ALLCLNTS_YES (1)
- SQLF_TRUST_ALLCLNTS_DRDAONLY (2)

Tabla 61. Parámetros no actualizables de configuración del gestor de bases de datos

Nombre de parámetro	Señal	Valor de señal	Tipo de datos
nodetype	SQLF_KTN_NODETYPE	100	Uint16
release	SQLF_KTN_RELEASE	101	Uint16

Los valores válidos para nodetype (definido en sqlutil.h) son:

- SQLF_NT_STANDALONE (0)
- SQLF_NT_SERVER (1)
- SQLF_NT_REQUESTOR (2)
- SQLF_NT_STAND_REQ (3)
- SQLF_NT_MPP (4)
- SQLF_NT_SATELLITE (5)

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqlfupd
{
   unsigned short  token;
   char *ptrvalue;
};
```

Estructura en COBOL

```
* Archivo: sqlutil.cbl
01 SQL-FUPD.
    05 SQL-TOKEN             PIC 9(4) COMP-5.
    05 FILLER                PIC X(2).
    05 SQL-VALUE-PTR         USAGE IS POINTER.
*
```

sqllob

Esta estructura se utiliza para representar un tipo de datos LOB en un lenguaje de programación de sistema principal.

Tabla 62. Campos de la estructura sqllob

Nombre de campo	Tipo de datos	Descripción
length	sqluint32	Longitud, en bytes, del parámetro data.
data	char(1)	Datos que se pasan.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqllob
{
        sqluint32       length;
        char data[1];
};
```

sqlma

La estructura del área del supervisor de SQL (SQLMA) se utiliza para enviar peticiones de instantánea de supervisor de base de datos al gestor de bases de datos. También se utiliza para calcular el tamaño (en bytes) de los datos de instantánea devueltos.

Tabla 63. Campos de la estructura SQLMA

Nombre de campo	Tipo de datos	Descripción
OBJ_NUM	INTEGER	Número de objetos que deben supervisarse.
OBJ_VAR	Matriz	Matriz de estructuras de datos sqlm_obj_struct que contiene las descripciones de los objetos que deben supervisarse. La longitud de la matriz está determinada por OBJ_NUM.

Tabla 64. Campos de la estructura SQLM-OBJ-STRUCT

Nombre de campo	Tipo de datos	Descripción
AGENT_ID	INTEGER	Descriptor de contexto de aplicación de la aplicación que debe supervisarse. Sólo se especifica si OBJ_TYPE requiere un agent_id (descriptor de contexto de aplicación). Para recuperar una instantánea de estado que incluya información completa, especifique SQLM_HMON_OPT_COLL_FULL en este campo.
OBJ_TYPE	INTEGER	El tipo de objeto que deben supervisarse.
OBJECT	CHAR(128)	Nombre del objeto que debe supervisarse. Sólo se especifica si OBJ_TYPE requiere un nombre, por ejemplo appl_id, o un alias de base de datos.

Los valores válidos para OBJ_TYPE (definidos en el archivo de cabecera sqlmon del directorio de inclusión) son:

SQLMA_DB2
Información referente a la instancia.

SQLMA_DBASE
Información referente a una base de datos determinada. Si utiliza el valor SQLMA_DBASE, debe proporcionar el nombre de la base de datos en el parámetro OBJECT de la estructura sqlm_obj_struct.

SQLMA_APPL
Información referente a una aplicación que corresponde al ID de aplicación proporcionado. Si utiliza el valor SQLMA_APPL, debe proporcionar un ID de aplicación en el parámetro OBJECT de la estructura sqlm_obj_struct.

SQLMA_AGENT_ID
Información referente a una aplicación que corresponde al ID de agente proporcionado. Si utiliza el valor SQLMA_AGENT_ID, debe proporcionar un ID de agente en el parámetro AGENT_ID de la estructura sqlm_obj_struct.

SQLMA_DBASE_TABLES
Información sobre tablas para una base de datos determinada. Si utiliza el

valor SQLMA_DBASE_TABLES, debe proporcionar el nombre de la base de datos en el parámetro OBJECT de la estructura sqlm_obj_struct.

SQLMA_DBASE_APPLS
Información sobre todas las aplicaciones conectadas a una base de datos determinada. Si utiliza el valor SQLMA_DBASE_APPLS, debe proporcionar el nombre de la base de datos en el parámetro OBJECT de la estructura sqlm_obj_struct.

SQLMA_DBASE_APPLINFO
Información de resumen sobre aplicaciones conectadas a una base de datos determinada. Si utiliza el valor SQLMA_DBASE_APPLINFO, debe proporcionar el nombre de la base de datos en el parámetro OBJECT de la estructura sqlm_obj_struct.

SQLMA_DBASE_LOCKS
Lista de los bloqueos mantenidos en una base de datos determinada. Si utiliza el valor SQLMA_DBASE_LOCKS, debe proporcionar el nombre de la base de datos en el parámetro OBJECT de la estructura sqlm_obj_struct.

SQLMA_APPL_LOCKS
Lista de los bloqueos mantenidos por una aplicación que corresponde al ID de aplicación proporcionado. Si utiliza el valor SQLMA_APPL_LOCKS, debe proporcionar un ID de aplicación en el parámetro OBJECT de la estructura sqlm_obj_struct.

SQLMA_APPL_LOCKS_AGENT_ID
Lista de los bloqueos mantenidos por una aplicación que corresponde al ID de agente proporcionado. Si utiliza el valor SQLMA_APPL_LOCKS_AGENT_ID, debe proporcionar un ID de agente en el parámetro AGENT_ID de la estructura sqlm_obj_struct.

SQLMA_DBASE_ALL
Información sobre todas las bases de datos activas existentes en la instancia.

SQLMA_APPL_ALL
Información sobre aplicaciones correspondiente a todas las conexiones de base de datos existentes en la instancia.

SQLMA_APPLINFO_ALL
Información de resumen sobre aplicaciones correspondiente a todas las conexiones establecidas con la instancia.

SQLMA_DCS_APPLINFO_ALL
Lista de las conexiones DCS (servicios de conexión de base de datos) establecidas con la instancia.

SQLMA_DYNAMIC_SQL
Información sobre sentencias de SQL dinámico para una base de datos determinada. Si utiliza el valor SQLMA_DYNAMIC_SQL, debe proporcionar el nombre de la base de datos en el parámetro OBJECT de la estructura sqlm_obj_struct.

SQLMA_DCS_DBASE
Información referente a una base de datos de DCS (Database Connection Services) determinada. Si utiliza el valor SQLMA_DCS_DBASE, debe proporcionar el nombre de la base de datos en el parámetro OBJECT de la estructura sqlm_obj_struct.

SQLMA_DCS_DBASE_ALL
Información sobre todas las bases de datos DCS (servicios de conexión de base de datos) activas.

SQLMA_DCS_APPL_ALL
Información sobre aplicaciones DCS (servicios de conexión de base de datos) para todas las conexiones.

SQLMA_DCS_APPL
Información sobre aplicaciones de DCS (Database Connection Services) para una aplicación que coincide con el IP de aplicación proporcionado. Si utiliza el valor SQLMA_DCS_APPL, debe proporcionar un ID de aplicación en el parámetro OBJECT de la estructura sqlm_obj_struct.

SQLMA_DCS_APPL_HANDLE
Información sobre aplicaciones DCS (servicios de conexión de base de datos) para una aplicación que corresponde al ID de agente proporcionado. Si utiliza el valor SQLMA_DCS_APPL_HANDLE, debe proporcionar un ID de agente en el parámetro agent_id de la estructura sqlm_obj_struct.

SQLMA_DCS_DBASE_APPLS
Información sobre aplicaciones DCS (servicios de conexión de base de datos) para todas las conexiones activas con una base de datos determinada. Si utiliza el valor SQLMA_DCS_DBASE_APPLS, debe proporcionar el nombre de la base de datos en el parámetro de objeto de la estructura sqlm_obj_struct.

SQLMA_DBASE_TABLESPACES
Información sobre espacios de tablas para una base de datos determinada. Si utiliza el valor SQLMA_DBASE_TABLESPACES, debe proporcionar el nombre de la base de datos en el parámetro OBJECT de la estructura sqlm_obj_struct.

SQLMA_DBASE_BUFFERPOOLS
Información sobre agrupaciones de almacenamientos intermedios para una base de datos determinada. Si utiliza el valor SQLMA_DBASE_BUFFERPOOLS, debe proporcionar el nombre de la base de datos en el parámetro OBJECT de la estructura sqlm_obj_struct.

SQLMA_BUFFERPOOLS_ALL
Información correspondiente a todas las agrupaciones de almacenamientos intermedios.

SQLMA_DBASE_REMOTE
Información sobre acceso remoto para una base de datos federada determinada. Si utiliza el valor SQLMA_DBASE_REMOTE, debe proporcionar el nombre de la base de datos en el parámetro OBJECT de la estructura sqlm_obj_struct.

SQLMA_DBASE_REMOTE_ALL
Información sobre acceso remoto para todas las bases de datos federadas.

SQLMA_DBASE_APPLS_REMOTE
Información sobre acceso remoto para una aplicación conectada a una base de datos federada determinada. Si utiliza el valor SQLMA_DBASE_APPLS_REMOTE, debe proporcionar el nombre de la base de datos en el parámetro OBJECT de la estructura sqlm_obj_struct.

SQLMA_APPL_REMOTE_ALL
Información sobre acceso remoto para todas las aplicaciones.

Sintaxis de la API y de las estructuras de datos

```
typedef struct sqlma
{
   sqluint32 obj_num;
   sqlm_obj_struct obj_var[1];
}sqlma;

typedef struct sqlm_obj_struct
{
   sqluint32 agent_id;
   sqluint32 obj_type;
   _SQLOLDCHAR object[SQLM_OBJECT_SZ];
}sqlm_obj_struct;
```

Estructura en COBOL

```
* Archivo: sqlmonct.cbl
01 SQLMA.
    05 OBJ-NUM                PIC 9(9) COMP-5.
    05 OBJ-VAR OCCURS 0 TO 100 TIMES DEPENDING ON OBJ-NUM.
        10 AGENT-ID            PIC 9(9) COMP-5.
        10 OBJ-TYPE            PIC 9(9) COMP-5.
        10 OBJECT              PIC X(128).
*
```

sqlopt

Esta estructura se utiliza para pasar opciones de vinculación a la API sqlabndx, opciones de precompilación a la API sqlaprep y opciones de revinculación a la API sqlarbnd.

Tabla 65. Campos de la estructura SQLOPT

Nombre de campo	Tipo de datos	Descripción
HEADER	Estructura	Una estructura sqloptheader.
OPTION	Matriz	Matriz de estructuras sqloptions. El número de elementos de esta matriz está determinado por el valor del campo asignado de la cabecera.

Tabla 66. Campos de la estructura SQLOPTHEADER

Nombre de campo	Tipo de datos	Descripción
ALLOCATED	INTEGER	Número de elementos de la matriz de opciones de la estructura sqlopt.
USED	INTEGER	Número de elementos de la matriz de opciones de la estructura sqlopt utilizada realmente. Es el número de pares de opciones (TYPE y VAL) suministrados.

Tabla 67. Campos de la estructura SQLOPTIONS

Nombre de campo	Tipo de datos	Descripción
TYPE VAL	INTEGER INTEGER	Tipo de opción de vinculación/precompilación/ revinculación. Valor de opción de vinculación/precompilación/ revinculación.

Nota: Los campos TYPE y VAL se repiten para cada opción de vinculación, precompilación o revinculación especificada.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqlopt
{
   SQL_STRUCTURE sqloptheader header;
   SQL_STRUCTURE sqloptions   option[1];
};

SQL_STRUCTURE sqloptheader
{
   sqluint32 allocated;
   sqluint32 used;
};

SQL_STRUCTURE sqloptions
{
   sqluint32 type;
   sqluintptr val;
};
```

Estructura en COBOL

```
* Archivo: sql.cbl
01 SQLOPT.
    05 SQLOPTHEADER.
        10 ALLOCATED   PIC 9(9) COMP-5.
        10 USED        PIC 9(9) COMP-5.
    05 SQLOPTIONS OCCURS 1 TO 50 DEPENDING ON ALLOCATED.
        10 SQLOPT-TYPE        PIC 9(9) COMP-5.
        10 SQLOPT-VAL         PIC 9(9) COMP-5.
        10 SQLOPT-VAL-PTR     REDEFINES SQLOPT-VAL
*
```

SQLU_LSN

Esta unión contiene la definición del número de secuencia de anotaciones y es utilizada por la API db2ReadLog. El número de secuencia de anotaciones (LSN) representa una dirección de byte relativa dentro del archivo de anotaciones de la base de datos. Todos los registros de anotaciones se identifican mediante este número. Un LSN representa el desplazamiento en bytes del registro de anotaciones con respecto al inicio del archivo de anotaciones de la base de datos.

Tabla 68. Campos de la unión SQLU-LSN

Nombre de campo	Tipo de datos	Descripción
lsnChar	Matriz de UNSIGNED CHAR	Especifica el número de secuencia de anotaciones de matriz de caracteres de 6 miembros.
lsnWord	Matriz de UNSIGNED SHORT	Especifica el número de secuencia de anotaciones de matriz corta de 3 miembros.

Sintaxis de la API y de las estructuras de datos

```
typedef union SQLU_LSN
{
   unsigned char   lsnChar[6];
   unsigned short  lsnWord[3];
} SQLU_LSN;
```

sqlu_media_list

Esta estructura se utiliza para pasar información a la API db2Load.

Tabla 69. Campos de la estructura SQLU-MEDIA-LIST

Nombre de campo	Tipo de datos	Descripción
MEDIA_TYPE	CHAR(1)	Carácter que indica el tipo de medio.
SESSIONS	INTEGER	Indica el número de elementos de la matriz a la que señala el campo destino de esta estructura.
TARGET	Unión	Este campo es un puntero a uno de cuatro tipos de estructuras. El tipo de estructura al que se señala está determinado por el valor del campo media_type. Para obtener más información acerca del valor que debe especificarse en este campo, consulte la API correspondiente.
FILLER	CHAR(3)	Rellenador utilizado para la alineación correcta de la estructura de datos en la memoria.

Tabla 70. Campos de la estructura SQLU-MEDIA-LIST-TARGETS

Nombre de campo	Tipo de datos	Descripción
MEDIA	Puntero	Puntero a la estructura sqlu_media_entry.
VENDOR	Puntero	Puntero a una estructura sqlu_vendor.

Tabla 70. Campos de la estructura SQLU-MEDIA-LIST-TARGETS (continuación)

Nombre de campo	Tipo de datos	Descripción
LOCATION	Puntero	Puntero a una estructura sqlu_location_entry.
PSTATEMENT	Puntero	Puntero a una estructura sqlu_statement_entry.
PREMOTEFETCH	Puntero	Puntero a una estructura sqlu_remotefetch_entry.

Tabla 71. Campos de la estructura SQLU-MEDIA-ENTRY

Nombre de campo	Tipo de datos	Descripción
RESERVE_LEN MEDIA_ENTRY	INTEGER CHAR(215)	Longitud del campo media_entry. Para lenguajes que no sean C. Vía de acceso para una imagen de copia de seguridad utilizada por los programas de utilidad de copia de seguridad y restauración.

Tabla 72. Campos de la estructura SQLU-VENDOR

Nombre de campo	Tipo de datos	Descripción
RESERVE_LEN1	INTEGER	Longitud del campo shr_lib. Para lenguajes que no sean C.
SHR_LIB	CHAR(255)	Nombre de la biblioteca compartida suministrada por los proveedores para almacenar o recuperar datos.
RESERVE_LEN2	INTEGER	Longitud del campo de nombre de archivo (filename). Para lenguajes que no sean C.
FILENAME	CHAR(255)	Nombre de archivo que identifica la fuente de entrada de carga al utilizar una biblioteca compartida.

Tabla 73. Campos de la estructura SQLU-LOCATION-ENTRY

Nombre de campo	Tipo de datos	Descripción
RESERVE_LEN	INTEGER	Longitud del campo location_entry. Para lenguajes que no sean C.
LOCATION_ENTRY	CHAR(256)	Nombre de los archivos de datos de entrada para el programa de utilidad de carga.

Tabla 74. Campos de la estructura SQLU-STATEMENT-ENTRY

Nombre de campo	Tipo de datos	Descripción
LENGTH	INTEGER	Longitud del campo de datos.
PDATA	Puntero	Puntero a la consulta SQL.

Tabla 75. Campos de la estructura SQLU-REMOTEFETCH-ENTRY

Nombre de campo	Tipo de datos	Descripción
pDatabaseName	Puntero	Nombre de la base de datos fuente.
iDatabaseNameLen	INTEGER	Longitud del nombre de la base de datos fuente.
pUserID	Puntero	Puntero al ID de usuario (UserID).
iUserIDLen	INTEGER	Longitud del ID de usuario (UserID).
pPassword	Puntero	Puntero a la contraseña (Password).
iPasswordLen	INTEGER	Longitud de la contraseña (Password).
pTableSchema	Puntero	Puntero al esquema de tabla fuente.
iTableSchemaLen	INTEGER	Longitud del esquema.
pTableName	Puntero	Puntero al nombre de tabla fuente.
iTableNameLen	INTEGER	Longitud del nombre de tabla fuente.
pStatement	Puntero	Puntero al nombre de sentencia.
iStatementLen	INTEGER	Longitud de sentencia.
pIsolationLevel	Puntero	Puntero al nivel de aislamiento (por omisión, CS).

Los valores válidos para MEDIA_TYPE (definido en sqlutil) son:

SQLU_LOCAL_MEDIA
Dispositivos locales (cintas, discos o disquetes)

SQLU_SERVER_LOCATION
Dispositivos de servidor (cintas, discos o disquetes: sólo carga) Sólo puede especificarse para el parámetro piSourceList.

SQLU_CLIENT_LOCATION
Dispositivos de cliente (archivos o conexiones con nombre). Sólo puede especificarse para el parámetro piSourceList o el parámetro piLobFileList.

SQLU_SQL_STMT
Consulta SQL (sólo carga). Sólo puede especificarse para el parámetro piSourceList.

SQLU_TSM_MEDIA
TSM

SQLU_XBSA_MEDIA
XBSA

SQLU_OTHER_MEDIA
Biblioteca de proveedor

SQLU_REMOTEFETCH
Medio de extracción remota (sólo carga). Sólo puede especificarse para el parámetro piSourceList.

SQLU_DISK_MEDIA
Disco (sólo para las API de proveedor)

SQLU_DISKETTE_MEDIA
Disquete (sólo para las API de proveedor)

SQLU_NULL_MEDIA
Nulo (generado internamente por la base de datos DB2)

SQLU_TAPE_MEDIA
Cinta (sólo para las API de proveedor)

SQLU_PIPE_MEDIA
Conexión con nombre (sólo para las API de proveedor)

Sintaxis de la API y de las estructuras de datos

```
typedef SQL_STRUCTURE sqlu_media_list
{
   char media_type;
   char filler[3];
   sqlint32 sessions;
   union sqlu_media_list_targets target;
} sqlu_media_list;

union sqlu_media_list_targets
{
   struct sqlu_media_entry *media;
   struct sqlu_vendor *vendor;
   struct sqlu_location_entry *location;
   struct sqlu_statement_entry *pStatement;
   struct sqlu_remotefetch_entry *pRemoteFetch;
};

typedef SQL_STRUCTURE sqlu_media_entry
{
   sqluint32 reserve_len;
   char media_entry[SQLU_DB_DIR_LEN+1];
} sqlu_media_entry;

typedef SQL_STRUCTURE sqlu_vendor
{
   sqluint32 reserve_len1;
   char shr_lib[SQLU_SHR_LIB_LEN+1];
   sqluint32 reserve_len2;
   char filename[SQLU_SHR_LIB_LEN+1];
} sqlu_vendor;

typedef SQL_STRUCTURE sqlu_location_entry
{
   sqluint32 reserve_len;
   char location_entry[SQLU_MEDIA_LOCATION_LEN+1];
} sqlu_location_entry;
```

```
typedef SQL_STRUCTURE sqlu_statement_entry
{
   sqluint32        length;
   char *pEntry;
} sqlu_statement_entry;

typedef SQL_STRUCTURE sqlu_remotefetch_entry
{
   char *pDatabaseName;
   sqluint32 iDatabaseNameLen;
   char *pUserID;
   sqluint32 iUserIDLen;
   char *pPassword;
   sqluint32 iPasswordLen;
   char *pTableSchema;
   sqluint32 iTableSchemaLen;
   char *pTableName;
   sqluint32 iTableNameLen;
   char *pStatement;
   sqluint32 iStatementLen;
   sqlint32 *pIsolationLevel;
   sqluint32 *piEnableParallelism;
} sqlu_remotefetch_entry;
```

Estructura en COBOL

```
* Archivo: sqlutil.cbl
01 SQLU-MEDIA-LIST.
    05 SQL-MEDIA-TYPE         PIC X.
    05 SQL-FILLER             PIC X(3).
    05 SQL-SESSIONS           PIC S9(9) COMP-5.
    05 SQL-TARGET.
        10 SQL-MEDIA          USAGE IS POINTER.
        10 SQL-VENDOR         REDEFINES SQL-MEDIA
        10 SQL-LOCATION       REDEFINES SQL-MEDIA
        10 SQL-STATEMENT      REDEFINES SQL-MEDIA
        10 FILLER             REDEFINES SQL-MEDIA
*


* Archivo: sqlutil.cbl
01 SQLU-MEDIA-ENTRY.
    05 SQL-MEDENT-LEN         PIC 9(9) COMP-5.
    05 SQL-MEDIA-ENTRY        PIC X(215).
    05 FILLER                 PIC X.
*


* Archivo: sqlutil.cbl
01 SQLU-VENDOR.
    05 SQL-SHRLIB-LEN         PIC 9(9) COMP-5.
    05 SQL-SHR-LIB            PIC X(255).
    05 FILLER                 PIC X.
    05 SQL-FILENAME-LEN       PIC 9(9) COMP-5.
    05 SQL-FILENAME           PIC X(255).
    05 FILLER                 PIC X.
*


* Archivo: sqlutil.cbl
01 SQLU-LOCATION-ENTRY.
    05 SQL-LOCATION-LEN       PIC 9(9) COMP-5.
    05 SQL-LOCATION-ENTRY     PIC X(255).
    05 FILLER                 PIC X.
*
```

```
* Archivo: sqlutil.cbl
01 SQLU-STATEMENT-ENTRY.
    05 SQL-STATEMENT-LEN       PIC 9(9) COMP-5.
    05 SQL-STATEMENT-ENTRY     USAGE IS POINTER.
*
```

SQLU_RLOG_INFO

Esta estructura contiene información sobre el estado de las llamadas realizadas a la API db2ReadLog y al archivo de anotaciones de la base de datos.

Tabla 76. Campos de la estructura SQLU-RLOG-INFO

Nombre de campo	Tipo de datos	Descripción
initialLSN	SQLU_LSN	Especifica el valor LSN del primer registro de anotaciones que se escribe después de emitir la primera sentencia CONNECT de base de datos. Para obtener más información, consulte SQLU-LSN.
firstReadLSN	SQLU_LSN	Especifica el valor LSN del primer registro de anotaciones leído.
lastReadLSN	SQLU_LSN	Especifica el valor LSN del último registro de anotaciones leído.
curActiveLSN	SQLU_LSN	Especifica el valor LSN del archivo de anotaciones actual (activo).
logRecsWritten	sqluint32	Especifica el número de registros de anotaciones escritos en el almacenamiento intermedio.
logBytesWritten	sqluint32	Especifica el número de bytes escritos en el almacenamiento intermedio.

Sintaxis de la API y de las estructuras de datos

```
typedef SQL_STRUCTURE SQLU_RLOG_INFO
{
   SQLU_LSN initialLSN;
   SQLU_LSN firstReadLSN;
   SQLU_LSN lastReadLSN;
   SQLU_LSN curActiveLSN;
   sqluint32 logRecsWritten;
   sqluint32 logBytesWritten;
} SQLU_RLOG_INFO;
```

Esta estructura se utiliza para almacenar información sobre particionamiento, tal como el mapa de distribución y la clave de distribución de una tabla.

Tabla 77. Campos de la estructura SQLUPI

Nombre de campo	Tipo de datos	Descripción
PMAPLEN	INTEGER	Longitud del mapa de distribución en bytes. Para tablas de un solo nodo, el valor es sizeof(SQL_PDB_NODE_TYPE). Para tablas de varios nodos, el valor es SQL_PDB_MAP_SIZE * sizeof(SQL_PDB_NODE_TYPE).
PMAP	SQL_PDB_NODE_TYPE	El mapa de distribución.
SQLD	INTEGER	Número de elementos SQLPARTKEY utilizados, es decir, el número de componentes esenciales de una clave de distribución.
SQLPARTKEY	Estructura	Descripción de una columna de distribución de una clave de distribución. El número máximo de columnas de distribución es SQL_MAX_NUM_PART_KEYS.

La tabla siguiente muestra los tipos de datos SQL y longitudes para la estructura de datos SQLUPI. La columna SQLTYPE especifica el valor numérico que representa el tipo de datos de un elemento.

Tabla 78. Tipos de datos SQL y longitudes para la estructura SQLUPI

Tipo de datos	SQLTYPE (Nulos no permitidos)	SQLTYPE (Nulos permitidos)	SQLLEN	AIX
Fecha	384	385	Se pasa por alto	Sí
Hora	388	389	Se pasa por alto	Sí
Indicación de fecha y hora	392	393	Se pasa por alto	Sí
Serie de caracteres de longitud variable	448	449	Longitud de la serie	Sí
Serie de caracteres de longitud fija	452	453	Longitud de la serie	Sí
Serie de caracteres larga	456	457	Se pasa por alto	No
Serie de caracteres terminada en nulo	460	461	Longitud de la serie	Sí
Coma flotante	480	481	Se pasa por alto	Sí

Tabla 78. Tipos de datos SQL y longitudes para la estructura SQLUPI (continuación)

Tipo de datos	SQLTYPE (Nulos no permitidos)	SQLTYPE (Nulos permitidos)	SQLLEN	AIX
Decimal	484	485	Byte 1 = precisión Byte 2 = escala	Sí
Número entero grande	496	497	Se pasa por alto	Sí
Número entero pequeño	500	501	Se pasa por alto	Sí
Serie de caracteres gráficos de longitud variable	464	465	Longitud en caracteres de doble byte	Sí
Serie de caracteres gráficos de longitud fija	468	469	Longitud en caracteres de doble byte	Sí
Serie gráfica larga	472	473	Se pasa por alto	No

Descripciones de parámetros de la estructura de datos sqlpartkey

sqltype
Entrada. Tipo de datos de la clave de distribución.

sqllen Entrada. Longitud de los datos de la clave de distribución.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqlupi
{
   unsigned short  pmaplen;
   SQL_PDB_NODE_TYPE pmap[SQL_PDB_MAP_SIZE];
   unsigned short  sqld;
   struct sqlpartkey sqlpartkey[SQL_MAX_NUM_PART_KEYS];
};

SQL_STRUCTURE sqlpartkey
{
   unsigned short  sqltype;
   unsigned short  sqllen;
};
```

SQLXA_XID

Esta estructura es utilizada por las API de transacciones para identificar las transacciones XA. Las API sqlxhfrg, sqlxphcm, sqlxphrl, sqlcspqy y db2XaListIndTrans constituyen el grupo de API de transacciones. Estas API se utilizan para la gestión de transacciones dudosas.

Tabla 79. Campos de la estructura SQLXA-XID

Nombre de campo	Tipo de datos	Descripción
FORMATID	INTEGER	ID de formato XA.

Tabla 79. Campos de la estructura SQLXA-XID (continuación)

Nombre de campo	Tipo de datos	Descripción
GTRID_LENGTH	INTEGER	Longitud del ID de transacción global.
BQUAL_LENGTH	INTEGER	Longitud del identificador de rama.
DATA	CHAR[128]	GTRID, seguido por BQUAL y blancos de cola, hasta un total de 128 bytes.

Nota: El tamaño máximo para GTRID y BQUAL es 64 bytes para cada uno.

Sintaxis de la API y de las estructuras de datos

```
struct sqlxa_xid_t {
  sqlint32 formatID;
  sqlint32 gtrid_length;
  sqlint32 bqual_length;
  char data[SQLXA_XIDDATASIZE];
  } ;
typedef struct sqlxa_xid_t SQLXA_XID;
```

Apéndice A. Las API de personalización de precompilador

Las API de personalización de precompilador

Es un conjunto de API documentadas que permiten que otras herramientas de desarrollo de aplicaciones implementen soporte de precompilador para DB2 directamente dentro de sus productos. Por ejemplo, IBM COBOL en AIX utiliza esta interfaz. La información sobre el conjunto de API de servicios de precompilador está disponible en el archivo PDF, `prepapi.pdf`, en el siguiente sitio web:

`http://www.ibm.com/software/data/db2/udb/support/manualsv9.html`

Apéndice B. Registros de anotaciones de DB2

Registros de anotaciones de DB2

Este apartado describe la estructura de los registros de anotaciones de DB2 devueltos por la API db2ReadLog cuando se especifica el valor de entrada de `iFilterOption DB2READLOG_FILTER_ON`. Sólo se devuelven los registros de anotaciones propagados cuando se utiliza este valor. Sólo se documentan los registros de anotaciones propagados. Todos los demás registros de anotaciones cronológicas están destinados solamente al uso interno de IBM y por lo tanto no están documentados.

Todos los registros de anotaciones cronológicas de DB2 comienzan con una cabecera del gestor de anotaciones cronológicas. Esta cabecera contiene el tamaño total y tipo del registro de anotaciones, e información específica de la transacción. No contiene información sobre contabilidad, estadísticas, rastreos ni evaluación del rendimiento. Para obtener más información consulte “Cabecera del gestor de anotaciones” en la página 594.

Los registros de anotaciones se identifican de forma unívoca mediante un número de secuencia de anotaciones (LSN). El LSN representa la dirección de byte relativa del primer byte del registro de anotaciones, dentro del archivo de anotaciones de la base de datos. Especifica el desplazamiento del registro de anotaciones con respecto al principio del archivo de anotaciones de la base de datos.

Los registros de anotaciones escritos por una transacción individual se identifican unívocamente mediante un campo contenido en la cabecera del registro de anotaciones. El identificador de transacción exclusivo es un campo de seis bytes que se incrementa en una unidad cada vez que se inicia una nueva transacción. Todos los registros de anotaciones escritos por una transacción individual contienen el mismo identificador.

Cuando una transacción realiza una tarea de escritura para una tabla con la opción DATA CAPTURE CHANGES activada, o invoca un programa de escritura de anotaciones, la transacción se marca como propagable. Solamente las transacciones propagables tienen marcados como propagables los registros de anotaciones del gestor de transacciones.

Tabla 80. Registros de anotaciones de DB2

Tipo	Nombre de registro	Descripción
Gestor de datos	“Registro de anotaciones de inicialización de tabla” en la página 612	Creación de nueva tabla permanente.
Gestor de datos	“Registro de anotaciones de sustitución de importación (truncamiento)” en la página 614	Actividad de importar sustitución.
Gestor de datos	“Registro de anotaciones de activación no registrada inicialmente” en la página 614	La actividad de modificar tabla que incluye la cláusula ACTIVATE NOT LOGGED INITIALLY.

Tabla 80. Registros de anotaciones de DB2 (continuación)

Tipo	Nombre de registro	Descripción
Gestor de datos	"Registro de anotaciones de retrotracción de inserción" en la página 615	Retrotraer inserción de fila.
Gestor de datos	"Registro de anotaciones de reorganización de tabla" en la página 615	REORG confirmado.
Gestor de datos	"Registros de anotaciones de crear índice, descartar índice" en la página 616	Actividad de índice.
Gestor de datos	"Registros de anotaciones para crear tabla, descartar tabla, retrotraer creación de tabla, retrotraer el descarte de la tabla" en la página 616	Actividad de tabla.
Gestor de datos	"Registro de anotaciones de atributo de alterar tabla" en la página 616	Actividad de propagación, pendiente de comprobación y modalidad de edición.
Gestor de datos	"Registro de anotaciones de añadir columnas por alteración de tabla, retrotraer adición de columnas" en la página 617	Adición de columnas a tablas existentes.
Gestor de datos	"Registro de anotaciones de atributo de alterar columna" en la página 618	Actividad de columnas.
Gestor de datos	"Registro de anotaciones del atributo de deshacer alteración de columna" en la página 618	Actividad de columna.
Gestor de datos	"Registro de anotaciones de insertar registro, retrotraer supresión de registro, retrotraer actualización de registro" en la página 619	Actividad en registros de tabla.
Gestor de datos	"Registros de anotaciones de insertar registro en página vacía, suprimir registro en página vacía, retrotraer supresión de registro en página vacía y retrotraer inserción de registro en página vacía" en la página 623	Actividad de tabla agrupada en clústers de varias dimensiones (MDC).
Gestor de datos	"Registro de anotaciones de actualizar registro" en la página 624	La fila se actualiza donde no ha cambiado la ubicación de almacenamiento.
Gestor de datos	"Registro de anotaciones de renombramiento de una tabla o un esquema" en la página 624	Actividad de nombres de esquema o tabla.
Gestor de datos	"Registro de anotaciones de deshacer renombramiento de una tabla o un esquema" en la página 625	Actividad de nombres de esquema o tabla.

Tabla 80. Registros de anotaciones de DB2 (continuación)

Tipo	Nombre de registro	Descripción
Gestor de campos largos	"Registro de anotaciones de adición/supresión/no actualización de campo largo" en la página 606	Actividad de registro en campos largos.
Gestor de transacciones	"Registro de anotaciones de confirmación normal" en la página 596	Se confirma la transacción.
Gestor de transacciones	"Registro de anotaciones de confirmación heurística" en la página 597	Se confirma la transacción pendiente.
Gestor de transacciones	"Registro de anotaciones de confirmación de coordinador de MPP" en la página 598	Se confirma la transacción. Esto se escribe en un nodo de coordinador para una aplicación que realiza actualizaciones en un nodo de subordinador como mínimo.
Gestor de transacciones	"Registro de anotaciones de confirmación de subordinador de MPP" en la página 598	Se confirma la transacción. Esto se escribe en un nodo de subordinador.
Gestor de transacciones	"Registro de anotaciones de cancelación normal" en la página 599	Se cancela la transacción.
Gestor de transacciones	"Registro de anotaciones de cancelación heurística" en la página 600	Se cancela la transacción pendiente.
Gestor de transacciones	"Registro de anotaciones de lista pendiente local" en la página 600	Se confirma la transacción con una lista pendiente existente.
Gestor de transacciones	"Registro de anotaciones de lista pendiente global" en la página 601	Se confirma la transacción (en dos fases) con una lista pendiente existente.
Gestor de transacciones	"Registro de anotaciones de preparación de XA" en la página 601	Preparación de transacciones XA en entornos con confirmación en dos fases.
Gestor de transacciones	"Registro de anotaciones cronológicas de preparación de subordinador de MPP" en la página 602	Preparación de transacciones MPP en entornos con confirmación en dos fases. Este registro de anotaciones solamente existe en nodos de subordinador.
Gestor de transacciones	"Registro de anotaciones de preparación de TM" en la página 603	La preparación de la transacción coordinada como parte de una confirmación en dos fases, donde la base de datos actúa como base de datos de TM.
Gestor de transacciones	"Registro de anotaciones libre de retrotracción" en la página 603	Indica el final de un intervalo libre de retrotracción. El intervalo libre de retrotracción es un conjunto de registros de anotaciones que no es necesario compensar si se cancela transacción.

Tabla 80. Registros de anotaciones de DB2 (continuación)

Tipo	Nombre de registro	Descripción
Gestor de transacciones	"Registro de anotaciones de información de la aplicación" en la página 604	Información sobre la aplicación que ha iniciado la transacción.
Gestor de transacciones	"Registro de anotaciones de preparación federada" en la página 604	Información sobre el gestor de recursos federado implicado en la transacción.
Gestor de programas de utilidad	"Registro de anotaciones de comienzo de migración" en la página 607	Se inicia la migración de catálogos.
Gestor de programas de utilidad	"Registro de anotaciones de final de migración" en la página 607	Finaliza la migración de catálogos.
Gestor de programas de utilidad	"Registro de anotaciones de inicio de carga" en la página 608	Se inicia la carga de la tabla.
Gestor de programas de utilidad	"Registro de anotaciones de final de copia de seguridad" en la página 608	Finaliza la actividad de copia de seguridad.
Gestor de programas de utilidad	"Registro de anotaciones de avance del espacio de tablas" en la página 609	Finaliza el avance del espacio de tablas.
Gestor de programas de utilidad	"Registro de anotaciones de inicio del avance del espacio de tablas hasta punto en el tiempo" en la página 609	Indica el inicio de un avance de un espacio de tablas hasta un punto en el tiempo.
Gestor de programas de utilidad	"Registro de anotaciones de final del avance del espacio de tablas hasta punto en el tiempo" en la página 609	Indica el final de un avance de un espacio de tablas hasta un punto en el tiempo.

Cabecera del gestor de anotaciones

Todos los registros de anotaciones cronológicas de DB2 comienzan con una cabecera del gestor de anotaciones cronológicas. Esta cabecera contiene información sobre el registro de anotaciones e información de transacciones del transcriptor de registros de anotaciones.

Nota: Si el tipo de registro de anotaciones es 'i', es un registro informativo solamente. DB2 no lo tendrá en cuenta durante las operaciones de avance, retrotracción y recuperación de una anomalía.

Tabla 81. Cabecera del registro de anotaciones del gestor de anotaciones (LogManagerLogRecordHeader)

Descripción	Tipo	Desplazamiento (Bytes)
Longitud del registro de anotaciones completo	int	0(4)
Tipo de registro de anotaciones (Vea la Tabla 82 en la página 596).	short	4(2)
Distintivo general del registro de anotaciones[1]	short	6(2)

Tabla 81. Cabecera del registro de anotaciones del gestor de anotaciones (LogManagerLogRecordHeader) (continuación)

Descripción	Tipo	Desplazamiento (Bytes)
Número de secuencia del registro de anotaciones anterior escrito por esta transacción. Se utiliza para encadenar registros de anotaciones de acuerdo con la transacción. Si el valor es 0000 0000 0000, este es el primer registro de anotaciones escrito por la transacción.	SQLU_LSN[2]	8(6)
Identificador de transacción exclusivo	SQLU_TID[3]	14(6)
Número de secuencia del registro de anotaciones para esta transacción antes de la compensación del registro de anotaciones. (Nota: solamente para compensación y registros de anotaciones libres de retrotracción).	SQLU_LSN	20(6)
Número de secuencia del registro de anotaciones para esta transacción que se está compensando. (Nota: solamente para registros de anotaciones de compensación propagable).	SQLU_LSN	26(6)
Longitud total de la cabecera del registro de anotaciones del gestor de anotaciones: • *Sin compensación: 20 bytes* • *Compensación: 26 bytes* • *Compensación propagable: 32 bytes*		

Nota:

1. Constantes del distintivo general del registro de anotaciones

```
Rehacer siempre                                        0x0001
Propagable                                             0x0002
Tabla temporal                                         0x0004
Deshacer avance de espacio de tablas                   0x0008
Transacción singular (sin confirmación/retrotracc.)    0x0010
Recuperable condicionalmente                           0x0080
Avance espacio de tablas en proceso restricc. comp.    0x0100
```

2. Número de secuencia de anotaciones (LSN)

```
   Identificador exclusivo del registro de anotaciones que
representa la dirección relativa de byte del registro de
anotaciones dentro del archivo de anotaciones de la base
de datos.

   SQLU_LSN: union { unsigned char  [6] ;
                     unsigned short [3] ;
                     }
```

3. Identificador de transacción (TID)

```
   Identificador exclusivo de registro de anotaciones que
representa a la transacción.

   SQLU_TID: union { unsigned char  [6] ;
                     unsigned short [3] ;
                     }
```

4. ID de registro (RID)

```
Número exclusivo que identifica la posición de un registro.
RID: Número de página  char [4];
número ranura  char [2];
```

Tabla 82. Valores y definiciones para el tipo de registro de cabecera del gestor de anotaciones

Valor	Definición
0x0041	Cancelación normal
0x0042	Libre de retrotracción
0x0043	Compensación
0x0049	Cancelación heurística
0x004A	Inicio de carga
0x004E	Registro de anotaciones normal
0x004F	Final de copia de seguridad
0x0051	Lista pendiente global
0x0052	Rehacer
0x0055	Deshacer
0x0056	Inicio de migración
0x0057	Final de migración
0x0069	Solamente con fines informativos
0x006F	Inicio de copia de seguridad
0x0071	Final del avance del espacio de tablas hasta el final de punto en el tiempo
0x007B	Preparación de MPP
0x007C	preparación de XA
0x007D	Preparación del gestor de transacciones (TM)
0x0084	Confirmación normal
0x0085	Confirmación de subordinado de MPP
0x0086	confirmación de coordinador de MPP
0x0087	Confirmación heurística
0x0089	Inicio del avance del espacio de tablas hasta punto en el tiempo
0x008A	Lista pendiente local
0x008B	Información de la aplicación

Registros de anotaciones del gestor de transacciones

El gestor de transacciones crea registros de anotaciones que indican la finalización de sucesos de transacción (tales como una operación de confirmación o retrotracción). Las indicaciones horarias de los registros de anotaciones cronológicas utilizan la Hora Universal Coordinada (UTC), y registran el tiempo (en segundos) desde el 1 de enero de 1970.

Registro de anotaciones de confirmación normal

Este registro de anotaciones se escribe para una transacción en un entorno de un solo nodo o en un entorno de varios nodos, mientras que la transacción sólo afecta

a un nodo. El registro de anotaciones se escribe cuando se confirma una transacción después de uno de los sucesos siguientes:

1. Un usuario ha emitido una sentencia COMMIT
2. Se produce una confirmación implícita durante una operación CONNECT RESET

Tabla 83. Estructura del registro de anotaciones para la confirmación normal

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Hora en que se confirmó la transacción	sqluint64	20(8)
Longitud del identificador de autorización [1] (si el registro de anotaciones está marcado como propagable)	unsigned short	28(2)
Identificador de autorización de la aplicación[1] (si el registro de anotaciones está marcado como propagable)	char [ ]	30(variable[2])
Longitud total: 30 bytes más propagable variable (28 bytes no propagable)		

Nota:

1. Si el registro de anotaciones está marcado como propagable
2. Variable basada en la longitud del identificador de autorización

Registro de anotaciones de confirmación heurística

El registro de anotaciones se escribe cuando se confirma una transacción dudosa.

Tabla 84. Estructura del registro de anotaciones para la confirmación heurística

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Hora en que se confirmó la transacción	sqluint64	20(8)
Longitud del identificador de autorización [1] (si el registro de anotaciones está marcado como propagable)	unsigned short	28(2)
Identificador de autorización de la aplicación[1] (si el registro de anotaciones está marcado como propagable)	char [ ]	30(variable[2])
Longitud total: 30 bytes más propagable variable (28 bytes no propagable)		

Nota:

1. Si el registro de anotaciones está marcado como propagable
2. Variable basada en la longitud del identificador de autorización

Registro de anotaciones de confirmación de coordinador de MPP

El registro de anotaciones se escribe en un nodo de coordinador para una aplicación que realiza actualizaciones en un nodo de subordinador como mínimo.

Tabla 85. Estructura del registro de anotaciones para la confirmación de coordinador MPP

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Hora en que se confirmó la transacción	sqluint64	20(8)
Identificador MPP de la transacción	SQLP_GXID	28(20)
Número de nodo máximo	unsigned short	48(2)
TNL	unsigned char [ ]	50(max node number/8 + 1)
Longitud del identificador de autorización [1] (si el registro de anotaciones está marcado como propagable)	unsigned short	variable(2)
Identificador de autorización de la aplicación[1] (si el registro de anotaciones está marcado como propagable)	char [ ]	variable(variable[2])
Longitud total: variable		

Nota:

1. TNL define los nodos excepto el nodo coordinador que ha participado en una transacción
2. Variable basada en la longitud del identificador de autorización

Registro de anotaciones de confirmación de subordinador de MPP

El registro de anotaciones se escribe en un nodo de subordinador en MPP.

Tabla 86. Estructura del registro de anotaciones para la confirmación de subordinador MPP

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Hora en que se confirmó la transacción	sqluint64	20(8)
Identificador MPP de la transacción	SQLP_GXID	28(20)
Reservado	unsigned short	48(2)
Longitud del identificador de autorización [1] (si el registro de anotaciones está marcado como propagable)	unsigned short	50(2)

Tabla 86. Estructura del registro de anotaciones para la confirmación de subordinador MPP (continuación)

Descripción	Tipo	Desplazamiento (Bytes)
Identificador de autorización de la aplicación[2] (si el registro de anotaciones está marcado como propagable)	char [ ]	52 (variable[3])
Longitud total: 48 bytes más variable		

Nota:

1. Es el número de partición de base de datos actual si la transacción sólo está en una partición de base de datos; de lo contrario, es el número de la partición coordinadora.
2. Si el registro de anotaciones está marcado como propagable
3. Variable basada en la longitud del identificador de autorización

Registro de anotaciones de cancelación normal

El registro de anotaciones se escribe cuando se cancela una transacción después de uno de los sucesos siguientes:

- Un usuario ha emitido una sentencia ROLLBACK
- Se ha producido un punto muerto
- Se ha producido una retrotracción implícita durante una recuperación tras un error
- Se ha producido una retrotracción implícita durante una recuperación ROLLFORWARD (en avance).

Tabla 87. Estructura del registro de anotaciones para la cancelación normal

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Longitud del identificador de autorización [1] (si el registro de anotaciones está marcado como propagable)	unsigned short	20(2)
Identificador de autorización de la aplicación[1] (si el registro de anotaciones está marcado como propagable)	char [ ]	22(variable[2])
Longitud total: 22 bytes más variable propagable (20 bytes no propagables)		

Nota:

1. Si el registro de anotaciones está marcado como propagable
2. Variable basada en la longitud del identificador de autorización

Registro de anotaciones de cancelación heurística

El registro de anotaciones se escribe cuando se cancela una transacción dudosa.

Tabla 88. Estructura del registro de anotaciones para la cancelación heurística

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Longitud del identificador de autorización [1] (si el registro de anotaciones está marcado como propagable)	unsigned short	20(2)
Identificador de autorización de la aplicación[1] (si el registro de anotaciones está marcado como propagable)	char [ ]	22(variable[2])
Longitud total: 22 bytes más variable propagable (20 bytes no propagables)		

Nota:

1. Si el registro de anotaciones está marcado como propagable
2. Variable basada en la longitud del identificador de autorización

Registro de anotaciones de lista pendiente local

El registro de anotaciones se escribe si se confirma una transacción y existe una lista pendiente. La lista pendiente es una lista enlazada de operaciones no recuperables (tales como la supresión de un archivo) que solamente se pueden realizar cuando el usuario/aplicación emite una sentencia COMMIT. La estructura de longitud variable contiene las entradas de la lista pendiente.

Tabla 89. Estructura del registro de anotaciones para la lista pendiente local

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Hora en que se confirmó la transacción	sqluint64	20(8)
Longitud del identificador de autorización[1]	unsigned short	28(2)
Identificador de autorización de la aplicación[1]	char [ ]	30(variable)[2]
Entradas de la lista pendiente	DBCLOB	variable (variable)
Longitud total: 30 bytes más variables propagables (28 bytes más entradas de lista pendiente no propagables)		

Nota:

1. Si el registro de anotaciones está marcado como propagable
2. Variable basada en la longitud del identificador de autorización

Registro de anotaciones de lista pendiente global

El registro de anotaciones se escribe si se confirma una transacción que interviene en una confirmación en dos fases y existe una lista pendiente. La lista pendiente contiene operaciones no recuperables (tales como la supresión de un archivo) que solamente se pueden realizar cuando el usuario/aplicación emite una sentencia COMMIT. La estructura de longitud variable contiene las entradas de la lista pendiente.

Tabla 90. Estructura del registro de anotaciones para la lista pendiente global

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Longitud del identificador de autorización[1]	unsigned short	20(2)
Identificador de autorización de la aplicación[1]	char [ ]	22(variable)[2]
Entradas de la lista pendiente global	DBCLOB	variable (variable)
Longitud total: 22 bytes más variables propagables (20 bytes más entradas de lista pendiente no propagables)		

Nota:

1. Si el registro de anotaciones está marcado como propagable
2. Variable basada en la longitud del identificador de autorización

Registro de anotaciones de preparación de XA

Este registro de anotaciones se escribe para transacciones XA que se ejecutan en un entorno de un solo nodo o en el nodo coordinador en MPP. Solamente se utiliza para aplicaciones XA. El registro de anotaciones se escribe para indicar la preparación de la transacción como parte de una confirmación en dos fases. El registro de anotaciones de preparación de XA describe la aplicación que inició la transacción y se utiliza para reconstruir una transacción dudosa.

Tabla 91. Estructura del registro de anotaciones de preparación de XA

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Hora en que se preparó la transacción	sqluint64	20(8)
Espacio de registro de anotaciones utilizado por la transacción	sqluint64	28(8)
Tamaño de la lista de nodos de la transacción	sqluint32	36(4)
Lista de nodos de la transacción	unsigned char [ ]	40(variable)
Reservado	sqluint32	variable(2)
Identificador XA de la transacción	SQLXA_XID[1]	variable(140)

Tabla 91. Estructura del registro de anotaciones de preparación de XA (continuación)

Descripción	Tipo	Desplazamiento (Bytes)
Información de Synclog	DBCLOB	variable (variable)
Longitud total: 182 bytes más variables		

Nota: 1. Para obtener detalles sobre el tipo de anotaciones cronológica SQLXA_XID, consulte "SQLXA_XID" en la página 587.

Registro de anotaciones cronológicas de preparación de subordinador de MPP

Este registro de anotaciones se escribe para transacciones MPP en nodos de subordinador. El registro de anotaciones se escribe para indicar la preparación de la transacción como parte de una confirmación en dos fases. El registro de anotaciones de preparación del subordinador MPP describe la aplicación que inició la transacción y se utiliza para reconstruir una transacción dudosa.

Tabla 92. Estructura del registro de anotaciones para la preparación del subordinador MPP

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Hora en que se preparó la transacción	sqluint64	20(8)
Espacio de registro de anotaciones utilizado por la transacción	sqluint64	28(8)
LSN de coordinador	SQLP_LSN	36(6)
Relleno	char [ ]	42(2)
Identificador MPP de la transacción	SQLP_GXID[1]	44(20)
Longitud total: 64 bytes más variable		

Nota: 1.El registro de anotaciones SQLP-GXID se utiliza para identificar transacciones en un entorno MPP.

Tabla 93. Campos en la estructura de SQLP-GXID

Nombre de campo	Tipo de datos	Descripción
FORMATID	INTEGER	ID de formato GXID
GXID_LENGTH	INTEGER	Longitud de GXID
BQAL_LENGTH	INTEGER	Longitud del identificador de rama
DATA	CHAR(8)	Los 2 primeros bytes contienen el número de nodo; el recordatorio es el ID de transacción

Registro de anotaciones de preparación de TM

Este registro de anotaciones cronológicas se escribe para transacciones coordinadas de DB2 en un entorno de bases de datos de una sola partición o en la partición de coordinador en MPP, donde la base de datos actúa como base de datos de TM. El registro de anotaciones se escribe para indicar la preparación de la transacción como parte de una confirmación en dos fases.

Tabla 94. Estructura del registro de anotaciones de preparación de TM

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Hora en que se preparó la transacción	sqluint64	20(8)
Espacio de registro de anotaciones utilizado por la transacción	sqluint64	28(8)
Tamaño de la lista de nodos de la transacción	sqluint32	36(4)
Lista de nodos de la transacción	unsigned char [ ]	40(variable)
Reservado	sqluint32	variable(2)
Identificador XA de la transacción	SQLXA_XID	variable(140)
Información de Synclog	DBCLOB	variable (variable)
Longitud total: 182 bytes más variables		

Registro de anotaciones libre de retrotracción

Este registro de anotaciones se utiliza para marcar el final de un intervalo libre de retrotracción. El intervalo libre de retrotracción es un conjunto de registros de anotaciones que no es necesario compensar si se cancela transacción. Este registro de anotaciones contiene un número de secuencia de anotaciones de 6 bytes (*complsn*, almacenado en la cabecera de registro de anotaciones que comienza en la posición cuyo desplazamiento es 20). En determinadas situaciones, el registro de anotaciones libre de retrotracción también contiene datos de anotaciones, que comienzan en la posición con desplazamiento 26, que son los mismos datos que se registran en los correspondientes registros de anotaciones del gestor de datos. Cuando este registro de anotaciones se lee durante una retrotracción (después de una transacción cancelada), *complsn* indica el siguiente registro de anotaciones que se debe compensar.

Tabla 95. Estructura del registro de anotaciones libre de retrotracción

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Complsn	SQLP_LSN	20(6)
Datos de registro de anotaciones[1]	DBCLOB	DBCLOB
Longitud total: 26 bytes más variables		

Nota: 1. Solamente se aplica en determinadas situaciones, y cuando se utiliza, la longitud del registro de anotaciones completo contenida en la cabecera de registro es mayor que 26 bytes.

Registro de anotaciones de información de la aplicación

Este registro de anotaciones contiene información sobre la aplicación que ha iniciado esta transacción.

Tabla 96. Estructura del registro de anotaciones de información de la aplicación

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Hora de inicio de la transacción	sqluint32	20 (4)
Reservado	char[ ]	24(16)
Página de códigos	sqluint32	40(4)
Longitud del nombre de la aplicación	sqluint32	44(4)
Nombre de la aplicación	char [ ]	48(variable)
Longitud del identificador de la aplicación	sqluint32	variable(4)
Identificador de aplicación	char [ ]	variable (variable)
Longitud del número de secuencia	sqluint32	variable(4)
Número de secuencia	char [ ]	variable (variable)
Longitud del alias de base de datos utilizado por el cliente	sqluint32	variable(4)
Alias de base de datos utilizado por el cliente	char [ ]	variable (variable)
Longitud del identificador de autorización	sqluint32	variable(4)
Identificador de autorización	char [ ]	variable (variable)
Longitud total: 64 bytes más variables		

Registro de anotaciones de preparación federada

Este registro de anotaciones contiene información sobre los gestores de recursos federados implicados en la transacción.

Tabla 97. Estructura del registro de anotaciones de preparación federada

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Número de gestores de recursos	sqluint32	20 (4)
Longitud del identificador de autorización	sqluint16	24(2)

Tabla 97. Estructura del registro de anotaciones de preparación federada (continuación)

Descripción	Tipo	Desplazamiento (Bytes)
Longitud de la contraseña cifrada	sqluint16	26(2)
Identificador de autorización	char [128]	28(128)
Contraseña cifrada	char [255]	156(255)
Entradas del gestor de recursos	DBCLOB	411(variable)
Longitud total: 411 bytes más variables		

Registros de anotaciones del gestor de campos largos

Los registros de anotaciones el gestor de campos largos solamente se escriben si una base de datos se configura con las opciones LOG RETAIN o USEREXITS habilitadas. Estos registros de anotaciones se escriben cada vez que se insertan, suprimen o actualizan datos de campo largo.

Nota: Los registros de anotaciones del gestor de LOB no se pueden propagar y, por lo tanto, no están documentados.

Para ahorrar espacio en el archivo de anotaciones, no registran anotaciones para la adición de datos de campo largo a tablas si la base de datos está configurada para el registro de anotaciones circular. Además, cuando se actualiza un valor de campo largo, se duplica la imagen anterior y no se registra en el archivo de anotaciones.

Todos los registros de anotaciones del gestor de campos largos comienzan con una cabecera.

Todos los desplazamientos de los registros de anotaciones del gestor de campos largos están determinados con respecto al final de la cabecera de registro del gestor de anotaciones.

Cuando se altera una tabla para capturar columnas LONG VARCHAR OR LONG VARGRAPHIC (especificando INCLUDE LONGVAR COLUMNS en la sentencia ALTER TABLE):

- El gestor de campos largos escribe el correspondiente registro de anotaciones de campo largo.
- Cuando se actualizan datos de campo largo, la actualización se trata como si fuera una supresión del valor antiguo del campo largo, seguida de una inserción del nuevo valor. Para determinar si un registro de Suprimir/añadir campo largo está asociado o no a una operación de actualización de la tabla, el valor original de la operación se registra en el registro de anotaciones del gestor de campos largos.
- Cuando se actualizan tablas que tienen columnas de campo largo, pero sin que se actualicen las propias columnas de campo largo, se escribe un registro de No actualización de campo largo.
- El Registro de supresión de campo largo y el Registro de no actualización de campo largo son registros de anotaciones con fines informativos solamente.

Tabla 98. Cabecera del registro de anotaciones del gestor de campos largos (LongFieldLogRecordHeader)

Descripción	Tipo	Desplazamiento (Bytes)
Código de origen (identificador de componente = 3)	unsigned char	0(1)
Tipo de operación (Vea la Tabla 99).	unsigned char	1(1)
Identificador de espacio de tablas	unsigned short	2(2)
Identificador de objeto	unsigned short	4(2)
Identificador del espacio de tablas padre[1]	unsigned short	6(2)
Identificador del objeto padre[2]	unsigned short	8(2)
Longitud total: 10 bytes		

Nota:

1. ID de espacio de tablas del objeto de datos
2. ID de objeto del objeto de datos

Tabla 99. Valores y definiciones para el tipo de operación del registro de cabecera del gestor de campos largos

Valor	Definición
113	Registro de adición de campo largo
114	Registro de supresión de campo largo
115	Registro de no actualización de campo largo

Registro de anotaciones de adición/supresión/no actualización de campo largo

Estos registros de anotaciones se escriben cada vez que se añaden, suprimen o actualizan datos de campo largo. La longitud de los datos se redondea por exceso hasta el múltiplo siguiente de 512 bytes.

Tabla 100. Estructura del registro de anotaciones de adición/supresión/no actualización de campo largo

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LongFieldLogRecordHeader	0(10)
Interno	Interno	10(1)
Tipo de operación original[1]	char	11(1)
Identificador de columna[2]	unsigned short	12(2)
Longitud de campo largo[3]	unsigned short	14(2)
Desplazamiento de archivo[4]	sqluint32	16(4)
Datos de campo largo	char[ ]	20(variable)

Nota:

1. Tipo de operación original

   ```
   1     Insertar
   2     Suprimir
   4     Actualizar
   ```

2. Número de columna al que se aplica el registro de anotaciones. El número de columna comienza en 0.
3. Longitud de datos de campo largo en sectores de 512 bytes (la longitud real de los datos se graba como los 4 primeros bytes del descriptor de campo largo (descriptor LF), que está registrado en el siguiente registro de anotaciones de inserción/supresión/actualización como parte de un registro de datos de usuario formateado). El valor de este campo es siempre positivo.

   El gestor de campos largos nunca escribe registros de anotaciones para datos de campo largo de longitud cero que se están añadiendo, suprimiendo o actualizando.
4. Desplazamiento, expresado en sectores de 512 bytes, dentro del objeto de campo largo donde se deben colocar los datos.

Registros de anotaciones del gestor de programas de utilidad

El gestor de programas de utilidad genera registros de anotaciones cronológicas asociados a los siguientes programas de utilidad de DB2:

- Migración
- Cargar
- Copia de seguridad
- Avance del espacio de tablas.

Los registros de anotaciones indican el comienzo o final de la actividad solicitada. Sólo se documentan los registros de anotaciones propagables para estos programas de utilidad.

Registro de anotaciones de comienzo de migración

Este registro de anotaciones está asociado con el comienzo de una migración de catálogos.

Tabla 101. Estructura del registro de anotaciones de comienzo de migración

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Hora de inicio de la migración	char[ ]	20(10)
Release desde el que se migra	unsigned short	30(2)
Release hacia el que se migra	unsigned short	32(2)
Longitud total: 34 bytes		

Registro de anotaciones de final de migración

Este registro de anotaciones está asociado con la finalización satisfactoria de una migración de catálogos.

Tabla 102. Estructura del registro de anotaciones de final de migración

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Hora de final de migración	char[ ]	20(10)
Release hacia el que se migra	unsigned short	30(2)
Longitud total: 32 bytes		

Registro de anotaciones de inicio de carga

Este registro de anotaciones está asociado con el comienzo de una carga.

Es el único registro de anotaciones de carga que es propagable. Se escribe al principio de la fase de carga. Este registro de anotaciones no debe confundirse con otros tipos de registros de inicio de carga que se escriben al principio de una fase de inicio que no es propagable.

Para lograr la propagación del registro de anotaciones, se recomienda que después de leer un registro de anotaciones de Inicio de anotaciones, no continúe propagando registros de anotaciones para la tabla específica a una tabla de destino. Después de un registro de anotaciones de Inicio de carga, se pueden omitir todos los registros de anotaciones propagables que pertenecen a la tabla que se está cargando independientemente de los límites de la transacción, hasta que llegue el momento de realizar un rearranque en frío. Es necesario realizar un rearranque en frío para sincronizar las tablas fuente y destino.

Tabla 103. Estructura del registro de anotaciones de comienzo de carga

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Identificador de registro de anotaciones	sqluint32	20 (4)
Identificador de agrupación	unsigned short	24(2)
Identificador de objeto	unsigned short	26(2)
Distintivo	sqluint32	28(4)
Lista de agrupaciones de objetos	DBCLOB	32(variable)
Longitud total: 32 bytes más variable		

Registro de anotaciones de final de copia de seguridad

Este registro de anotaciones está asociado con el final de una copia de seguridad satisfactoria.

Tabla 104. Estructura del registro de anotaciones de final de copia de seguridad

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Hora de finalización de la copia de seguridad	sqluint64	20(8)

Tabla 104. Estructura del registro de anotaciones de final de copia de seguridad (continuación)

Descripción	Tipo	Desplazamiento (Bytes)
Longitud total: 28 bytes		

Registro de anotaciones de avance del espacio de tablas

Este registro de anotaciones está asociado con la recuperación en avance (ROLLFORWARD) del espacio de tablas. Este registro se escribe para cada espacio de tablas para el que se ha realizado satisfactoriamente un avance.

Tabla 105. Estructura del registro de anotaciones de avance del espacio de tablas

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Identificador de espacio de tablas	sqluint32	20 (4)
Longitud total: 24 bytes		

Registro de anotaciones de inicio del avance del espacio de tablas hasta punto en el tiempo

Este registro de anotaciones está asociado con la recuperación en avance (ROLLFORWARD) del espacio de tablas. Indica el inicio de un avance de un espacio de tablas hasta un punto en el tiempo.

Tabla 106. Estructura del inicio del avance del espacio de tablas hasta punto en el tiempo

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Indicación de fecha y hora para este registro de anotaciones cronológicas	sqluint64	20(8)
Indicación de fecha y hora para la que se realiza el avance de los espacios de tablas	sqluint32	28(4)
Número de agrupaciones sometidas al avance	sqluint32	32(4)
Longitud total: 36 bytes		

Registro de anotaciones de final del avance del espacio de tablas hasta punto en el tiempo

Este registro de anotaciones está asociado con la recuperación en avance (ROLLFORWARD) del espacio de tablas. Indica el final de un avance de un espacio de tablas hasta un punto en el tiempo.

Tabla 107. Estructura del registro del final del avance del espacio de tablas hasta punto en el tiempo

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	LogManagerLogRecordHeader	0(20)
Indicación de fecha y hora para este registro de anotaciones cronológicas	sqluint64	20(8)
Indicación de fecha y hora en la que se realizó el avance de los espacios de tablas	sqluint32	28(4)
Indicador cuyo valor es `TRUE` si el avance fue correcto o `FALSE` si el avance se canceló.	sqluint32	32(4)
Longitud total: 36 bytes		

Son necesarios dos campos de indicación de fecha y hora para proporcionar la precisión adecuada de modo que los tiempos de sucesos de anotaciones de sucesos se puedan diferenciar. La primera indicación de fecha y hora utiliza 8 bytes para indicar la hora a la que se escribió la anotación en una precisión de segundos. Los 4 primeros bytes de esta indicación de fecha y hora indican la parte de segundos. Puesto que muchas acciones pueden realizarse en un segundo, para comprender el orden de los sucesos es necesario que haya mayor precisión. El segundo campo de indicación de fecha y hora proporciona 4 bytes que se utilizan para representar nanosegundos. Si las indicaciones de fecha y hora del registro de anotaciones son idénticas, el campo de indicación de fecha y hora de 4 bytes adicional se puede utilizar para determinar el orden de los sucesos de anotaciones asociados.

Registros de anotaciones del gestor de datos

Los registros de anotaciones del gestor de datos son el resultado de actividades de DDL, DML o de programas de utilidad.

Existen dos tipos de registros de anotaciones del gestor de datos:

- Los registros de anotaciones de Data Management System (DMS) tienen un 1 como identificador de componente en su cabecera.
- Los registros de anotaciones de Data Object Manager (DOM) tienen un 4 como identificador de componente en su cabecera.

Tabla 108. Estructura de la cabecera del registro de anotaciones de DMS (DMSLogRecordHeader)

Descripción	Tipo	Desplazamiento (Bytes)
Identificador de componente (=1)	unsigned char	0(1)
Identificador de función (Vea la Tabla 109 en la página 611.)	unsigned char	1(1)
Identificadores de tabla		
Identif. espacio tablas	unsigned short	2(2)
Identificador de tabla	unsigned short	4(2)
Longitud total: 6 bytes		

Tabla 109. Valores y definiciones del identificador de función en la estructura de la cabecera del registro de anotaciones de DMS

Valor	Definición
102	Añadir columnas a tabla
104	Deshacer adición de columnas
110	Deshacer inserción de registro
111	Deshacer supresión de registro
112	Deshacer actualización de registro
113	Alterar columna
115	Deshacer alteración de columna
122	Renombrar un esquema o tabla
123	Rehacer renombramiento de un esquema o una tabla
124	Atributo de alterar tabla
128	Inicializar tabla
131	Deshacer inserción de registro en página vacía
161	Suprimir registro
162	Insertar registro
163	Actualizar registro
164	Suprimir registro de página vacía
165	Insertar registro en página vacía
166	Deshacer supresión de registro de página vacía
167	Insertar varios registros
168	Deshacer inserción de varios registros

Tabla 110. Estructura de la cabecera del registro de anotaciones de DOM (DOMLogRecordHeader)

Descripción	Tipo	Desplazamiento (Bytes)
Identificador de componente (=4)	unsigned char	0(1)
Identificador de función (Vea la Tabla 111 en la página 612.)	unsigned char	1(1)
Identificadores de objeto		
Identif. espacio tablas	unsigned short	2(2)
Identificador de objeto	unsigned short	4(2)
Identificadores de tabla		
Identif. espacio tablas	unsigned short	6(2)
Identificador de tabla	unsigned short	8(2)
Tipo de objeto	unsigned char	10(1)

Tabla 110. Estructura de la cabecera del registro de anotaciones de DOM (DOMLogRecordHeader) (continuación)

Descripción	Tipo	Desplazamiento (Bytes)
Distintivos	unsigned char	11(1)
Longitud total: 12 bytes		

Tabla 111. Valores y definiciones del identificador de función en la estructura de la cabecera del registro de anotaciones de DOM

Valor	Definición
2	Crear índice
3	Descartar índice
4	Descartar tabla
5	Deshacer descarte de tabla
11	Truncar tabla (importar sustitución)
12	Activar NOT LOGGED INITIALLY
35	Reorganizar tabla
101	Crear tabla
130	Deshacer creación de tabla

Nota: Todos los desplazamientos de los registros de anotaciones del gestor de datos están determinados con respecto al final de la cabecera de registro del gestor de anotaciones.

Todos los registros de anotaciones cuyo nombre corto de identificador de función comienza con `UNDO` son registros de anotaciones escritos durante la operación UNDO o ROLLBACK de la acción en cuestión.

La operación ROLLBACK puede ser el resultado de lo siguiente:

- El usuario ha emitido la sentencia de transacción ROLLBACK
- Un punto muerto ha causado la retrotracción de una transacción seleccionada
- Se ha producido la retrotracción de transacciones no confirmadas después de una recuperación tras un error
- Se ha producido la retrotracción de transacciones no confirmadas después de ejecutar RESTORE y emitir ROLLFORWARD para los archivos de anotaciones.

Registro de anotaciones de inicialización de tabla

El registro de anotaciones de inicializar tabla se escribe cuando se crea una nueva tabla permanente. El registro de anotaciones indica la inicialización de la tabla. Este registro aparece después de los registros de anotaciones que crean los objetos de almacenamiento de datos, y antes de los registros de anotaciones que crean los objetos de almacenamiento de LF y de LOB. Esto es un registro de anotaciones de rehacer. El ID de función es 128.

Tabla 112. Estructura del registro de anotaciones de inicializar tabla

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DMSLogRecordHeader	0(6)
LSN de creación de archivo	SQLU_LSN	6(8)

Tabla 112. Estructura del registro de anotaciones de inicializar tabla (continuación)

Descripción	Tipo	Desplazamiento (Bytes)
Interno	Interno	14(74)
Longitud de la descripción de la tabla	sqluint32	88(4)
Registro de descripción de la tabla	DBCLOB	92(variable)
Longitud total: 92 bytes más longitud del registro de descripción de la tabla		

Tabla 113. Registro de descripción de la tabla

Descripción	Tipo	Desplazamiento (Bytes)
tipo de registro	unsigned char	0(1)
Interno	Interno	1(1)
número de columnas	unsigned short	2(2)
matriz de descriptores de columna	variable long	DBCLOB
Longitud total: 4 bytes más la longitud de la matriz de descriptores de columna		

Registro de descripción de tabla: matriz de descriptores de columna

(número de columnas) * 8, donde cada elemento de la matriz contiene:

- tipo de campo (unsigned short, 2 bytes)

```
SMALLINT     0x0000
INTEGER      0x0001
DECIMAL      0x0002
DOUBLE       0x0003
REAL         0x0004
BIGINT       0x0005
DECFLOAT64   0x0006
DECFLOAT128  0x0007
CHAR         0x0100
VARCHAR      0x0101
LONG VARCHAR 0x0104
DATE         0x0105
TIME         0x0106
TIMESTAMP    0x0107
BLOB         0x0108
CLOB         0x0109
STRUCT       0x010D
XMLTYPE      0x0112
GRAPHIC      0x0200
VARGRAPH     0x0201
LONG VARG    0x0202
DBCLOB       0x0203
```

- longitud (2 bytes)
  - Si es BLOB, CLOB o DBCLOB, este campo no se utiliza. Para conocer la longitud máxima de este campo, vea la matriz que sigue a continuación de la matriz de descriptores de columna.
  - Si no es DECIMAL, la longitud es la longitud máxima del campo (short).
  - Si es PACKED DECIMAL: Byte 0, unsigned char, precisión (longitud total) Byte 1, unsigned char, escala (dígitos de fracción).
- distintivo nulo (unsigned short, 2 bytes)
  - mutuamente exclusivo: permite nulos o no permite nulos

– opciones válidas: ningún valor por omisión, valor por omisión de tipo, valor por omisión de usuario, generado o valor de tipo de compresión

```
ISNULL                  0x0001
NONULLS                 0x0002
TYPE_DEFAULT            0x0004
USER_DEFAULT            0x0008
GENERATED               0x0040
COMPRESS_SYSTEM_DEFAULT 0x0080
```

- desplazamiento de campo (unsigned short, 2 bytes) Esto es el desplazamiento desde el inicio de la parte de longitud fija del registro de usuario hasta el lugar donde se encuentra el valor fijado del campo.

Registro de descripción de tabla: matriz de descriptores de columna de LOB

(número de campos de LOB, CLOB y DBCLOB) * 12, donde cada elemento de la matriz contiene:

- longitud (MAX LENGTH OF FIELD, sqluint32, 4 bytes)
- reservado (interno, sqluint32, 4 bytes)
- distintivo de registro (IS COLUMN LOGGED, sqluint32, 4 bytes)

El primer LOB, CLOB o DBCLOB encontrado en la matriz de descriptores de columna utiliza el primer elemento de la matriz de descriptores de LOB. El segundo LOB, CLOB o DBCLOB encontrado en la matriz de descriptores de columna utiliza el segundo elemento de la matriz de descriptores de LOB, y así sucesivamente.

Registro de anotaciones de sustitución de importación (truncamiento)

El registro de anotaciones de importar sustitución (truncar) se escribe cuando se ejecuta una acción IMPORT REPLACE. Este registro indica la re-inicialización de la tabla (ningún registro de usuario, LSN nuevo). Los identificadores de tabla de la cabecera de anotaciones identifican la tabla que se está truncando (IMPORT REPLACE). Esto es un registro de anotaciones normal. El ID de función es 11.

Tabla 114. Estructura del registro de anotaciones de importar sustitución (truncar)

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DOMLogRecordHeader	0(12)
Interno	Interno	12(variable)
Longitud total: 12 bytes más longitud variable		

Registro de anotaciones de activación no registrada inicialmente

El registro de anotaciones de activación no registrada inicialmente se escribe cuando un usuario emite una sentencia ALTER TABLE que incluye la cláusula ACTIVATE NOT LOGGED INITIALLY. Esto es un registro de anotaciones normal. Es el ID de función 12.

Tabla 115. Estructura de Activación no registrada inicialmente

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DOMLogRecordHeader	0(12)
Interno	Interno	12(4)
ID espacio de tablas largo*	unsigned short	16(2)

Tabla 115. Estructura de Activación no registrada inicialmente (continuación)

Descripción	Tipo	Desplazamiento (Bytes)
ID de espacio de tablas de índice*	unsigned short	18(2)
ID de objeto de índice	unsigned short	20(2)
ID de objeto LF	unsigned short	22(2)
ID de objeto LOB	unsigned short	24(2)
ID de objeto XML	unsigned short	26(2)
Longitud total: 28 bytes		

** Mismo que los Identificadores del espacio de tablas en la cabecera DOM; es un identificador exclusivo para cada espacio de tablas definido en la base de datos.*

Registro de anotaciones de retrotracción de inserción

El registro de anotaciones de retrotraer inserción se escribe cuando se retrotrae una acción de inserción de fila (INSERT RECORD). Esto es un registro de anotaciones de compensación. El ID de función es 110.

Tabla 116. Estructura del registro de anotaciones de retrotraer inserción

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DMSLogRecordHeader	0(6)
Interno	Interno	6(2)
Longitud de registro	unsigned short	8(2)
Espacio libre	unsigned short	10(2)
RID	char[]	12(6)
Longitud total: 16 bytes		

Registro de anotaciones de reorganización de tabla

El registro de anotaciones de reorganizar tabla se escribe cuando se ha confirmado la ejecución del programa REORG para reorganizar una tabla. Esto es un registro de anotaciones normal. El ID de función es 35.

Tabla 117. Estructura del registro de anotaciones de reorganizar tabla

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DOMLogRecordHeader	0(12)
Interno	DBCLOB	12(476)
Símbolo de índice[1]	unsigned short	488(2)
ID de espacio de tablas temporales[2]	unsigned short	490(2)
ID de espacio de tablas temporal largo	unsigned short	492(2)
Longitud total: 494 bytes		

Nota:

1. Si el valor del símbolo de índice no es 0, este valor es el índice utilizado para agrupar la reorganización (índice de agrupación).
2. Si el ID del espacio de tablas temporales no es 0, este valor es el espacio de tablas temporales del sistema que se utilizó para crear la tabla reorganizada.

Registros de anotaciones de crear índice, descartar índice

Estos registros de anotaciones se escriben cuando se crean o se descartan índices. Los dos elementos del registro de anotaciones son:

- La página raíz del índice, que es un identificador interno
- El símbolo de índice, que es equivalente a la columna `IID` de SYSIBM.SYSINDEXES. Si el valor de este elemento es 0, el registro de anotaciones representa una acción sobre un índice interno, y no está relacionado con ningún índice de usuario.

Esto es un registro de anotaciones normal. El ID de función es 2 (crear índice) o 3 (descartar índice).

Tabla 118. Estructura de los registros de anotaciones de crear índice, descartar índice

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DOMLogRecordHeader	0(12)
Interno	Interno	12(2)
Símbolo de índice	unsigned short	14(2)
Página raíz del índice	sqluint32	16(4)
Longitud total: 20 bytes		

Registros de anotaciones para crear tabla, descartar tabla, retrotraer creación de tabla, retrotraer el descarte de la tabla

Estos registros de anotaciones se escriben cuando se crea o se descarta el objeto DATA para una tabla permanente. Para la creación de una tabla MDC, existe también un registro de anotaciones de crear tabla para la creación del objeto Mapa de bloques. El objeto DATA (y el objeto Mapa de bloques si es aplicable) se crea durante una operación CREATE TABLE y antes de la inicialización de la tabla (Inicializar tabla). Crear tabla y descartar tabla son registros de anotaciones normales. Retrotraer creación de tabla y retrotraer el descarte de tabla son registros de anotaciones de compensación. El ID de función es 101 (crear tabla), 4 (descartar tabla), 130 (retrotraer creación de tabla) o 5 (retrotraer descarte de tabla).

Tabla 119. Estructura de los registros de anotaciones de crear tabla, descartar tabla, retrotraer creación de tabla, retrotraer descarte de tabla

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DOMLogRecordHeader	0(12)
Interno	DBCLOB	12(72)
Longitud total: 84 bytes		

Registro de anotaciones de atributo de alterar tabla

El registro de anotaciones del atributo de alterar tabla se escribe cuando se cambia el estado de una tabla utilizando la sentencia ALTER TABLE o como resultado de añadir o validar restricciones. Puede ser un registro de anotaciones Normal o de

Compensación. El ID de función es 124.

Tabla 120. Atributo de alterar tabla, atributo de deshacer alteración de tabla

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DMSLogRecordHeader	0(6)
Máscara (atributo) de alterar bit	sqluint64	6(8)
Valores (atributo) de alterar bit	sqluint64	14(8)
Longitud total: 22 bytes		

Bits de atributo

```
0x00000001   Propagación
0x00000002   Comprobación pendiente
0x00000010   Compresión de valores
0x00010000   Modalidad de adición
0x00200000   Propagación LF
```

Todos los demás bits son para uso interno.

Si uno de los bits indicados anteriormente está presente en la máscara de alterar bit, significa que se está alterando este atributo de la tabla. Para determinar el nuevo valor del atributo de la tabla (0 = OFF y 1 = ON), examine el bit correspondiente en el valor de alterar bit.

Registro de anotaciones de añadir columnas por alteración de tabla, retrotraer adición de columnas

El registro de anotaciones de añadir columnas por alteración de tabla se escribe cuando el usuario añade a columnas a una tabla existente mediante una sentencia ALTER TABLE. Se registra información completa sobre las columnas antiguas y las columnas nuevas.

- Los elementos de número de columnas representan el número antiguo de columnas y el nuevo número total de columnas.
- Las matrices paralelas contienen información sobre las columnas definidas en la tabla. La matriz paralela antigua define la tabla existente antes de emitir la sentencia ALTER TABLE, mientras que la nueva matriz paralela define la tabla resultante de la sentencia ALTER TABLE.
- Cada matriz paralela consta de:
  - Un elemento de 8 bytes para cada columna.
  - Si existe alguna columna de LOB, un elemento de 12 bytes para cada columna de LOB. Esto va a continuación de la matriz de elementos de 8 bytes.

Añadir columnas por alteración de tabla es un registro de anotaciones normal. Retrotraer la adición de columnas es un registro de anotaciones de compensación. Los ID de función son 102 (añadir columna) o 104 (deshacer adición de columna).

Tabla 121. Estructura de los registros de anotaciones de añadir columnas por alteración de tabla, retrotraer adición de columnas

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DMSLogRecordheader	0(6)
Interno	Interno	6(2)
Número de columnas antiguo	sqluint32	8(4)

Tabla 121. Estructura de los registros de anotaciones de añadir columnas por alteración de tabla, retrotraer adición de columnas (continuación)

Descripción	Tipo	Desplazamiento (Bytes)
Número de columnas nuevo	sqluint32	12(4)
Matrices paralelas antiguas[1]	DBCLOB	16(variable)
Matrices paralelas nuevas	DBCLOB	variable(variable)
Longitud total: 16 bytes más 2 conjuntos de matrices paralelas.		

Elementos de la matriz:

1. Las longitudes de los elementos de esta matriz se definen del modo siguiente:
   - Si el elemento es un descriptor de columna, la longitud de columna es de 8 bytes.
   - Si el elemento es un descriptor de columna de LOB, la longitud del elemento es de 12 bytes.

Para obtener información sobre la matriz de descriptores de columna o la matriz de descriptores de columna de LOB, vea la descripción que hay después de la Tabla 113 en la página 613.

Registro de anotaciones de atributo de alterar columna

El ID de función es 113.

Tabla 122. Estructura del registro del atributo de alterar columna

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DMSLogRecordheader	0(6)
ID de columna	unsigned short	6(2)
Definición de columna antigua	Descriptor de columna[1]	8(8)
Definición de columna nueva	Descriptor de columna[1]	16(8)
Longitud total: 24 bytes más longitud de registro.		

[1]Para obtener una descripción de la matriz del descriptor de columna, vea la descripción que hay después de la Tabla 113 en la página 613.

Registro de anotaciones del atributo de deshacer alteración de columna

El ID de función es 115.

Tabla 123. Estructura del registro del atributo de deshacer alteración de columna

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DMSLogRecordHeader	0(6)
ID de columna	unsigned short	6(2)
Definición de columna antigua	Descriptor de columna[1]	8(8)
Definición de columna nueva	Descriptor de columna[1]	16(8)

Tabla 123. Estructura del registro del atributo de deshacer alteración de columna (continuación)

Descripción	Tipo	Desplazamiento (Bytes)
Longitud total: 24 bytes más longitud de registro.		

[1]Para obtener una descripción de la matriz del descriptor de columna, vea la descripción que hay después de la Tabla 113 en la página 613.

Registro de anotaciones de insertar registro, retrotraer supresión de registro, retrotraer actualización de registro

Estos registros de anotaciones se escriben cuando se insertan filas en una tabla, o se retrotrae una supresión o actualización. Los registros de anotaciones Insertar registro y Suprimir registro también se pueden crear durante una actualización, si la ubicación del registro que se actualiza se debe cambiar para dar cabida a los datos del registro modificado. Los registros de anotaciones Insertar registro son registros de anotaciones normales. Los registros Retrotraer supresión y Retrotraer actualización son registros de anotaciones de comprensación. Los ID de función son 162 (insertar), 111 (retrotraer supresión) o 112 (retrotraer actualización).

Tabla 124. Estructura de los registros de anotaciones de insertar registro, retrotraer supresión de registro, retrotraer actualización de registro

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DMSLogRecordHeader	0(6)
Interno	Interno	6(2)
Longitud de registro	unsigned short	8(2)
Espacio libre	unsigned short	10(2)
RID	char[]	12(6)
Desplazamiento de registro	unsigned short	18(2)
Cabecera y datos de registro	DBCLOB	20(variable)
Longitud total: 20 bytes más longitud de registro		

A continuación se ofrecen detalles sobre la cabecera y los datos del registro:

Cabecera de registro

- 4 bytes
- Tipo de registro (unsigned char, 1 byte).
- Reservado (char, 1 byte)
- Longitud de registro (unsigned short, 2 bytes)

Registro

- Longitud variable
- Tipo de registro (unsigned char, 1 byte).
- Reservado (char, 1 byte)
- El resto del registro depende del tipo de registro y del registro descriptor de tabla definido para la tabla.
- Los campos siguientes son válidos para registros de datos de usuario con un tipo de registro con el bit 1 establecido:

- Longitud fija (unsigned short, 2 bytes). Esto es la longitud de la sección de longitud fija de la fila de datos.
- Registro formateado (todas las columnas de longitud fija, seguidas de las columnas de longitud variable).

• Los campos siguientes son válidos para registros de datos de usuario con un tipo de registro con el bit 2 establecido:
  - Número de columnas (unsigned short, 2 bytes). Esto es el número de columnas en la porción de datos de la fila de datos. Consulte "Registro de datos de usuario formateado para una tabla con VALUE COMPRESSION" en la página 622.

    Nota: la matriz de desplazamiento contendrá 1 + el número de columnas.
  - Registro formateado (matriz de desplazamiento, seguida de las columnas de datos).

Un registro de usuario se especifica completamente mediante las características siguientes:

1. El tipo de registro exterior es 0, o
2. El tipo de registro exterior es 0x10, o
3. El tipo de registro exterior tiene el bit 0x04 establecido y
1. El tipo de registro interior tiene el bit 0x01 establecido o
2. El tipo de registro interior tiene el bit 0x02 establecido.

Nota: La compresión de filas y la captura de datos no son compatibles.

Registro de datos de usuario formateado para una tabla sin VALUE COMPRESSION

Para registros formateados sin VALUE COMPRESSION, todos los campos contienen una parte de longitud fija. Además, existen ocho tipos de campo que tienen partes de longitud variable:

- VARCHAR
- LONG VARCHAR
- BLOB
- CLOB
- VARGRAPHIC
- LONG VARG
- DBCLOB

La longitud de la porción fija de los diferentes tipos de campos se puede determinar de esta manera:

- DECIMAL

  Este campo es un decimal empaquetado estándar con el formato siguiente: *nnnnnn...s*. La longitud del campo es: (precisión + 2)/2. La parte del signo (s) es xC para positivo (+), y xD o xB para negativo (-).
- SMALLINT INTEGER BIGINT DOUBLE REAL CHAR GRAPHIC

  El campo de longitud contenido en el elemento para esta columna en el registro descriptor de tabla contiene el tamaño de la longitud fija del campo.
- DATE

Este campo es un decimal empaquetado de 4 bytes, en la forma: *aaaammdd*. Por ejemplo, el 3 de abril de 1996 se representa como x'19960403'.

- TIME

  Este campo es un decimal empaquetado de 3 bytes, en la forma: *hhmmss*. Por ejemplo, 1:32PM se representa como x'133200'.

- TIMESTAMP

  Este campo es un decimal empaquetado de 10 bytes con el formato siguiente: *aaaammddhhmmssuuuuuu* (DATE | TIME | microsegundos).

- VARCHAR LONG VARCHAR BLOB CLOB VARGRAPHIC LONG VARG DBCLOB

  La longitud de la porción fija de todos los campos de longitud variables es 4.

Las secciones siguientes describen la ubicación de la porción fija de cada campo dentro del registro formateado.

El registro descriptor de tabla describe el formato de columna de la tabla. Contiene una matriz de estructuras de columna, cuyos elementos representan el tipo de campo, la longitud de campo, el indicador de nulos y el desplazamiento de campo. El desplazamiento de campo es el desplazamiento con respecto al principio del registro formateado, donde reside la porción de longitud fija del campo.

Tabla 125. Estructura del registro descriptor de tabla

tipo de registro	número de columnas	estructura de columnas • tipo de campo • length • indicador de nulos • desplazamiento de campo	información de LOB

Nota: Para obtener más información, consulte la descripción que hay después de la Tabla 112 en la página 612.

Para las columnas que pueden contener valores nulos (de acuerdo con lo especificado por el indicador de nulos), existe un byte adicional a continuación de la porción fija del campo. Este byte contiene uno de estos dos valores:

- NOT NULL (0x00)
- NULL (0x01)

Si el valor del indicador de nulos del registro formateado para una columna que puede contener nulos es 0x00, existe un valor válido en la porción de datos de longitud fija del registro. Si el valor del indicador de nulos es 0x01, el valor del campo de datos es NULL.

El registro de datos de usuario formateado contiene los datos de la tabla que son visibles para el usuario. El registro de datos de usuario es un registro formateado con una sección de longitud fija seguido de una sección de longitud variable.

Tabla 126. Estructura del registro de datos de usuario formateado para la tabla sin VALUE COMPRESSION

tipo de registro	longitud de sección fija	sección de longitud fija	sección de datos variables

Nota: Para obtener más información, consulte la descripción que hay después de la Tabla 124 en la página 619.

Todos los tipos de campo variable tienen una porción de datos fija de 4 bytes en la sección de longitud fija (más un indicador de nulos si la columna puede contener nulos). Los primeros 2 bytes (short) representan el desplazamiento respecto del comienzo de la sección de longitud fija, donde residen los datos variables. Los 2 bytes siguientes (short) especifican la longitud de los datos variables referenciados por el valor de desplazamiento.

Registro de datos de usuario formateado para una tabla con VALUE COMPRESSION

Los registros formateados con VALUE COMPRESSION constan de la matriz de desplazamiento y la porción de datos. Cada entrada de la matriz es un desplazamiento de 2 bytes con respecto a los correspondientes datos de columna de la porción de datos. El número de datos de columna de la porción de datos se puede encontrar en la cabecera de registro, y el número de entradas de la matriz de desplazamiento es 1 más el número de datos de columna existentes en la porción de datos.

1. Los valores de columna comprimidos ocupan solamente 1 byte de espacio de disco, que se utiliza para el byte de atributo. El byte de atributo indica que los datos de la columna están comprimidos, por ejemplo, se conoce el valor de datos, pero no está almacenado en disco. El bit de orden superior (0x8000) del desplazamiento se utiliza para indicar que los datos accedidos son un byte de atributo. (Solamente se utilizan 15 bits para representar el desplazamiento de los correspondientes datos de columna).
2. Para los datos de columna normales, los datos de columna siguen a continuación de la matriz de desplazamiento. No existe ningún byte de atributo ni indicador de longitud.
3. Los datos accedidos pueden tomar dos valores diferente si son un byte de atributo:
   - ```
     NULL   0x01  (el valor es NULL)
     ```
   - ```
     COMPRESSED SYSTEM DEFAULT   0x80   (el valor es igual al
     valor por omisión del sistema)
     ```
4. La longitud de los datos de columna es la diferencia entre el desplazamiento actual y el desplazamiento de la columna siguiente.

Tabla 127. Estructura del registro de datos de usuario formateado para la tabla con VALUE COMPRESSION

tipo de registro	número de columna en porción de datos	matriz de desplazamiento	porción de datos

Nota: Para obtener más información, consulte la descripción que hay después de la Tabla 124 en la página 619.

Registros de anotaciones de insertar registro en página vacía, suprimir registro en página vacía, retrotraer supresión de registro en página vacía y retrotraer inserción de registro en página vacía

Estos registros de anotaciones se escriben cuando la tabla es una tabla agrupada multidimensional (tabla MDC). El registro de anotaciones Insertar registro en página vacía se escribe cuando se inserta un registro y es el primer registro de una página, y esa página no es la primera página de un bloque. Este registro de anotaciones registra la inserción en la página, así como la actualización de un bit en la primera página del bloque, para indicar que esa página ya no está vacía. El registro de anotaciones Suprimir registro en página vacía se escribe cuando se suprime el último registro de una página, y esa página no es la primera página de un bloque. Este registro de anotaciones registra la supresión realizada en la página, así como la actualización de un bit en la primera página del bloque, para indicar que la página está vacía. Los registros de anotaciones Insertar registro en página vacía y Suprimir registro en página vacía son registros de anotaciones normales. Los registros de anotaciones Retrotraer supresión de registro y Retrotraer inserción de registro son registros de anotaciones de comprensión. Los ID de función son 165 (insertar registro en página vacía), 164 (suprimir registro en página vacía), 166 (retrotraer supresión de registro en página vacía) o 131 (retrotraer inserción de registro en página vacía).

Tabla 128. Retrotraer inserción de registro en página vacía

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DMSLogRecordHeader	0(6)
Interno	Interno	6(2)
Longitud de registro	unsigned short	8(2)
Espacio libre	unsigned short	10(2)
RID	char[]	12(6)
Interno	Interno	18(2)
Primera página del bloque	sqluint32	20(4)
Longitud total: 24 bytes		

Tabla 129. Insertar registro en página vacía, retrotraer supresión de registro en página vacía, suprimir registro en página vacía

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DMSLogRecordHeader	0(6)
Interno	Interno	6(2)
Longitud de registro	unsigned short	8(2)
Espacio libre	unsigned short	10(2)
RID	char[]	12(6)
Interno	Interno	18(2)
Primera página del bloque	sqluint32	20(4)
Desplazamiento de registro	unsigned short	24(2)
Cabecera y datos de registro	DBCLOB	26(variable)
Longitud total: 26 bytes más longitud de registro		

Nota: Para obtener información detallada sobre la cabecera y los datos del registro, consulte la descripción que hay después de la Tabla 124 en la página 619.

Registro de anotaciones de actualizar registro

El registro de anotaciones de actualizar registro se escribe cuando se actualiza una fila y su ubicación en el almacenamiento sigue siendo la misma. Existen dos formatos de registro disponibles; son iguales a los registros de anotaciones de insertar registro y suprimir registro (vea "Registro de anotaciones de insertar registro, retrotraer supresión de registro, retrotraer actualización de registro" en la página 619). Un formato de registro contiene la imagen *anterior*a la actualización de la fila que se está actualizando; el otro formato de registro contiene la imagen *posterior* a la actualización de la fila que se está actualizando. Esto es un registro de anotaciones normal. El ID de función es 163.

Tabla 130. Estructura del registro de anotaciones de actualizar registro

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DMSLogRecordHeader	0(6)
Interno	Interno	6(2)
Longitud de registro nueva	unsigned short	8(2)
Espacio libre	unsigned short	10(2)
RID	char[]	12(6)
Desplazamiento de registro	unsigned short	18(2)
Cabecera y datos de registro antiguo	DBCLOB	20(variable)
Cabecera de registro	DMSLogRecordHeader	variable(6)
Interno	Interno	variable(2)
Longitud de registro antigua	unsigned short	variable(2)
Espacio libre	unsigned short	variable(2)
RID	char[]	variable(6)
Desplazamiento de registro	unsigned short	variable(2)
Cabecera y datos de registro nuevo	DBCLOB	variable(variable)
Longitud total: 40 bytes más 2 longitudes de registro		

Registro de anotaciones de renombramiento de una tabla o un esquema

El registro de anotaciones de renombramiento de una tabla o un esquema se escribe cuando se modifica un nombre de tabla o de esquema. Es el ID de función 122.

Tabla 131. Estructura del registro de renombramiento de una tabla o un esquema

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DMSLogRecordHeader	0(6)
Longitud total: 6 bytes		

El registro de anotaciones de nombramiento de una tabla o un esquema no contiene información sobre los nombres antiguos y nuevos de un objeto de tabla o

esquema. Separe los registros de anotaciones de inserción, actualización y supresión asociados a operaciones de tablas de catálogos del sistema, se generan cuando se realiza un renombramiento de una tabla o un esquema.

Registro de anotaciones de deshacer renombramiento de una tabla o un esquema

El registro de anotaciones de deshacer renombramiento de una tabla o un esquema se escribe cuando se retrotrae una modificación de nombre de una tabla o un esquema. Es el ID de función 123.

Tabla 132. Estructura del registro de deshacer renombramiento de una tabla o un esquema

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DMSLogRecordHeader	0(6)
Longitud total: 6 bytes		

El registro de anotaciones de nombramiento de una tabla o un esquema no contiene información sobre los nombres antiguos y nuevos de un objeto de tabla o esquema. Separe los registros de anotaciones de inserción, actualización y supresión asociados a operaciones de tablas de catálogos del sistema, se generan cuando se realiza un renombramiento de una tabla o un esquema.

Insertar varios registros, deshacer inserción de varios registros

Estos registros de anotaciones se graban cuando se insertan varias filas en la misma página de una tabla. Retrotraer la inserción de varios registros es un registro de anotaciones de compensación. Los ID de función son 167 y 168.

Tabla 133. Inserción de varias estructuras de registro

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de registro	DMSLogRecordHeader	0(6)
Relleno	char[]	6(2)
Número de registros	unsigned short	8(2)
Espacio libre	unsigned short	10(2)
Suma de longitudes de registro	unsigned short	12(2)
Longitud de parte variable	unsigned short	14(2)
Número de página de agrupación	sqluint32	16(4)
Registrar descripciones o retrotraer descripciones Consulte la Tabla 134 en la página 626 y la Tabla 135 en la página 626.	DBCLOB	20(variable)
Longitud total: 20 bytes más longitud de registro		

Tabla 134. Descripciones de registro (uno para cada registro)

Descripción	Tipo	Desplazamiento (Bytes)
RID	unsigned char[6]	0(6)
Desplazamiento de registro	unsigned short	6(2)
Cabecera y datos de registro	DBCLOB	8(variable)
Longitud total: 8 bytes más longitud de registro		

Tabla 135. Descripciones de retrotracción (una para cada registro)

Descripción	Tipo	Desplazamiento (Bytes)
RID	unsigned char[6]	0(6)
Desplazamiento de registro	unsigned short	6(2)
Longitud total: 8 bytes		

Para ver los detalles de la cabecera y los datos, consulte la descripción después de la Tabla 124 en la página 619.

Apéndice C. Visión general de la información técnica de DB2

La información técnica de DB2 está disponible a través de las herramientas y los métodos siguientes:

- *Centro de información de DB2*
  - Temas (Tareas, concepto y temas de consulta)
  - Ayuda para herramientas de DB2
  - Programas de ejemplo
  - Guías de aprendizaje
- Manuales de DB2
  - Archivos PDF (descargables)
  - Archivos PDF (desde el DVD con PDF de DB2)
  - Manuales en copia impresa
- Ayuda de línea de mandatos
  - Ayuda de mandatos
  - Ayuda de mensajes

Nota: Los temas del *Centro de información de DB2* se actualizan con más frecuencia que los manuales en PDF o impresos. Para obtener la información más actualizada, instale las actualizaciones de la documentación cuando estén disponibles, o consulte el *Centro de información de DB2* en ibm.com.

Puede acceder a información técnica adicional de DB2 como, por ejemplo, notas técnicas, documentos técnicos y publicaciones IBM Redbooks en línea, en el sitio ibm.com. Acceda al sitio de la biblioteca de software de gestión de información de DB2 en http://www.ibm.com/software/data/sw-library/.

Comentarios sobre la documentación

Agradecemos los comentarios sobre la documentación de DB2. Si tiene sugerencias sobre cómo podemos mejorar la documentación de DB2, envíe un correo electrónico a db2docs@ca.ibm.com. El personal encargado de la documentación de DB2 lee todos los comentarios de los usuarios, pero no puede responderlos directamente. Proporcione ejemplos específicos siempre que sea posible de manera que podamos comprender mejor sus problemas. Si realiza comentarios sobre un tema o archivo de ayuda determinado, incluya el título del tema y el URL.

No utilice esta dirección de correo electrónico para contactar con el Soporte al cliente de DB2. Si tiene un problema técnico de DB2 que no está tratado por la documentación, consulte al centro local de servicio técnico de IBM para obtener ayuda.

Si desea ayudar a IBM para que los productos IBM Information Management sean más fáciles de utilizar, obtenga el cuestionario de consumibilidad: http://www.ibm.com/software/data/info/consumability-survey/.

Biblioteca técnica de DB2 en copia impresa o en formato PDF

Las tablas siguientes describen la biblioteca de DB2 que está disponible en el Centro de publicaciones de IBM en www.ibm.com/shop/publications/order. Los manuales de DB2 Versión 9.5 en inglés en formato PDF y las versiones traducidas se pueden descargar del sitio www.ibm.com/support/docview.wss?rs=71 &uid=swg2700947.

Aunque las tablas identifican los manuales en copia impresa disponibles, puede que dichos manuales no estén disponibles en su país o región.

El número de documento se incrementa cada vez que se actualiza un manual. Asegúrese de que lee la versión más reciente de los manuales, tal como aparece a continuación:

Nota: El *Centro de información de DB2* se actualiza con más frecuencia que los manuales en PDF o impresos.

Tabla 136. Información técnica de DB2

Nombre	Número de documento	Copia impresa disponible	Última actualización
Consulta de las API administrativas	SC11-3505-02	Sí	Abril de 2009
Rutinas y vistas administrativas	SC11-3507-02	No	Abril de 2009
Call Level Interface Guide and Reference, Volume 1	SC23-5844-02	Sí	Abril de 2009
Call Level Interface Guide and Reference, Volume 2	SC23-5845-02	Sí	Abril de 2009
Consulta de mandatos	SC11-3506-02	Sí	Abril de 2009
Data Movement Utilities Guide and Reference	SC23-5847-02	Sí	Abril de 2009
Data Recovery and High Availability Guide and Reference	SC23-5848-02	Sí	Abril de 2009
Data Servers, Databases, and Database Objects Guide	SC23-5849-02	Sí	Abril de 2009
Database Security Guide	SC23-5850-02	Sí	Abril de 2009
Desarrollo de aplicaciones ADO.NET y OLE DB	SC11-3499-02	Sí	Abril de 2009
Desarrollo de aplicaciones de SQL incorporado	SC11-3500-02	Sí	Abril de 2009
Desarrollo de aplicaciones Java	SC11-3501-02	Sí	Abril de 2009
Desarrollo de aplicaciones Perl y PHP	SC11-3502-02	No	Abril de 2009
Desarrollo de rutinas definidas por el usuario (SQL y externas)	SC11-3503-02	Sí	Abril de 2009

Tabla 136. Información técnica de DB2 (continuación)

Nombre	Número de documento	Copia impresa disponible	Última actualización
Iniciación al desarrollo de aplicaciones de bases de datos	GC11-3504-02	Sí	Abril de 2009
Iniciación a la instalación y administración de DB2 en Linux y Windows	GC11-3511-02	Sí	Abril de 2009
Internationalization Guide	SC23-5858-02	Sí	Abril de 2009
Consulta de mensajes, Volumen 1	GI11-7823-01	No	Abril de 2009
Consulta de mensajes, Volumen 2	GI11-7824-01	No	Abril de 2009
Guía de migración	GC11-3510-02	Sí	Abril de 2009
Net Search Extender Guía de administración y del usuario	SC11-3615-02	Sí	Abril de 2009
Partitioning and Clustering Guide	SC23-5860-02	Sí	Abril de 2009
Query Patroller Administration and User's Guide	SC23-8507-01	Sí	Abril de 2009
Guía rápida para clientes de IBM Data Server	GC11-3513-02	No	Abril de 2009
Guía rápida para servidores DB2	GC11-3512-02	Sí	Abril de 2009
Spatial Extender y Geodetic Data Management Feature Guía del usuario y manual de consulta	SC11-3614-02	Sí	Abril de 2009
Consulta de SQL, Volumen 1	SC11-3508-02	Sí	Abril de 2009
Consulta de SQL, Volumen 2	SC11-3509-02	Sí	Abril de 2009
System Monitor Guide and Reference	SC23-5865-02	Sí	Abril de 2009
Guía de Text Search	SC11-3717-01	Sí	Abril de 2009
Troubleshooting Guide	GI11-7857-02	No	Abril de 2009
Tuning Database Performance	SC23-5867-02	Sí	Abril de 2009
Guía de aprendizaje de Visual Explain	SC11-3518-00	No	
Novedades	SC11-3517-02	Sí	Abril de 2009
Workload Manager Guide and Reference	SC23-5870-02	Sí	Abril de 2009
pureXML Guide	SC23-5871-02	Sí	Abril de 2009
XQuery Reference	SC23-5872-02	No	Abril de 2009

Tabla 137. Información técnica específica de DB2 Connect

Nombre	Número de documento	Copia impresa disponible	Última actualización
Guía rápida para DB2 Connect Personal Edition	GC11-3515-02	Sí	Abril de 2009
Guía rápida para servidores DB2 Connect	GC11-3516-02	Sí	Abril de 2009
Guía del usuario de DB2 Connect	SC11-3514-02	Sí	Abril de 2009

Tabla 138. Información técnica de Information Integration

Nombre	Número de documento	Copia impresa disponible	Última actualización
Information Integration: Administration Guide for Federated Systems	SC19-1020-01	Sí	Marzo de 2008
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-02	Sí	Marzo de 2008
Information Integration: Configuration Guide for Federated Data Sources	SC19-1034-01	No	
Information Integration: SQL Replication Guide and Reference	SC19-1030-01	Sí	Marzo de 2008
Information Integration: Introduction to Replication and Event Publishing	SC19-1028-01	Sí	Marzo de 2008

Pedido de manuales de DB2 en copia impresa

Si necesita manuales de DB2 en copia impresa, puede comprarlos en línea en varios países o regiones, pero no en todos. Siempre puede hacer pedidos de manuales de DB2 en copia impresa a través del representante local de IBM. Recuerde que algunas publicaciones en copia software del DVD *Documentación en PDF de DB2* no están disponibles en copia impresa. Por ejemplo, no está disponible la publicación *Consulta de mensajes de DB2* en copia impresa.

Las versiones impresas de muchas de las publicaciones de DB2 disponibles en el DVD de Documentación en PDF de DB2 se pueden solicitar a IBM por una cantidad. Dependiendo desde dónde realice el pedido, podrá solicitar manuales en línea, desde el Centro de publicaciones de IBM. Si la realización de pedidos en línea no está disponible en su país o región, siempre puede hacer pedidos de manuales de DB2 en copia impresa al representante local de IBM. Tenga en cuenta que no todas las publicaciones del DVD de Documentación en PDF de DB2 están disponibles en copia impresa.

Nota: La documentación más actualizada y completa de DB2 se conserva en el Centro de información de DB2 en http://publib.boulder.ibm.com/infocenter/db2luw/v9r5.

Para hacer pedidos de manuales de DB2 en copia impresa:

- Para averiguar si puede hacer pedidos de manuales de DB2 en copia impresa en línea en su país o región, consulte el Centro de publicaciones de IBM en el sitio http://www.ibm.com/shop/publications/order. Debe seleccionar un país, región o idioma para poder acceder a la información sobre pedidos de publicaciones y, a continuación, seguir las instrucciones sobre pedidos para su localidad.
- Para hacer pedidos de manuales de DB2 en copia impresa a través del representante local de IBM:
  1. Localice la información de contacto de su representante local desde uno de los siguientes sitios Web:
     - El directorio de IBM de contactos en todo el mundo en el sitio www.ibm.com/planetwide
     - El sitio Web de publicaciones de IBM en el sitio http://www.ibm.com/shop/publications/order. Tendrá que seleccionar su país, región o idioma para acceder a la página de presentación de las publicaciones apropiadas para su localidad. Desde esta página, siga el enlace "Acerca de este sitio".
  2. Cuando llame, indique que desea hacer un pedido de una publicación de DB2.
  3. Proporcione al representante los títulos y números de documento de las publicaciones que desee solicitar. Si desea consultar los títulos y los números de documento, consulte el apartado "Biblioteca técnica de DB2 en copia impresa o en formato PDF" en la página 628.

Visualización de la ayuda para estados de SQL desde el procesador de línea de mandatos

DB2 devuelve un valor de SQLSTATE para las condiciones que pueden ser el resultado de una sentencia de SQL. La ayuda de SQLSTATE explica los significados de los estados de SQL y los códigos de las clases de estados de SQL.

Para invocar la ayuda para estados de SQL, abra el procesador de línea de mandatos y entre:

```
? sqlstate o ? código de clase
```

donde *sqlstate* representa un estado de SQL válido de cinco dígitos y *código de clase* representa los dos primeros dígitos del estado de SQL.
Por ejemplo, `? 08003` visualiza la ayuda para el estado de SQL 08003, y `? 08` visualiza la ayuda para el código de clase 08.

Acceso a diferentes versiones del Centro de información de DB2

Para los temas de DB2 Versión 9.5, el URL del Centro de información de DB2 es http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/

Para los temas de DB2 Versión 9, el URL del Centro de información de DB2 es http://publib.boulder.ibm.com/infocenter/db2luw/v9/

Para los temas de DB2 Versión 8, vaya al URL del Centro de información de la Versión 8 en el sitio: http://publib.boulder.ibm.com/infocenter/db2luw/v8/

Visualización de temas en su idioma preferido en el Centro de información de DB2

El Centro de información de DB2 intenta visualizar los temas en el idioma especificado en las preferencias del navegador. Si un tema no se ha traducido al idioma preferido, el Centro de información de DB2 visualiza dicho tema en inglés.

- Para visualizar temas en su idioma preferido en el navegador Internet Explorer:
  1. En Internet Explorer, pulse en el botón **Herramientas** —> **Opciones de Internet** —> **Idiomas...**. Se abrirá la ventana Preferencias de idioma.
  2. Asegúrese de que su idioma preferido esté especificado como la primera entrada de la lista de idiomas.
     - Para añadir un nuevo idioma a la lista, pulse el botón **Agregar...**.

       Nota: La adición de un idioma no garantiza que el sistema tenga los fonts necesarios para visualizar los temas en el idioma preferido.
     - Para mover un idioma hacia el principio de la lista, seleccione el idioma y pulse el botón **Subir** hasta que el idioma esté en primer lugar en la lista de idiomas.
  3. Borre la antememoria del navegador y, a continuación, renueve la página para visualizar el Centro de información de DB2 en su idioma preferido.
- Para visualizar temas en su idioma preferido en un navegador Firefox o Mozilla:
  1. Seleccione el botón en la sección **Idiomas** del diálogo **Herramientas** —> **Opciones** —> **Avanzado**. Se visualizará el panel Idiomas en la ventana Preferencias.
  2. Asegúrese de que su idioma preferido esté especificado como la primera entrada de la lista de idiomas.
     - Para añadir un nuevo idioma a la lista, pulse el botón **Añadir...** a fin de seleccionar un idioma en la ventana Añadir idiomas.
     - Para mover un idioma hacia el principio de la lista, seleccione el idioma y pulse el botón **Subir** hasta que el idioma esté en primer lugar en la lista de idiomas.
  3. Borre la antememoria del navegador y, a continuación, renueve la página para visualizar el Centro de información de DB2 en su idioma preferido.

En algunas combinaciones de navegador y sistema operativo, puede que también tenga que cambiar los valores regionales del sistema operativo al entorno local y al idioma de su elección.

Actualización del Centro de información de DB2 instalado en el sistema o en el servidor de intranet

Si ha instalado localmente el Centro de información de DB2, puede obtener las actualizaciones de la documentación de IBM e instalarlas.

Para actualizar el *Centro de información de DB2* instalado localmente es preciso que:

1. Detenga el *Centro de información de DB2* en el sistema, y reinicie el Centro de información en modalidad autónoma. La ejecución del Centro de información en modalidad autónoma impide que otros usuarios de la red accedan al Centro de información y permite al usuario aplicar las actualizaciones. Los Centros de información no administrativos y no root de *DB2* se ejecutan siempre en modalidad autónoma.

2. Utilice la característica Actualizar para ver qué actualizaciones están disponibles. Si hay actualizaciones que desee instalar, puede utilizar la característica Actualizar para obtenerlos e instalarlos.

   Nota: Si su entorno requiere la instalación de actualizaciones del *Centro de información de DB2* en una máquina no conectada a Internet, debe duplicar el sitio de actualizaciones en un sistema de archivos local utilizando una máquina que esté conectada a Internet y tenga instalado el *Centro de información de DB2*. Si muchos usuarios en la red van a instalar las actualizaciones de la documentación, puede reducir el tiempo necesario para realizar las actualizaciones duplicando también el sitio de actualizaciones localmente y creando un proxy para el sitio de actualizaciones.
   Si hay paquetes de actualización disponibles, utilice la característica Actualizar para obtener los paquetes. Sin embargo, la característica Actualizar sólo está disponible en modalidad autónoma.
3. Detenga el Centro de información autónomo y reinicie el *Centro de información de DB2* en su equipo.

Nota: En Windows Vista, los mandatos listados más abajo se deben ejecutar como administrador. Para iniciar un indicador de mandatos o una herramienta gráfica con privilegios de administrador completos, pulse con el botón derecho del ratón el atajo y, a continuación, seleccione **Ejecutar como administrador**.

Para actualizar el Centro de información de *Centro de información de DB2* instalado en el sistema o en el servidor de Intranet:

1. Detenga el *Centro de información de DB2*.
   - En Windows, pulse **Inicio → Panel de control → Herramientas administrativas → Servicios**. A continuación, pulse con el botón derecho del ratón en el servicio **Centro de información de DB2** y seleccione **Detener**.
   - En Linux, especifique el mandato siguiente:
     ```
     /etc/init.d/db2icdv95 stop
     ```
2. Inicie el Centro de información en modalidad autónoma.
   - En Windows:
     a. Abra una ventana de mandatos.
     b. Navegue hasta la vía de acceso en la que está instalado el Centro de información. De manera predeterminada, el *Centro de información de DB2* está instalado en el directorio *Archivos_de_programa*\IBM\DB2 Information Center\Version 9.5, donde *Archivos_de_programa* representa la ubicación del directorio Archivos de programa.
     c. Navegue desde el directorio de instalación al directorio doc\bin.
     d. Ejecute el archivo help_start.bat:
        ```
        help_start.bat
        ```
   - En Linux:
     a. Navegue hasta la vía de acceso en la que está instalado el Centro de información. De forma predeterminada, el *Centro de información de DB2* se instala en el directorio /opt/ibm/db2ic/V9.5.
     b. Navegue desde el directorio de instalación al directorio doc/bin.
     c. Ejecute el script help_start:
        ```
        help_start
        ```

   Se inicia el navegador Web por omisión de los sistemas para visualizar el Centro de información autónomo.

3. Pulse en el botón **Actualizar** (). En la derecha del panel del Centro de información, pulse en **Buscar actualizaciones**. Se visualiza una lista de actualizaciones para la documentación existente.
4. Para iniciar el proceso de instalación, compruebe las selecciones que desee instalar y, a continuación, pulse **Instalar actualizaciones**.
5. Cuando finalice el proceso de instalación, pulse **Finalizar**.
6. Detenga el Centro de información autónomo:
   - En Windows, navegue hasta el directorio doc\bin del directorio de instalación y ejecute el archivo help_end.bat:
     ```
     help_end.bat
     ```
     Nota: El archivo help_end de proceso por lotes contiene los mandatos necesarios para concluir sin peligro los procesos que se iniciaron mediante el archivo help_start de proceso por lotes. No utilice `Control-C` ni ningún otro método para concluir help_start.bat.
   - En Linux, navegue hasta el directorio de instalación doc/bin y ejecute el script help_end:
     ```
     help_end
     ```
     Nota: El script help_end contiene los mandatos necesarios para concluir sin peligro los procesos que se iniciaron mediante el script help_start. No utilice ningún otro método para concluir el script help_start.
7. Reinicie el *Centro de información de DB2*:
   - En Windows, pulse **Inicio** → **Panel de control** → **Herramientas administrativas** → **Servicios**. A continuación, pulse con el botón derecho del ratón en el servicio **Centro de información de DB2** y seleccione **Iniciar**.
   - En Linux, especifique el mandato siguiente:
     ```
     /etc/init.d/db2icdv95 start
     ```

El *Centro de información de DB2* actualizado visualiza los temas nuevos y actualizados.

Guías de aprendizaje de DB2

Las guías de aprendizaje de DB2 le ayudan a conocer diversos aspectos de productos DB2. Se proporcionan instrucciones paso a paso a través de lecciones.

Antes de comenzar

Puede ver la versión XHTML de la guía de aprendizaje desde el Centro de información en el sitio http://publib.boulder.ibm.com/infocenter/db2help/.

Algunas lecciones utilizan datos o código de ejemplo. Consulte la guía de aprendizaje para obtener una descripción de los prerrequisitos para las tareas específicas.

Guías de aprendizaje de DB2

Para ver la guía de aprendizaje, pulse el título.

"pureXML" en *pureXML Guide*
: Configure una base de datos DB2 para almacenar datos XML y realizar operaciones básicas con el almacén de datos XML nativos.

"Visual Explain" en *Guía de aprendizaje de Visual Explain*
Analizar, optimizar y ajustar sentencias de SQL para obtener un mejor rendimiento al utilizar Visual Explain.

Información de resolución de problemas de DB2

Existe una gran variedad de información para la resolución y determinación de problemas para ayudarle en la utilización de productos de base de datos DB2.

Documentación de DB2
Puede encontrar información sobre la resolución de problemas en la publicación DB2 Troubleshooting Guide o en la sección Conceptos fundamentales sobre bases de datos del Centro de información de DB2. En ellas encontrará información sobre cómo aislar e identificar problemas utilizando herramientas y programas de utilidad de diagnóstico de DB2, soluciones a algunos de los problemas más habituales y otros consejos sobre cómo solucionar problemas que podría encontrar en los productos DB2.

Sitio web de soporte técnico de DB2
Consulte el sitio Web de soporte técnico de DB2 si tiene problemas y desea obtener ayuda para encontrar las causas y soluciones posibles. El sitio de soporte técnico tiene enlaces a las publicaciones más recientes de DB2, notas técnicas, Informes autorizados de análisis del programa (APAR o arreglos de defectos), fixpacks y otros recursos. Puede buscar en esta base de conocimiento para encontrar posibles soluciones a los problemas.

Acceda al sitio Web de soporte técnico de DB2 en la dirección http://www.ibm.com/software/data/db2/support/db2_9/

Términos y condiciones

Los permisos para utilizar estas publicaciones se otorgan sujetos a los siguientes términos y condiciones.

Uso personal: Puede reproducir estas publicaciones para su uso personal, no comercial, siempre y cuando se mantengan los avisos sobre la propiedad. No puede distribuir, visualizar o realizar trabajos derivados de estas publicaciones, o de partes de las mismas, sin el consentimiento expreso de IBM.

Uso comercial: Puede reproducir, distribuir y visualizar estas publicaciones únicamente dentro de su empresa, siempre y cuando se mantengan todos los avisos sobre la propiedad. No puede realizar trabajos derivativos de estas publicaciones, ni reproducirlas, distribuirlas o visualizarlas, ni de partes de las mismas fuera de su empresa, sin el consentimiento expreso de IBM.

Excepto lo expresamente concedido en este permiso, no se conceden otros permisos, licencias ni derechos, explícitos o implícitos, sobre las publicaciones ni sobre ninguna información, datos, software u otra propiedad intelectual contenida en el mismo.

IBM se reserva el derecho de retirar los permisos aquí concedidos cuando, a su discreción, el uso de las publicaciones sea en detrimento de su interés o cuando, según determine IBM, las instrucciones anteriores no se cumplan correctamente.

No puede descargar, exportar ni volver a exportar esta información excepto en el caso de cumplimiento total con todas las leyes y regulaciones vigentes, incluyendo todas las leyes y regulaciones sobre exportación de los Estados Unidos.

IBM NO GARANTIZA EL CONTENIDO DE ESTAS PUBLICACIONES. LAS PUBLICACIONES SE PROPORCIONAN ″TAL CUAL″ Y SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUYENDO PERO SIN LIMITARSE A LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN, NO VULNERACIÓN E IDONEIDAD PARA UN FIN DETERMINADO.

Apéndice D. Avisos

Esta información ha sido desarrollada para productos y servicios que se ofrecen en Estados Unidos de América

Es posible que IBM no comercialice en otros países algunos productos, servicios o características descritos en este manual. Consulte al representante local de IBM para obtener información sobre los productos y servicios que actualmente pueden adquirirse en su zona. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se pueda utilizar dicho producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ninguno de los derechos de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patentes en tramitación que afecten al tema tratado en este documento. La posesión de este documento no confiere ninguna licencia sobre dichas patentes. Puede realizar consultas sobre licencias escribiendo a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
EE.UU.

Para realizar consultas sobre licencias referentes a información de doble byte (DBCS), puede ponerse en contacto con el Departamento de Propiedad Intelectual de IBM de su país/región o escribir a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokio 106, Japón

El párrafo siguiente no es aplicable al Reino Unido ni a ningún país/región en donde tales disposiciones sean incompatibles con la legislación local:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL", SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN DE DERECHOS, COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN DETERMINADO. Algunos estados no permiten la exclusión de garantías expresas o implícitas en determinadas transacciones, por lo que es posible que esta declaración no sea aplicable en su caso.

Esta publicación puede contener inexactitudes técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; dichos cambios se incorporarán a las nuevas ediciones de la publicación. IBM puede efectuar, en cualquier momento y sin previo aviso, mejoras y cambios en los productos y programas descritos en esta publicación.

Este documento puede proporcionar enlaces o referencias a sitios y recursos que no son de IBM. IBM no representa, no da garantías, ni se compromete con los recursos de terceros ni con los recursos que no son de IBM a los cuales se puede hacer referencia, acceder desde o enlazarse con desde este documento. Un enlace a un sitio que no es de IBM no implica que IBM apruebe el contenido o la utilización de dicho sitio Web o a su propietario. Además, IBM no forma parte ni es responsable de ninguna transacción que el usuario pueda realizar con terceros, aún cuando llegue a conocerlos (o utilice un enlace a ellos) desde un sitio de IBM. De acuerdo a esto, el usuario reconoce y acepta que IBM no es responsable de la disponibilidad de dichos recursos o sitios externos ni tampoco es responsable de ningún contenido, servicio, producto u otros materiales que estén o se encuentren disponibles desde dichos sitios o recursos. Cualquier software que proporcionen terceras partes, estarán sujetos a los términos y condiciones de licencia que acompañen al software.

IBM puede utilizar o distribuir cualquier información que se le facilite de la manera que considere adecuada, sin contraer por ello ninguna obligación con el remitente.

Los licenciatarios de este programa que deseen obtener información sobre él con el fin de habilitar: (i) el intercambio de información entre programas creados de forma independiente y otros programas (incluido éste) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADÁ

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluido en algunos casos el pago de una tarifa.

El programa bajo licencia descrito en este documento y todo el material bajo licencia asociado a él, los proporciona IBM según los términos del Acuerdo de Cliente de IBM, el Acuerdo Internacional de Programas Bajo Licencia de IBM o cualquier acuerdo equivalente entre el usuario e IBM.

Los datos de rendimiento contenidos en este documento se obtuvieron en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse realizado en sistemas experimentales y no es seguro que estas mediciones sean las mismas en los sistemas disponibles comercialmente. Además, algunas mediciones pueden haberse calculado mediante extrapolación. Los resultados reales pueden variar. Los usuarios del presente manual deben verificar los datos aplicables para su entorno específico.

La información referente a productos que no son de IBM se ha obtenido de los proveedores de esos productos, de sus anuncios publicados o de otras fuentes disponibles públicamente. IBM no ha probado esos productos y no puede confirmar la exactitud del rendimiento, la compatibilidad ni ninguna otra afirmación referente a productos que no son de IBM. Las preguntas sobre las prestaciones de productos que no son de IBM deben dirigirse a los proveedores de esos productos.

Todas las declaraciones de intenciones de IBM están sujetas a cambio o cancelación sin previo aviso, y sólo representan objetivos.

Este manual puede contener ejemplos de datos e informes que se utilizan en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con nombres y direcciones utilizados por una empresa real es totalmente fortuita.

LICENCIA DE COPYRIGHT:

Este manual puede contener programas de aplicaciones de ejemplo escritos en lenguaje fuente, que muestran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo como desee, sin pago alguno a IBM con la intención de desarrollar, utilizar, comercializar o distribuir programas de aplicaciones de acuerdo con la interfaz de programación de aplicaciones correspondiente a la plataforma operativa para la que están escritos los programas de ejemplo. Estos ejemplos no se han probado exhaustivamente bajo todas las condiciones. Por lo tanto, IBM no puede asegurar ni implicar la fiabilidad, utilidad o función de estos programas.

Cada copia o parte de estos programas de ejemplo o cualquier trabajo derivado debe incluir una nota de copyright como la siguiente:

© (*nombre de la empresa*) (*año*). Partes de este código proceden de programas de ejemplo de IBM Corp. © Copyright IBM Corp. *_entre el o los años_*. Reservados todos los derechos.

Marcas registradas

IBM, el logotipo de IBM e ibm.com son marcas registradas de International Business Machines Corp., que se han registrado en muchas otras jurisdicciones. Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas. Puede consultarse en línea una lista actualizada de las marcas registradas de IBM en la sección Copyright and trademark information de la web www.ibm.com/legal/copytrade.shtml.

Los siguientes términos son marcas registradas de otras empresas.

- Linux es una marca registrada de Linus Torvalds en los Estados Unidos y/o en otros países.
- Java y todas las marcas registradas basadas en Java son marcas registradas de Sun Microsystems, Inc. en los Estados Unidos y/o en otros países.
- UNIX es una marca registrada de The Open Group en los Estados Unidos y/o en otros países.
- Intel, el logotipo de Intel, Intel Inside, el logotipo de Intel Inside, Intel Centrino, el logotipo de Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium y Pentium son marcas registradas de Intel Corporation o de sus empresas subsidiarias en Estados Unidos y/o en otros países. Información sobre marcas registradas de Intel
- Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en los Estados Unidos y/o en otros países.

Otros nombres de empresas, productos o servicios, pueden ser marcas registradas o marcas de servicio de otras empresas.

Índice

A

B

C

D

E

F

G

I

K

L

M

N

O

P

Q

R

S

T

U

V

Z

SC11-3505-02

Spine information:

DB2 Versión 9.5 para Linux, UNIX y Windows

Consulta de las API administrativas