

IBM DB2 10.1
para Linux, UNIX y Windows

Consulta de las API administrativas



IBM DB2 10.1
para Linux, UNIX y Windows

Consulta de las API administrativas



Nota

Antes de utilizar esta información y el producto al que da soporte, lea la información general contenida en el apartado Apéndice D, "Avisos", en la página 783.

Nota de edición

Este manual es la traducción del original en inglés *IBM DB2 10.1 for Linux, UNIX, and Windows Administrative API Reference* (SC27-3864-00).

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 solicitar publicaciones en línea, vaya a IBM Publications Center en <http://www.ibm.com/shop/publications/order>
- Para encontrar al representante local de IBM que le corresponde, vaya a la sección Worldwide Contacts de IBM Directory en <http://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, está otorgando a IBM el derecho no exclusivo de utilizar o distribuir la información de cualquier forma que considere adecuada sin incurrir por ello a ninguna obligación para con usted.

© Copyright IBM Corporation 1993, 2012.

Contenido

Acerca de este manual ix

Quién debe utilizar este manual ix

Cómo está estructurado este manual ix

Convenios de resaltado x

Capítulo 1. Las API de DB2 1

Capítulo 2. Cambios en las API y estructuras de datos. 23

Capítulo 3. Cambios en el número de secuencia de anotaciones cronológicas que afectan al comportamiento de las API y las aplicaciones 27

Capítulo 4. Cómo están organizadas las descripciones de las API 31

Archivos de inclusión para aplicaciones de las API de DB2 35

Capítulo 5. API administrativas 37

db2AddContact - Añadir un contacto al que se pueden enviar mensajes de notificación 38

db2AddContactGroup - Añadir un grupo de contactos al que se pueden enviar mensajes de notificación 40

db2AddSnapshotRequest - Añadir una petición de instantánea 42

db2AdminMsgWrite - Grabar mensajes de anotación cronológica para la función de administración y duplicación 44

db2ArchiveLog - Archivar el archivo de anotaciones cronológicas activo 46

db2AutoConfig - Acceder al Asesor de configuración. 49

db2AutoConfigFreeMemory - Liberar la memoria asignada por la API db2AutoConfig 53

db2Backup - Hacer copia de seguridad de una base de datos o un espacio de tablas. 54

db2CfgGet - Obtener los parámetros de configuración del gestor de bases de datos o de la base de datos. 65

db2CfgSet - Definir los parámetros de configuración del gestor de bases de datos o de la base de datos . 68

db2ConvMonStream - Convertir la corriente de supervisor en el formato anterior a la versión 6 . . 73

db2DatabasePing - Sondar la base de datos para probar el tiempo de respuesta de la red 76

db2DatabaseQuiesce - Inmovilizar la base de datos 78

db2DatabaseRestart - Reiniciar base de datos . . . 80

db2DatabaseUnquiesce - Movilizar base de datos. . 83

db2DatabaseUpgrade - Actualizar la versión anterior de la base de datos DB2 al release actual 85

db2DbDirCloseScan - Finalizar una exploración del directorio de bases de datos locales o del sistema. . 87

db2DbDirGetNextEntry - Obtener la entrada siguiente del directorio de bases de datos locales o del sistema 88

db2DbDirOpenScan - Iniciar una exploración del directorio de bases de datos locales o del sistema. . 92

db2DropContact - Eliminar un contacto de la lista de contactos a los que se pueden enviar mensajes de notificación 94

db2DropContactGroup - Eliminar un grupo de contactos de la lista de contactos a los que se pueden enviar mensajes de notificación 95

db2Export - Exportar datos de una base de datos. . 96

db2GetAlertCfg - Obtener los valores de configuración de alertas para los indicadores de salud 103

db2GetAlertCfgFree - Liberar la memoria asignada por la API db2GetAlertCfg 108

db2GetContactGroup - Obtener la lista de contactos de un solo grupo de contactos al que se puedan enviar mensajes de notificación 109

db2GetContactGroups - Obtener la lista de grupos de contactos a los que se puedan enviar mensajes de notificación 111

db2GetContacts - Obtener la lista de contactos a los que se pueden enviar mensajes de notificación . . 113

db2GetDistMap - Obtener correlación de distribución 115

db2GetHealthNotificationList - Obtener la lista de contactos a los que se puedan enviar notificaciones de alerta de salud 117

db2GetRecommendations - Obtener recomendaciones para resolver un indicador de salud en estado de alerta 119

db2GetRecommendationsFree - Liberar la memoria asignada por la API db2GetRecommendations . . 122

db2GetSnapshot - Obtener una instantánea del estado operacional del gestor de bases de datos . 123

db2GetSnapshotSize - Calcular el tamaño del almacenamiento intermedio de salida necesario para la API db2GetSnapshot 127

db2GetSyncSession - Obtener un identificador de sesión de sincronización de satélites 130

db2HADRStart - Iniciar operaciones de HADR (high availability disaster recovery) 131

db2HADRStop - Detener operaciones de HADR (high availability disaster recovery) 133

db2HADRTakeover - Dar instrucciones a una base de datos para que se convierta en la base de datos primaria de HADR (high availability disaster recovery) 135

db2HistoryCloseScan - Finalizar la exploración de los registros del histórico de la base de datos. . . 138

db2HistoryGetEntry - Obtener la entrada siguiente de los registros del histórico de la base de datos. . 139

db2HistoryOpenScan - Iniciar una exploración de los registros del histórico de la base de datos . . .	141	db2Runstats - Actualizar estadísticas para tablas e índices	307
db2HistoryUpdate - Actualizar una entrada de los registros del histórico de la base de datos . . .	145	db2SelectDB2Copy - Seleccionar la copia de DB2 que la aplicación utiliza	320
db2Import - Importar datos a una tabla, jerarquía, apodo o vista	149	db2SetSyncSession - Establecer sesión de sincronización de satélites	322
db2Inspect - Inspeccionar la base de datos para comprobar la integridad de la arquitectura . . .	164	db2SetWriteForDB - Suspender o reanudar las grabaciones de E/S para la base de datos	323
db2InstanceQuiesce - Inmovilizar instancia . . .	172	db2SpmListIndTrans - Listar transacciones dudosas SPM	325
db2InstanceStart - Iniciar instancia	175	db2SyncSatellite - Iniciar sincronización de satélites	329
db2InstanceStop - Detener instancia	181	db2SyncSatelliteStop - Pausar sincronización de satélites	330
db2InstanceUnquiesce - Movilizar instancia . . .	186	db2SyncSatelliteTest - Probar si se puede sincronizar un satélite	331
db2LdapCatalogDatabase - Registrar la base de datos en el servidor LDAP	188	db2UpdateAlertCfg - Actualizar los valores de configuración de alertas para los indicadores de salud	332
db2LdapCatalogNode - Proporcionar un alias para el nombre de nodo en el servidor LDAP	191	db2UpdateAlternateServerForDB - Actualizar el servidor alternativo para un alias de base de datos en el directorio de bases de datos del sistema . .	338
db2LdapDeregister - Desregistrar el servidor DB2 y las bases de datos catalogadas del servidor LDAP .	193	db2UpdateContact - Actualizar los atributos de un contacto	340
db2LdapRegister - Registrar el servidor DB2 en el servidor LDAP	194	db2UpdateContactGroup - Actualizar los atributos de un grupo de contactos	342
db2LdapUncatalogDatabase - Desregistrar base de datos del servidor LDAP	198	db2UpdateHealthNotificationList - Actualizar la lista de contactos a los que se puedan enviar notificaciones de alerta de salud	344
db2LdapUncatalogNode - Suprimir alias para nombre de nodo del servidor LDAP.	199	db2UtilityControl - Establecer el nivel de prioridad de los programas de utilidad en ejecución. . . .	346
db2LdapUpdate - Actualizar los atributos del servidor DB2 en el servidor LDAP	200	sqlabndx - Programa de aplicación de vinculación para crear un paquete	348
db2LdapUpdateAlternateServerForDB - Actualizar el servidor alternativo de la base de datos en el servidor LDAP.	203	sqlaintp - Obtener mensaje de error	351
db2Load - Cargar datos en una tabla	205	sqlaprep - Precompilar programa de aplicación . .	353
db2LoadQuery - Obtener el estado de una operación de carga	228	sqlarbn - Volver a vincular paquete	356
db2MonitorSwitches - Obtener o actualizar los valores de los conmutadores del supervisor . . .	236	sqlbctq - Cerrar una consulta de contenedor de espacio de tablas	359
db2Prune - Suprimir las entradas del archivo histórico o archivos de anotaciones cronológicas de la vía de acceso de anotación cronológica activa .	239	sqlbctsq - Cerrar una consulta de espacio de tablas	360
db2QuerySatelliteProgress - Obtener el estado de una sesión de sincronización de satélites	242	sqlbftcq - Captar los datos de la consulta para filas de un contenedor de espacio de tablas	361
db2ReadLog - Leer registros de anotaciones cronológicas	244	sqlbftpq - Captar los datos de la consulta para filas de un espacio de tablas	363
db2ReadLogNoConn - Leer las anotaciones cronológicas de la base de datos sin una conexión de base de datos	251	sqlbgtss - Obtener estadísticas de utilización del espacio de tablas	365
db2ReadLogNoConnInit - Inicializar la lectura de las anotaciones cronológicas de la base de datos sin una conexión de base de datos	255	sqlbmtsq - Obtener los datos de la consulta para todos los espacios de tablas	367
db2ReadLogNoConnTerm - Terminar la lectura de las anotaciones cronológicas de la base de datos sin una conexión de base de datos	258	sqlbotcq - Abrir una consulta de contenedor de espacio de tablas	369
db2Recover - Restaurar y avanzar una base de datos	259	sqlbotsq - Abrir una consulta de espacio de tablas	371
db2Reorg - Reorganizar un índice o una tabla . .	265	sqlbstpq - Obtener información sobre un espacio de tablas individual	373
db2ResetAlertCfg - Restablecer la configuración de alertas de los indicadores de salud	275	sqlbstsc - Definir contenedores de espacios de tablas	375
db2ResetMonitor - Restaurar los datos del supervisor del sistema de base de datos	277	sqlbtcq - Obtener los datos de la consulta para todos los contenedores de espacios de tablas . . .	378
db2Restore - Restaurar una base de datos o un espacio de tablas	280	sqlcspqy - Listar transacciones dudosas DRDA . .	380
db2Rollforward - Avanzar una base de datos . . .	296	sqlc_activate_db - Activar base de datos	381
		sqlc_deactivate_db - Desactivar base de datos . .	384
		sqlleadn - Añadir una partición de base de datos al entorno de base de datos particionado	387

sqlcatcp - Conectar a instancia y cambiar contraseña	390
sqlcatin - Conectar a instancia	393
sqlcadb - Catalogar una base de datos del directorio de bases de datos del sistema	396
sqlcran - Crear una base de datos en un servidor de particiones de base de datos	402
sqlcrea - Crear una base de datos	404
sqlctnd - Catalogar una entrada en el directorio de nodos	412
sqledcgd - Cambiar un comentario de base de datos en el directorio de bases de datos locales o del sistema	415
sqledpan - Descartar una base de datos de un servidor de particiones de base de datos	418
sqledrpd - Descartar base de datos	420
sqledrpn - Comprobar si se puede descartar un servidor de particiones de base de datos	422
sqledtin - Desconectar de instancia	424
sqlfmem - Liberar la memoria asignada por las API sqlbctq y sqlbmtsq	425
sqlfrce - Desconectar usuarios y aplicaciones del sistema	426
sqlgdad - Catalogar una base de datos en el directorio de DCS (Database Connection Services)	429
sqlgdcl - Finalizar una exploración del directorio de DCS (Database Connection Services)	431
sqlgdcl - Descatalogar una base de datos del directorio de DCS (Database Connection Services)	432
sqlgdge - Obtener una entrada específica del directorio de DCS (Database Connection Services)	434
sqlgdgt - Obtener entradas del directorio de DCS (servicios de conexión de base de datos)	436
sqlgdcl - Iniciar una exploración del directorio de DCS (Database Connection Services)	438
sqlgins - Obtener instancia actual	439
sqlintr - Interrumpir peticiones de aplicaciones	440
sqlisig - Instalar manejador de señales	442
sqlmgdb - Actualizar la versión anterior de la base de datos DB2 a la versión actual	443
sqlncl - Finalizar una exploración del directorio de nodos	445
sqlngne - Obtener la entrada siguiente del directorio de nodos	446
sqlnops - Iniciar una exploración del directorio de nodos	448
sqlqryc - Consultar valores de conexión del cliente	450
sqlqryi - Consultar información sobre el cliente	452
sqlsact - Establecer serie de contabilidad	454
sqlsdeg - Establecer el nivel o grado máximo de paralelismo intrapartición para la ejecución de sentencias de SQL	455
sqlsetc - Establecer valores de conexión del cliente	457
sqlseti - Establecer información sobre el cliente	460
sqluncd - Descatalogar una base de datos del directorio de bases de datos del sistema	463
sqluncn - Descatalogar una entrada del directorio de nodos	465
sqlgaddr - Obtener la dirección de una variable	467
sqlgdref - Eliminar la referencia de una dirección	468

sqlgmcpy - Copiar datos de un área de memoria a otra	469
sqlogstt - Obtener el mensaje de SQLSTATE	470
sqludrtd: redistribuir datos a través de un grupo de particiones de base de datos	472
sqlugrpn - Obtener el número de servidor de particiones de base de datos para una fila	476
qlugtpi - Obtener información de distribución de tablas	479
sqluvqdp - Inmovilizar espacios de tablas para una tabla	481

Capítulo 6. Llamada a las API de DB2 en REXX	485
Cambiar el nivel de aislamiento	487

Capítulo 7. Las API de gestión de transacciones dudosas	489
db2XaGetInfo - Obtener información para un gestor de recursos	491
db2XaListIndTrans - Listar transacciones dudosas	492
sqlxhfrg - Olvidar estado de transacción	497
sqlxphcm - Confirmar una transacción dudosa	498
sqlxphrl - Retrotraer una transacción dudosa	499

Capítulo 8. Aplicaciones por hebras con acceso simultáneo	501
sqlAttachToCtx - Conectar a contexto	502
sqlBeginCtx - Crear y conectar a un contexto de aplicación	503
sqlDetachFromCtx - Desconectar de contexto	504
sqlEndCtx - Desconectar y liberar la memoria asociada a un contexto de aplicación	505
sqlGetCurrentCtx - Obtener contexto actual	507
sqlInterruptCtx - Interrumpir contexto	508
sqlSetTypeCtx - Definir el tipo de contexto de aplicación	509

Capítulo 9. Plugins del sistema de base de datos de DB2 para personalizar la gestión de bases de datos	511
Habilitación de los plugins	511
Despliegue de un plugin de recuperación de grupos	511
Despliegue de un plugin de ID de usuario/contraseña	512
Despliegue de un plugin de GSS-API	513
Despliegue de un plugin de Kerberos	515
Escritura de plugins de seguridad	516
Cómo carga DB2 los plugins de seguridad	516
Restricciones en el desarrollo de bibliotecas de plugins de seguridad	518
Restricciones para plugins de seguridad	520
Códigos de retorno para plugins de seguridad	523
Manejo de mensajes de error para los plugins de seguridad	526
Secuencias de llamada para las API de plugins de seguridad	527

Plugins de seguridad	530
Ubicaciones de las bibliotecas de plugins de seguridad	535
Convenios de denominación para los plugins de seguridad	536
Soporte de plugin de seguridad para los ID de usuario de dos componentes	537
Mantenimiento de las versiones de las API de plugins de seguridad	539
Consideraciones sobre los sistemas de 32 y 64 bits para los plugins de seguridad	539
Determinación de problemas para plugins de seguridad	540
Las API del plugin de seguridad	541
API para plugins de recuperación de grupos	544
API db2secDoesGroupExist - Comprobar si existe el grupo	546
API db2secFreeErrorMsg - Liberar la memoria de mensajes de error	547
API db2secFreeGroupListMemory - Liberar memoria de lista de grupos	548
API db2secGetGroupsForUser - Obtener la lista de grupos del usuario	549
API db2secGroupPluginInit - Inicializar plugin de grupo	553
db2secPluginTerm - Liberar los recursos de plugin de grupo	555
Las API de los plugins de autenticación por ID de usuario/contraseña	556
API db2secClientAuthPluginInit - Inicializar el plugin de autenticación del cliente	562
API db2secClientAuthPluginTerm - Liberar los recursos de plugin de autenticación de cliente	564
db2secDoesAuthIDExist - Comprobar si existe el ID de autenticación	565
API db2secFreeInitInfo - Liberar los recursos retenidos por la API db2secGenerateInitialCred	566
API db2secFreeToken - Liberar memoria retenida por símbolo (token)	567
API db2secGenerateInitialCred - Generar credenciales iniciales	568
API db2secGetAuthIDs - Obtener los ID de autenticación	570
API db2secGetDefaultLoginContext - Obtener contexto de conexión por omisión	572
API db2secProcessServerPrincipalName - Procesar nombre de principal de servicio devuelto desde servidor	574
API db2secRemapUserid - Volver a correlacionar el ID de usuario y la contraseña	575
db2secServerAuthPluginInit - Inicializar el plugin de autenticación del servidor	577
API db2secServerAuthPluginTerm - Liberar los recursos de plugin de autenticación de servidor	580
API db2secValidatePassword - Validar contraseña	581
Las API y definiciones necesarias para los plugins de autenticación de GSS-API	584
Restricciones para los plugins de autenticación de GSS-API	586

Ejemplos de plugins de seguridad	587
Las API de DB2 para hacer copias de seguridad y restauraciones en gestores de almacenamiento	588
db2VendorGetNextObj - Obtener el objeto siguiente en el dispositivo	589
db2VendorQueryApiVersion - Obtener el nivel soportado de la API de almacenamiento de proveedor	592
sqluvdel - Suprimir sesión confirmada	593
sqluvend - Desenlazar un dispositivo de proveedor y liberar sus recursos	594
sqluvget - Leer datos de un dispositivo de proveedor	596
sqluvint - Inicializar y enlazar con un dispositivo de proveedor	598
sqluvpt - Grabar datos en un dispositivo de proveedor	602
DB2_info	604
Vendor_info	608
Init_input	610
Init_output	612
Data	613
Return_code	614
Las API de DB2 para utilizar la compresión con operaciones de copia de seguridad y restauración	615
COMPR_CB	618
COMPR_DB2INFO	619
COMPR_PIINFO	621
Compress - Comprimir un bloque de datos	623
Decompress - Descomprimir un bloque de datos	625
GetMaxCompressedSize - Calcular el tamaño máximo posible del almacenamiento intermedio	627
GetSavedBlock - Obtener el proveedor del bloque de datos para la imagen de copia de seguridad	628
InitCompression - Inicializar la biblioteca de compresión	629
InitDecompression - Inicializar la biblioteca de descompresión	630
TermCompression - Detener la biblioteca de compresión	631
TermDecompression - Detener la biblioteca de descompresión	632

Capítulo 10. Estructuras de datos que las API utilizan	633
db2DistMapStruct	634
db2HistoryData	635
Estructura de datos db2LSN	641
sql_dir_entry	642
SQLB_TBS_STATS	644
SQLB_TBSCONTQRY_DATA	646
SQLB_TBSPQRY_DATA	648
SQLCA	653
sqlchar	654
SQLDA	655
sqldcol	657
sqle_addn_options	660
sqle_client_info	662
sqle_conn_setting	665
sqle_node_local	668

sqle_node_npipe	669
sqle_node_struct	670
sqle_node_tcpip	671
sqledbdesc	672
sqledbdescext	681
sqledbterritoryinfo.	688
sqleninfo	689
sqlfupd	692
sqllob	701
sqlma	702
sqlopt	706
SQLU_LSN	708
sqlu_media_list.	709
SQLU_RLOG_INFO	714
sqlupi	715
SQLXA_XID.	717

Apéndice A. Las API de personalización de precompilador	719
Las API de personalización de precompilador	720

Apéndice B. Registros de anotaciones de DB2	721
Registros de anotaciones cronológicas de DB2	722
Cabecera de registro de anotaciones cronológicas	725
Registros de anotaciones del gestor de transacciones	728

Registros de anotaciones cronológicas del gestor de campos largos	738
Registros de anotaciones cronológicas del gestor de programas de utilidad	740
Registros de anotaciones del gestor de datos	744
Registros de anotaciones del gestor de relaciones	766

Apéndice C. Visión general de la información técnica de DB2 773

Biblioteca técnica de DB2 en copia impresa o en formato PDF	774
Visualización de la ayuda para estados de SQL desde el procesador de línea de mandatos.	776
Acceso a diferentes versiones del Centro de información de DB2	776
Actualización del Centro de información de DB2 instalado en el sistema o en el servidor de intranet	777
Actualización manual del Centro de información de DB2 instalado en el sistema o en el servidor de intranet	778
Guías de aprendizaje de DB2	780
Información de resolución de problemas de DB2	780
Términos y condiciones	781

Apéndice D. Avisos 783

Índice. 787

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 sobre el 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:

- Lenguaje de consulta estructurada (SQL)
- Los lenguajes de programación C, C++, COBOL, FORTRAN, y/o REXX
- El diseño de programas de aplicación

Cómo está 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 general 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.

Cómo esta estructurado este manual

- 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 personalizar la gestión de las bases de datos” presenta las API de plugins de seguridad, copia de seguridad, restauración, archivado de anotaciones cronológicas 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 cronológicas de DB2,” describe la estructura de los distintos registros de anotaciones cronológicas 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 están predefinidos por el sistema.
<i>Cursiva</i>	Indica uno de los siguientes valores: <ul style="list-style-type: none">• 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 siguientes valores: <ul style="list-style-type: none">• 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

Enumera las API de DB2 que aparecen en los ejemplos de DB2.

Importante: Las API administrativas de DB2 no se admiten en aplicaciones multihebra, a menos que gestione explícitamente la serialización de hebras con el uso de las API de contexto de DB2. En lugar de las API administrativas de DB2 junto con las API de contexto de DB2, utilice el procedimiento SYSPROC.ADMIN_CMD almacenado y utilice el gestor de base de datos de DB2 para gestionar automáticamente los contextos del usuario con el soporte de aplicaciones multihebra. Si desea más información sobre el soporte de aplicaciones multihebra y el uso de contextos para aplicaciones con hebras con acceso simultáneo, consulte: “Transacciones simultáneas y acceso a base de datos de varias hebras en aplicaciones de SQL incorporado” en *Developing Embedded SQL Applications*.

La primera tabla incluye las API de DB2 agrupadas por categoría funcional, sus correspondientes 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.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
API de control de bases de datos	“db2DatabaseQuiesce - Inmovilizar la base de datos” en la página 78	db2ApiDf	n/d
API de control de bases de datos	“db2DatabaseUnquiesce - Movilizar base de datos” en la página 83	db2ApiDf	n/d
API de control de bases de datos	“db2DatabaseRestart - Reiniciar base de datos” en la página 80	db2ApiDf	C: dbconn.sqc C++: dbconn.sqC
API de control de bases de datos	“sqlcrea - Crear una base de datos” en la página 404	sqlenv	C: dbcreate.c dbrecov.sqc dbsample.sqc C++: dbcreate.C dbrecov.sq COBOL: db_udcs.cb1 dbconf.cb1 ebcdicdb.cb1
API de control de bases de datos	“sqlcran - Crear una base de datos en un servidor de particiones de base de datos” en la página 402	sqlenv	n/d

Las API de DB2

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

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
API de control de bases de datos	"sqlcdrpd - Descartar base de datos" en la página 420	sqlenv	C: dbcreate.c C++: dbcreate.C COBOL: dbconf.cb1
API de control de bases de datos	"sqlcpan - Descartar una base de datos de un servidor de particiones de base de datos" en la página 418	sqlenv	n/d
API de control de bases de datos	"db2DatabaseUpgrade - Actualizar la versión anterior de la base de datos DB2 al release actual" en la página 85	db2ApiDf	C: dbupgrade.c C++: dbupgrade.C COBOL: dbupgrade.cb1
API de control de bases de datos	"db2XaListIndTrans - Listar transacciones dudosas" en la página 492	db2ApiDf	n/d
API de control de bases de datos	"sqle_activate_db - Activar base de datos" en la página 381	sqlenv	n/d
API de control de bases de datos	"sqle_deactivate_db - Desactivar base de datos" en la página 384	sqlenv	n/d
API de control de bases de datos	"sqlcspqy - Listar transacciones dudosas DRDA" en la página 380	sqlxa	n/d
API de control de bases de datos	"db2SetWriteForDB - Suspender o reanudar las grabaciones de E/S para la base de datos" en la página 323	db2ApiDf	n/d
API de control de bases de datos	"sqlefrce - Desconectar usuarios y aplicaciones del sistema" en la página 426	sqlenv	C: dbconn.sqc dbsample.sqc instart.c C++: dbconn.sqC instart.C COBOL: dbstop.cb1
API de control de instancias	"db2InstanceStart - Iniciar instancia" en la página 175	db2ApiDf	C: instart.c C++: instart.C
API de control de instancias	"db2InstanceStop - Detener instancia" en la página 181	db2ApiDf	C: instart.c C++: instart.C
API de control de instancias	"db2InstanceQuiesce - Inmovilizar instancia" en la página 172	db2ApiDf	n/d
API de control de instancias	"db2InstanceUnquiesce - Movilizar instancia" en la página 186	db2ApiDf	n/d
API de control de instancias	"sqleatin - Conectar a instancia" en la página 393	sqlenv	C: inattach.c utilapi.c C++: inattach.C utilapi.C COBOL: dbinst.cb1
API de control de instancias	"sqleatcp - Conectar a instancia y cambiar contraseña" en la página 390	sqlenv	C: inattach.c C++: inattach.C COBOL: dbinst.cb1
API de control de instancias	"sqledtin - Desconectar de instancia" en la página 424	sqlenv	C: inattach.c utilapi.c C++: inattach.C utilapi.C COBOL: dbinst.cb1
API de control de instancias	"sqlegins - Obtener instancia actual" en la página 439	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dbinst.cb1

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

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
API de control de instancias	“db2UtilityControl - Establecer el nivel de prioridad de los programas de utilidad en ejecución” en la página 346	db2ApiDf	n/d
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 65	db2ApiDf	C: dbinfo.c dbrecov.sqC ininfo.c tscreate.sqC C++: dbinfo.C dbrecov.sqC ininfo.C tscreate.sqC
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 68	db2ApiDf	C: dbinfo.c dbrecov.sqC ininfo.c C++: dbinfo.C dbrecov.sqC ininfo.C
API del gestor de bases de datos y de configuración de bases de datos	“db2AutoConfig - Acceder al Asesor de configuración” en la página 49	db2AuCfg	C: dbcfg.sqC C++: dbcfg.sqC
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 53	db2AuCfg	C: dbcfg.sqC C++: dbcfg.sqC
API de supervisión de bases de datos	“db2GetSnapshotSize - Calcular el tamaño del almacenamiento intermedio de salida necesario para la API db2GetSnapshot” en la página 127	db2ApiDf	n/d
API de supervisión de bases de datos	“db2AddSnapshotRequest - Añadir una petición de instantánea” en la página 42	db2ApiDf	n/d
API de supervisión de bases de datos	“db2MonitorSwitches - Obtener o actualizar los valores de los conmutadores del supervisor” en la página 236	db2ApiDf	C: utilsnap.c C++: utilsnap.C
API de supervisión de bases de datos	“db2GetSnapshot - Obtener una instantánea del estado operacional del gestor de bases de datos” en la página 123	db2ApiDf	C: utilsnap.c C++: utilsnap.C
API de supervisión de bases de datos	“db2ResetMonitor - Restaurar los datos del supervisor del sistema de base de datos” en la página 277	db2ApiDf	n/d
API de supervisión de bases de datos	“db2ConvMonStream - Convertir la corriente de supervisor en el formato anterior a la versión 6” en la página 73	db2ApiDf	n/d
API de supervisión de bases de datos	“db2Inspect - Inspeccionar la base de datos para comprobar la integridad de la arquitectura” en la página 164	db2ApiDf	n/d

Las API de DB2

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

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
API de supervisión de salud de bases de datos	“db2AddContact - Añadir un contacto al que se pueden enviar mensajes de notificación” en la página 38	db2ApiDf	n/d
API de supervisión de salud de bases de datos	“db2AddContactGroup - Añadir un grupo de contactos al que se pueden enviar mensajes de notificación” en la página 40	db2ApiDf	n/d
API de supervisión de salud 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 94	db2ApiDf	n/d
API de supervisión de salud 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 95	db2ApiDf	n/d
API de supervisión de salud de bases de datos	“db2GetAlertCfg - Obtener los valores de configuración de alertas para los indicadores de salud” en la página 103	db2ApiDf	n/d
API de supervisión de salud de bases de datos	“db2GetAlertCfgFree - Liberar la memoria asignada por la API db2GetAlertCfg” en la página 108	db2ApiDf	n/d
API de supervisión de salud 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 109	db2ApiDf	n/d
API de supervisión de salud de bases de datos	“db2GetContactGroups - Obtener la lista de grupos de contactos a los que se puedan enviar mensajes de notificación” en la página 111	db2ApiDf	n/d
API de supervisión de salud de bases de datos	“db2GetContacts - Obtener la lista de contactos a los que se pueden enviar mensajes de notificación” en la página 113	db2ApiDf	n/d
API de supervisión de salud de bases de datos	“db2GetHealthNotificationList - Obtener la lista de contactos a los que se puedan enviar notificaciones de alerta de salud” en la página 117	db2ApiDf	n/d
API de supervisión de salud de bases de datos	“db2ResetAlertCfg - Restablecer la configuración de alertas de los indicadores de salud” en la página 275	db2ApiDf	n/d

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

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
API de supervisión de salud de bases de datos	“db2UpdateAlertCfg - Actualizar los valores de configuración de alertas para los indicadores de salud” en la página 332	db2ApiDf	n/d
API de supervisión de salud de bases de datos	“db2UpdateContact - Actualizar los atributos de un contacto” en la página 340	db2ApiDf	n/d
API de supervisión de salud de bases de datos	“db2UpdateContactGroup - Actualizar los atributos de un grupo de contactos” en la página 342	db2ApiDf	n/d
API de supervisión de salud de bases de datos	“db2UpdateHealthNotificationList - Actualizar la lista de contactos a los que se puedan enviar notificaciones de alerta de salud” en la página 344	db2ApiDf	n/d
API de supervisión de salud de bases de datos	“db2GetSnapshot - Obtener una instantánea del estado operacional del gestor de bases de datos” en la página 123	db2ApiDf	C: utilsnap.c C++: utilsnap.C
API de supervisión de salud de bases de datos	“db2GetSnapshotSize - Calcular el tamaño del almacenamiento intermedio de salida necesario para la API db2GetSnapshot” en la página 127	db2ApiDf	n/d
API de supervisión de salud de bases de datos	“db2GetRecommendations - Obtener recomendaciones para resolver un indicador de salud en estado de alerta” en la página 119	db2ApiDf	n/d
API de supervisión de salud de bases de datos	“db2GetRecommendationsFree - Liberar la memoria asignada por la API db2GetRecommendations” en la página 122	db2ApiDf	n/d
API de traslado de datos	“db2Export - Exportar datos de una base de datos” en la página 96	sqlutil	C: tbmove.sqc C++: tbmove.sqc COBOL: expsamp.sqb impexp.sqb tload.sqb
API de traslado de datos	“db2Import - Importar datos a una tabla, jerarquía, apodo o vista” en la página 149	db2ApiDf	C: dtformat.sqc tbmove.sqc C++: tbmove.sqc COBOL: expsamp.sqb impexp.sqb
API de traslado de datos	“db2Load - Cargar datos en una tabla” en la página 205	db2ApiDf	C: dtformat.sqc tload.sqc tbmove.sqc C++: tbmove.sqc
API de traslado de datos	“db2LoadQuery - Obtener el estado de una operación de carga” en la página 228	db2ApiDf	C: tbmove.sqc C++: tbmove.sqc COBOL: loadqry.sqb
API de recuperación	“db2Backup - Hacer copia de seguridad de una base de datos o un espacio de tablas” en la página 54	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqc
API de recuperación	“db2Restore - Restaurar una base de datos o un espacio de tablas” en la página 280	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqc

Las API de DB2

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

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
API de recuperación	"db2Recover - Restaurar y avanzar una base de datos" en la página 259	db2ApiDf	n/d
API de recuperación	"db2Rollforward - Avanzar una base de datos" en la página 296	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
API de recuperación	"db2HistoryOpenScan - Iniciar una exploración de los registros del histórico de la base de datos" en la página 141	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
API de recuperación	"db2HistoryGetEntry - Obtener la entrada siguiente de los registros del histórico de la base de datos" en la página 139	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
API de recuperación	"db2HistoryCloseScan - Finalizar la exploración de los registros del histórico de la base de datos" en la página 138	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
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 239	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
API de recuperación	"db2HistoryUpdate - Actualizar una entrada de los registros del histórico de la base de datos" en la página 145	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
API de recuperación	"db2ArchiveLog - Archivar el archivo de anotaciones cronológicas activo" en la página 46	db2ApiDf	n/d
API de HADR (High Availability Disaster Recovery)	"db2HADRStart - Iniciar operaciones de HADR (high availability disaster recovery)" en la página 131	db2ApiDf	n/d
API de HADR (High Availability Disaster Recovery)	"db2HADRStop - Detener operaciones de HADR (high availability disaster recovery)" en la página 133	db2ApiDf	n/d
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 135	db2ApiDf	n/d
API de gestión de directorios de bases de datos y de DCS	"sqlcadb - Catalogar una base de datos del directorio de bases de datos del sistema" en la página 396	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dbcat.cb1

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

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
API de gestión de directorios de bases de datos y de DCS	“sqlleuncd - Descatalogar una base de datos del directorio de bases de datos del sistema” en la página 463	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dbcat.cb1
API de gestión de directorios de bases de datos y de DCS	“sqlcgdad - Catalogar una base de datos en el directorio de DCS (Database Connection Services)” en la página 429	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cb1
API de gestión de directorios de bases de datos y de DCS	“sqlcgdel - Descatalogar una base de datos del directorio de DCS (Database Connection Services)” en la página 432	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cb1
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 415	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dbcmt.cb1
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 92	db2ApiDf	C: ininfo.c C++: ininfo.C COBOL: dbcat.cb1 dbcmt.cb1
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 88	db2ApiDf	C: ininfo.c C++: ininfo.C COBOL: dbcat.cb1 dbcmt.cb1
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 87	db2ApiDf	C: ininfo.c C++: ininfo.C COBOL: dbcat.cb1 dbcmt.cb1
API de gestión de directorios de bases de datos y de DCS	“sqlcgdcl - Iniciar una exploración del directorio de DCS (Database Connection Services)” en la página 438	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cb1
API de gestión de directorios de bases de datos y de DCS	“sqlcgdgt - Obtener entradas del directorio de DCS (servicios de conexión de base de datos)” en la página 436	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cb1
API de gestión de directorios de bases de datos y de DCS	“sqlcgdcl - Finalizar una exploración del directorio de DCS (Database Connection Services)” en la página 431	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cb1
API de gestión de directorios de bases de datos y de DCS	“sqlcgdge - Obtener una entrada específica del directorio de DCS (Database Connection Services)” en la página 434	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cb1
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 338	db2ApiDf	n/d

Las API de DB2

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

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
API de gestión de cliente/servidor	"sqleqryc - Consultar valores de conexión del cliente" en la página 450	sqlenv	C: cli_info.c C++: cli_info.C COBOL: client.cb1
API de gestión de cliente/servidor	"sqleqryi - Consultar información sobre el cliente" en la página 452	sqlenv	C: cli_info.c C++: cli_info.C
API de gestión de cliente/servidor	"sqlesetc - Establecer valores de conexión del cliente" en la página 457	sqlenv	C: cli_info.c dbcfg.sqc dbmcon.sqc C++: cli_info.C dbcfg.sqc dbmcon.sqc COBOL: client.cb1
API de gestión de cliente/servidor	"sqleseti - Establecer información sobre el cliente" en la página 460	sqlenv	C: cli_info.c C++: cli_info.C
API de gestión de cliente/servidor	"sqlesact - Establecer serie de contabilidad" en la página 454	sqlenv	COBOL: setact.cb1
API de gestión de cliente/servidor	"db2DatabasePing - Sondear la base de datos para probar el tiempo de respuesta de la red" en la página 76	db2ApiDf	n/d
API de gestión de cliente/servidor	"sqleisig - Instalar manejador de señales" en la página 442	sqlenv	COBOL: dbcmt.cb1
API de gestión de cliente/servidor	"sqleintr - Interrumpir peticiones de aplicaciones" en la página 440	sqlenv	n/d
API de gestión de directorios de LDAP (Lightweight Directory Access Protocol)	"db2LdapRegister - Registrar el servidor DB2 en el servidor LDAP" en la página 194	db2ApiDf	n/d
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 200	db2ApiDf	n/d
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 193	db2ApiDf	n/d
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 191	db2ApiDf	n/d

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

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
API de gestión de directorios de LDAP (Lightweight Directory Access Protocol)	“db2LdapUncatalogNode - Suprimir alias para nombre de nodo del servidor LDAP” en la página 199	db2ApiDf	n/d
API de gestión de directorios de LDAP (Lightweight Directory Access Protocol)	“db2LdapCatalogDatabase - Registrar la base de datos en el servidor LDAP” en la página 188	db2ApiDf	n/d
API de gestión de directorios de LDAP (Lightweight Directory Access Protocol)	“db2LdapUncatalogDatabase - Desregistrar base de datos del servidor LDAP” en la página 198	db2ApiDf	n/d
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 203	db2ApiDf	n/d
API de programación y preparación de aplicaciones	“sqlaintp - Obtener mensaje de error” en la página 351	sql	C: dbcfg.sqcutilapi.c C++: dbcfg.sqlutilapi.C COBOL: checkerr.cb1
API de programación y preparación de aplicaciones	“sqllogstt - Obtener el mensaje de SQLSTATE” en la página 470	sql	C: utilapi.c C++: utilapi.C COBOL: checkerr.cb1
API de programación y preparación de aplicaciones	“sqleisig - Instalar manejador de señales” en la página 442	sqlenv	COBOL: dbcmt.cb1
API de programación y preparación de aplicaciones	“sqleintr - Interrumpir peticiones de aplicaciones” en la página 440	sqlenv	n/d
API de programación y preparación de aplicaciones	“sqlaprep - Precompilar programa de aplicación” en la página 353	sql	C: dbpkg.sqc C++: dbpkg.sqlC
API de programación y preparación de aplicaciones	“sqlabndx - Programa de aplicación de vinculación para crear un paquete” en la página 348	sql	C: dbpkg.sqc dbsample.sqc C++: dbpkg.sqlC

Las API de DB2

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

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
API de programación y preparación de aplicaciones	"sqlarbnf - Volver a vincular paquete" en la página 356	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 467	sqlutil	n/d
Las API específicas de aplicaciones COBOL, FORTRAN y REXX	"sqlgdref - Eliminar la referencia de una dirección" en la página 468	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 469	sqlutil	n/d
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 378 Este mandato o API ha quedado en desuso y puede que se elimine en un futuro release. Consulte tinfo.db2 para obtener un programa de ejemplo que utiliza una función sustituta.	sqlutil	n/d
API de gestión de espacios de tablas y gestión de tablas	"sqlbotcq - Abrir una consulta de contenedor de espacio de tablas" en la página 369	sqlutil	COBOL: tabscont.sqb tspace.sqb
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 361	sqlutil	COBOL: tabscont.sqb tspace.sqb
API de gestión de espacios de tablas y gestión de tablas	"sqlbctcq - Cerrar una consulta de contenedor de espacio de tablas" en la página 359	sqlutil	COBOL: tabscont.sqb tspace.sqb
API de gestión de espacios de tablas y gestión de tablas	"sqlbstsc - Definir contenedores de espacios de tablas" en la página 375	sqlutil	C: dbrecov.sqc C++: dbrecov.sqc COBOL: tabscont.sqb tspace.sqb

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

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
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 367Este mandato o API ha quedado en desuso y puede que se elimine en un futuro release. Consulte tinfo.db2 tsinfo.db2 para obtener un programa de ejemplo que utiliza una función sustituta.	sqlutil	n/d
API de gestión de espacios de tablas y gestión de tablas	“sqlbstpq - Obtener información sobre un espacio de tablas individual” en la página 373	sqlutil	COBOL: tabspace.sqb tspace.sqb
API de gestión de espacios de tablas y gestión de tablas	“sqlbotsq - Abrir una consulta de espacio de tablas” en la página 371Este mandato o API ha quedado en desuso y puede que se elimine en un futuro release.	sqlutil	n/d
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 363	sqlutil	COBOL: tabspace.sqb tspace.sqb
API de gestión de espacios de tablas y gestión de tablas	“sqlbctsq - Cerrar una consulta de espacio de tablas” en la página 360Este mandato o API ha quedado en desuso y puede que se elimine en un futuro release.	sqlutil	n/d
API de gestión de espacios de tablas y gestión de tablas	Este mandato o API ha quedado en desuso y puede que se elimine en un futuro release.	sqlutil	n/d
API de gestión de espacios de tablas y gestión de tablas	“sqluvqdp - Inmovilizar espacios de tablas para una tabla” en la página 481	sqlutil	C: tbmove.sqc C++: tbmove.sqC COBOL: tload.sqb
API de gestión de espacios de tablas y gestión de tablas	“db2Runstats - Actualizar estadísticas para tablas e índices” en la página 307	db2ApiDf	C: tbreorg.sqc C++: tbreorg.sqC COBOL: dbstat.sqb
API de gestión de espacios de tablas y gestión de tablas	“db2Reorg - Reorganizar un índice o una tabla” en la página 265	db2ApiDf	C: tbreorg.sqc C++: tbreorg.sqC COBOL: dbstat.sqb
API de gestión de espacios de tablas y gestión de tablas	“sqlfmem - Liberar la memoria asignada por las API sqlbtcq y sqlbmtsq” en la página 425	sqlenv	C: dbrecov.sqc C++: dbrecov.sqC COBOL: tabscont.sqb tabspace.sqb tspace.sqb
API de gestión de directorios de nodos	“sqllectnd - Catalogar una entrada en el directorio de nodos” en la página 412	sqlenv	C: ininfo.c C++: ininfo.C COBOL: nodecat.cbl

Las API de DB2

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

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
API de gestión de directorios de nodos	"sqlenunc - Descatalogar una entrada del directorio de nodos" en la página 465	sqlenv	C: ininfo.c C++: ininfo.C COBOL: nodecat.cbl
API de gestión de directorios de nodos	"sqlenops - Iniciar una exploración del directorio de nodos" en la página 448	sqlenv	C: ininfo.c C++: ininfo.C COBOL: nodecat.cbl
API de gestión de directorios de nodos	"sqlengne - Obtener la entrada siguiente del directorio de nodos" en la página 446	sqlenv	C: ininfo.c C++: ininfo.C COBOL: nodecat.cbl
API de gestión de directorios de nodos	"sqlencl - Finalizar una exploración del directorio de nodos" en la página 445	sqlenv	C: ininfo.c C++: ininfo.C COBOL: nodecat.cbl
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 338	db2ApiDf	n/d
API de sincronización de satélites	"db2GetSyncSession - Obtener un identificador de sesión de sincronización de satélites" en la página 130	db2ApiDf	n/d
API de sincronización de satélites	"db2QuerySatelliteProgress - Obtener el estado de una sesión de sincronización de satélites" en la página 242	db2ApiDf	n/d
API de sincronización de satélites	"db2SetSyncSession - Establecer sesión de sincronización de satélites" en la página 322	db2ApiDf	n/d
API de sincronización de satélites	"db2SyncSatellite - Iniciar sincronización de satélites" en la página 329	db2ApiDf	n/d
API de sincronización de satélites	"db2SyncSatelliteStop - Pausar sincronización de satélites" en la página 330	db2ApiDf	n/d
API de sincronización de satélites	"db2SyncSatelliteTest - Probar si se puede sincronizar un satélite" en la página 331	db2ApiDf	n/d
API de archivos de anotaciones de lectura	"db2ReadLog - Leer registros de anotaciones cronológicas" en la página 244	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqc
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 251	db2ApiDf	n/d
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 255	db2ApiDf	n/d

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

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
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 258	db2ApiDf	n/d
API de gestión de transacciones dudosas	"db2XaListIndTrans - Listar transacciones dudosas" en la página 492	db2ApiDf	n/d
API de gestión de transacciones dudosas	"sqlxhfrg - Olvidar estado de transacción" en la página 497	sqlxa	n/d
API de gestión de transacciones dudosas	"sqlxphcm - Confirmar una transacción dudosa" en la página 498	sqlxa	n/d
API de gestión de transacciones dudosas	"sqlxphrl - Retrotraer una transacción dudosa" en la página 499	sqlxa	n/d
API de gestión de transacciones dudosas	"sqlcspqy - Listar transacciones dudosas DRDA" en la página 380	sqlxa	n/d
Las API para obtener acceso simultáneo a una base de datos	"sqleAttachToCtx - Conectar a contexto" en la página 502	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 503	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 504	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 505	sql	n/d
Las API para obtener acceso simultáneo a una base de datos	"sqleGetCurrentCtx - Obtener contexto actual" en la página 507	sql	n/d
Las API para obtener acceso simultáneo a una base de datos	"sqleInterruptCtx - Interrumpir contexto" en la página 508	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 509	sql	C: dbthrds.sqc C++: dbthrds.sqC

Las API de DB2

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

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Tipo de API	API de DB2	Archivo de inclusión	Programas de ejemplo
API de gestión de particiones de base de datos	"sqleaddn - Añadir una partición de base de datos al entorno de base de datos particionado" en la página 387	sqlenv	n/d
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 422	sqlenv	n/d
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 402	sqlenv	n/d
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 418	sqlenv	n/d
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 455	sqlenv	C: ininfo.c C++: ininfo.C
API de gestión de particiones de base de datos	"qlugtpi - Obtener información de distribución de tablas" en la página 479	sqlutil	n/d
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 476	sqlutil	n/d
API diversas	"db2AdminMsgWrite - Grabar mensajes de anotación cronológica para la función de administración y duplicación" en la página 44	db2ApiDf	n/d
API diversas	"db2XaGetInfo - Obtener información para un gestor de recursos" en la página 491	sqlxa	n/d
<p>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:</p> <p>C/C++ (UNIX): sqllib/include</p> <p>C/C++ (Windows): sqllib\include</p> <p>COBOL (UNIX): sqllib/include/cobol_a sqllib/include/cobol_i sqllib/include/cobol_mf</p> <p>COBOL (Windows): sqllib\include\cobol_a sqllib\include\cobol_i sqllib\include\cobol_mf</p>			

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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • API db2DatabaseRestart - Reiniciar base de datos • API sqlfrce - Desconectar usuarios y aplicaciones del sistema
dbcreate.c, dbcreate.C	<ul style="list-style-type: none"> • API sqlecrea - Crear base de datos • API sqledrpd - Descartar base de datos
dbinfo.c, dbinfo.C	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • API sqlesetc - Establecer valores de conexión del cliente
dbmigrat.c, dbmigrat.C	<ul style="list-style-type: none"> • API sqlmgdb - Migrar la versión anterior de la base de datos DB2 a la versión actual
dbpkg.sqc, dbpkg.sqC	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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 tablas • 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 sqlcrea - Crear base de datos • API sqledrpd - Descartar base de datos • API sqlfmem - 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	<ul style="list-style-type: none"> • API db2DatabaseRestart - Reiniciar base de datos • API sqlcrea - Crear base de datos • API sqlfrce - Desconectar usuarios y aplicaciones del sistema • API sqlabndx - Vincular programa de aplicación para crear un paquete
dbthrds.sqc, dbthrds.sqC	<ul style="list-style-type: none"> • API sqlAttachToCtx - Conectar a contexto • API sqlBeginCtx - Crear y conectar a un contexto de aplicación • API sqlDetachFromCtx - Desconectar de contexto • API sqlSetTypeCtx - Definir tipo de contexto de aplicación
dtformat.sqc	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • API sqleatcp - Conectar a instancia y cambiar contraseña • API sqleatin - Conectar a instancia • API sqledtin - Desconectar de instancia
ininfo.c, ininfo.C	<ul style="list-style-type: none"> • 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 (Directorio de servicios de conexión de bases de datos) • API sqlegdsc - Iniciar una exploración del directorio de DCS (Directorio de servicios de conexión de bases de datos) • API sqlegdge - Obtener una entrada específica del directorio de DCS (Directorio de servicios de conexión de bases de datos) • API sqlegdgt - Obtener entradas del directorio de DCS (Directorio de servicios de conexión de bases de datos) • API sqlegdcl - Finalizar una exploración del directorio de DCS (Directorio de servicios de conexión de bases de datos) • API sqlegdel - Descatalogar una base de datos del directorio de DCS (Directorio de servicios de conexión de bases de datos) • API sqlesdeg - Establecer el nivel o grado máximo de paralelismo intrapartición para la ejecución de sentencias de SQL
instart.c, instart.C	<ul style="list-style-type: none"> • API sqlefrce - 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • API db2CfgGet - Obtener los parámetros de configuración del gestor de bases de datos o de la base de datos
utilapi.c, utilapi.C	<ul style="list-style-type: none"> • API sqlaintp - Obtener mensaje de error • API sqllogstt - Obtener el mensaje de SQLSTATE • API sqleatin - Conectar a instancia • API sqledtin - Desconectar de instancia
utilsnap.c, utilsnap.C	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • API sqlaintp - Obtener mensaje de error • API sqllogstt - Obtener el mensaje de SQLSTATE
client.cbl	<ul style="list-style-type: none"> • API sqleqryc - Consultar valores de conexión del cliente • API sqlesetc - Establecer valores de conexión del cliente
db_udcs.cbl	<ul style="list-style-type: none"> • API sqleatin - Conectar a instancia • API sqlecrea - Crear base de datos • API sqledrpd - Descartar base de datos
dbcatt.cbl	<ul style="list-style-type: none"> • 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

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

Programa de ejemplo	Las API incluidas
dbcmt.cbl	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • API db2Reorg - Reorganizar un índice o una tabla • API db2Runstats - Actualizar estadísticas sobre las características de una tabla e índices asociados
dcscat.cbl	<ul style="list-style-type: none"> • API sqlegdad - Catalogar una base de datos en el directorio de DCS (Directorio de servicios de conexión de bases de datos) • API sqlegdcl - Finalizar una exploración del directorio de DCS (Directorio de servicios de conexión de bases de datos) • API sqlegdel - Descatalogar una base de datos del directorio de DCS (Directorio de servicios de conexión de bases de datos) • API sqlegdge - Obtener una entrada específica del directorio de DCS (Directorio de servicios de conexión de bases de datos) • API sqlegdgt - Obtener entradas del directorio de DCS (Directorio de servicios de conexión de bases de datos) • API sqlegdsc - Iniciar una exploración del directorio de DCS (Directorio de servicios de conexión de bases de datos)
ebcdicdb.cbl	<ul style="list-style-type: none"> • API sqleatin - Conectar a instancia • API sqlecrea - Crear base de datos • API sqledrpd - Descartar base de datos
expsamp.sqb	<ul style="list-style-type: none"> • API db2Export - Exportar datos de una base de datos • API db2Import - Importar datos a una tabla, jerarquía, apodo o vista
impexp.sqb	<ul style="list-style-type: none"> • API db2Export - Exportar datos de una base de datos • API db2Import - Importar datos a una tabla, jerarquía, apodo o vista
loadqry.sqb	<ul style="list-style-type: none"> • API db2LoadQuery - Obtener el estado de una operación de carga
migrate.cbl	<ul style="list-style-type: none"> • API sqlemgdb - Migrar la versión anterior de la base de datos DB2 a la versión actual

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

Programa de ejemplo	Las API incluidas
nodecat.cbl	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • API sqlarbnd - Volver a vincular paquete
tabscont.sqb	<ul style="list-style-type: none"> • 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 sqlfmem - Liberar la memoria asignada por las API sqlbtcq y sqlbmtsq
tabspace.sqb	<ul style="list-style-type: none"> • API sqlbctsq - Cerrar una consulta 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 tablas • API sqlbotsq - Abrir una consulta de espacio de tablas • API sqlbstpq - Obtener información sobre un espacio de tablas individual • API sqlfmem - Liberar la memoria asignada por las API sqlbtcq y sqlbmtsq
tload.sqb	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • API sqlbctcq - Cerrar una consulta de contenedor de espacio de tablas • API sqlbctsq - Cerrar una consulta de espacio 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 tablas • API sqlbotcq - Abrir una consulta de contenedor de espacio de tablas • API sqlbotsq - Abrir una consulta de espacio 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	<ul style="list-style-type: none"> • API sqlesact - Establecer serie de contabilidad

Capítulo 2. Cambios en las API y estructuras de datos

En esta sección se proporciona información acerca de las API y estructuras de datos que han cambiado.

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, estructura de datos o función de tabla (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	sqlectadb (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)
sqlpstart (V5)	Iniciar gestor de bases de datos	db2InstanceStart (V8)
sqlpstp (V5)	Detener gestor de bases de datos	db2InstanceStop (V8)
sqlpstr (V2)	Iniciar el gestor de bases de datos (DB2 Parallel Edition Versión 1.2)	db2InstanceStart (V8)
sqlstar (V2)	Iniciar gestor de bases de datos (DB2 Versión 2)	db2InstanceStart (V8)
sqlstop (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)

Cambios en las API y estructuras de datos

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, estructura de datos o función de tabla (Versión)
sqlmonsz (V6)	Calcular tamaño necesario para almacenamiento intermedio de salida de sqlmonss()	db2GetSnapshotSize (V7)
sqlmrset (V6)	Restaurar supervisor	db2ResetMonitor (V7)
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)
SQLB-TBSQRY-DATA (V2)	Estructura de datos del espacio de tablas.	SQLB-TBSPQRY-DATA (V5)
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)

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, estructura de datos o función de tabla (Versión)
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)
SQLEXP-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)
sqlbctsq	Cerrar una consulta de espacio de tablas	Funciones de tabla MON_GET_TABLESPACE y MON_GET_CONTAINER (V9.7)
sqlbotsq	Abrir una consulta de espacio de tablas	Funciones de tabla MON_GET_TABLESPACE y MON_GET_CONTAINER (V9.7)
sqlbftpq	Captar los datos de la consulta para filas de un espacio de tablas	Funciones de tabla MON_GET_TABLESPACE y MON_GET_CONTAINER (V9.7)
sqlbgtss	Obtener estadísticas de utilización del espacio de tablas	Funciones de tabla MON_GET_TABLESPACE y MON_GET_CONTAINER (V9.7)
sqlbmtsq	Obtener los datos de la consulta para todos los espacios de tablas	Funciones de tabla MON_GET_TABLESPACE y MON_GET_CONTAINER (V9.7)
sqlbstpq	Obtener información sobre un espacio de tablas individual	Funciones de tabla MON_GET_TABLESPACE y MON_GET_CONTAINER (V9.7)
sqlbtcq	Obtener los datos de la consulta para todos los contenedores de espacios de tablas	Funciones de tabla MON_GET_TABLESPACE y MON_GET_CONTAINER (V9.7)

Cambios en las API y estructuras de datos

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

Nombre	Nombre descriptivo	API de sustitución, estructura de datos o función de tabla
sqlufrol/sqlgfrol	Avance de base de datos (DB2 Versión 1.1)	db2Rollforward
sqluprflw	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
sqluadaw	Obtener autorizaciones del usuario actual	Función de tabla AUTH_LIST_AUTHORITIES_FOR_AUTHID
SQL-AUTHORIZATIONS	Estructura de autorizaciones	No se necesita

Capítulo 3. Cambios en el número de secuencia de anotaciones cronológicas que afectan al comportamiento de las API y las aplicaciones

El tamaño del número de secuencia de anotaciones cronológicas (LSN) ha pasado de seis bytes a ocho bytes. Se ha creado un nuevo tipo de datos, db2LSN, para dar soporte al nuevo tamaño del LSN.

Algunas API con estructuras de entrada y de salida que contienen campos de LSN se han actualizado para utilizar el nuevo tipo de datos db2LSN. Las API actualizadas se tratan en los apartados siguientes. Para asegurarse de que se utilizan las versiones más recientes de las API, es necesario pasar la nueva constante de número de versión de DB2 a las API por medio del parámetro de entrada de API de número de versión.

Como resultado del aumento del tamaño del LSN y de la creación de nuevas versiones de las API, el comportamiento de las API afectadas depende de la configuración de cliente-servidor actual y del nivel de las aplicaciones que se utilizan. Bajo ciertas condiciones, existen restricciones y condiciones que deben tenerse en cuenta acerca de las prestaciones de las API.

Nota: En el caso de las API antiguas listadas en los apartados siguientes, no tienen nuevas versiones y, por lo tanto, sus estructuras no utilizan el tipo de datos db2LSN, si bien su comportamiento sigue dependiendo de la configuración de cliente-servidor que se utilice.

API db2ReadLog

El comportamiento de la API db2ReadLog cambia en función de la configuración de cliente-servidor.

Éstas son las configuraciones posibles:

- Cliente y servidor más recientes: se devuelve el mensaje de error SQL2032N si las aplicaciones existentes utilizan la versión anterior de la API db2ReadLog. Las llamadas a la versión anterior de la API db2ReadLog *no están soportadas*. Dichas aplicaciones deben actualizarse para utilizar la versión más reciente de la API db2ReadLog. Si utiliza una aplicación desarrollada con la versión más reciente de la API db2ReadLog, no hay restricción alguna en las funciones de la API.

Nota: Si el cliente y el servidor están en plataformas endian diferentes, se cambia el orden de los bytes de todos los campos con un tipo de datos nativo de la estructura de salida db2ReadLogInfoStruct, *incluidos* los campos de LSN.

- Cliente antiguo y servidor más reciente: al igual que en la configuración anterior, las llamadas a la API db2ReadLog anterior no están soportadas y se devuelve el mensaje de error SQL2032N. El cliente y todas las aplicaciones existentes deben actualizarse para utilizar la versión más reciente de la API db2ReadLog para poder funcionar.
- Cliente más reciente y servidor antiguo: si se llama a la versión más reciente de la API db2ReadLog desde una aplicación recién desarrollada o si se realiza una llamada a una versión anterior de la API db2ReadLog desde una aplicación existente, no hay restricción alguna en las funciones de la API. Sin embargo, en

Cambios en el número de secuencia de anotaciones cronológicas que afectan al comportamiento de las API y las aplicaciones

ambos casos los registros de anotación devueltos en el almacenamiento intermedio de la anotación cronológica representan los registros antiguos, con el formato de LSN antiguo.

Nota: Si el cliente y el servidor están en plataformas endian diferentes, se cambia el orden de los bytes de todos los campos con un tipo de datos nativo de la estructura de salida `db2ReadLogInfoStruct`, *excluidos* los campos de LSN.

API `db2ReadLogNoConn`

El comportamiento de la API `db2ReadLogNoConn` cambia en función de la configuración de cliente-servidor.

Éstas son las configuraciones posibles:

- Cliente y servidor más recientes: se devuelve el mensaje de error SQL2032N si las aplicaciones existentes utilizan la versión anterior de la API `db2ReadLogNoConn`. Las llamadas a la versión anterior de la API `db2ReadLogNoConn` *no están soportadas*. Dichas aplicaciones deben actualizarse para utilizar la versión más reciente de la API `db2ReadLogNoConn`. Si utiliza una aplicación desarrollada con la versión más reciente de la API `db2ReadLogNoConn`, no hay restricción alguna en las funciones de la API.

API `db2HistoryGetEntry`

El comportamiento de la API `db2HistoryGetEntry` cambia en función de la configuración de cliente-servidor.

Éstas son las configuraciones posibles:

- Cliente y servidor más recientes: se devuelve el mensaje de error SQL2032N si las aplicaciones existentes utilizan la versión anterior de la API `db2HistoryGetEntry` y los LSN de las entradas del archivo histórico son de un tamaño mayor del que puede contener el campo "oLastLSN" de la estructura de salida de la API antigua. Dichas aplicaciones deben actualizarse para utilizar la versión más reciente de la API `db2HistoryGetEntry` con el fin de corregir este comportamiento. Si utiliza una aplicación desarrollada con la versión más reciente de la API `db2HistoryGetEntry`, no hay restricción alguna en las funciones de la API.
- Cliente antiguo y servidor más recientes: se devuelve el mensaje de error SQL2032N si se utiliza una versión anterior de la API `db2HistoryGetEntry` y los LSN de las entradas del archivo histórico son de un tamaño mayor del que puede contener el campo `oLastLSN` de la estructura de salida de la API antigua. Es necesario actualizar el cliente y las aplicaciones existentes para corregir el problema.
- Cliente más reciente y servidor antiguo: si se llama a la versión más reciente de la API `db2HistoryGetEntry` desde una aplicación recién desarrollada o si se realiza una llamada a una versión anterior de la API `db2HistoryGetEntry` desde una aplicación existente, no hay restricción alguna en las funciones de la API.

`db2Prune`, API

El comportamiento de la API `db2Prune` cambia en función de la configuración de cliente-servidor.

Éstas son las configuraciones posibles:

Cambios en el número de secuencia de anotaciones cronológicas que afectan al comportamiento de las API y las aplicaciones

- Cliente y servidor más recientes: si utiliza una nueva aplicación con la versión más reciente de la API db2Prune o si utiliza una aplicación existente que usa una versión anterior de la API db2Prune, no existe ninguna restricción relacionada con las funciones de la API.
- Cliente antiguo y servidor más reciente: si utiliza una aplicación existente y realiza llamadas a una versión anterior de la API db2Prune, no hay restricción alguna en las funciones de la API.
- Cliente más reciente y servidor antiguo: si se utiliza la versión más reciente de la API db2Prune con una serie de LSN como entrada, se devuelve el mensaje de error SQL2032N si la serie de LSN representa un LSN que sobrepase el valor 0xFFFF FFFF FFFF o si tiene menos de 12 caracteres de longitud. Para evitar este error, el servidor debe actualizarse de modo que tenga como mínimo el nivel del cliente. Si se realizan llamadas a una versión anterior de la API db2Prune a través de una aplicación existente, no hay restricción alguna en las funciones de la API.

API sqlbftpq, sqlbstpq y sqlbmtsq

El comportamiento de las API sqlbftpq, sqlbstpq y sqlbmtsq cambia en función de la configuración de cliente-servidor.

Nota: Estas API han quedado en desuso y puede que se eliminen en un futuro release. En su lugar, pueden utilizarse las funciones de tabla MON_GET_TABLESPACE y MON_GET_CONTAINER, que devuelven más información.

Éstas son las configuraciones posibles:

- Cliente y servidor más recientes: para cada una de estas API, si se realizan llamadas a una versión anterior de la API a través de una aplicación existente, se devuelve el mensaje de error SQL2032N, siempre y cuando el LSN devuelto tenga un tamaño mayor que el que el campo **lifeLSN** de la estructura SQLB_TBSPQRY_DATA anterior puede albergar. Hay una consideración adicional que debe tenerse en cuenta para la API sqlbmtsq: al realizar una llamada a una versión anterior de esta API, el campo **lifeLSN** de la estructura SQLB_TBSPQRY_DATA contendrá un valor no válido, y todos los demás campos de la estructura contendrán valores válidos. La actualización de las aplicaciones para que utilicen las versiones más recientes de las API puede resolver estos problemas. Si se utilizan las versiones más recientes de estas API, no hay restricción alguna en sus funciones.
- Cliente anterior y servidor más reciente: en esta configuración, el uso de la versión anterior de cualquiera de las tres API da como resultado un comportamiento que refleja el de las API en la configuración donde tanto el cliente como el servidor pertenecen a la versión más reciente, como se indica anteriormente, pero sin la limitación indicada para la API sqlbmtsq. Para resolver estas limitaciones, el cliente debe actualizarse a como mínimo el nivel de release del servidor, y las aplicaciones que se utilizan deben actualizarse para que utilicen las versiones más recientes de las API.
- Cliente más reciente y servidor antiguo: al realizar una llamada a una versión anterior de la API sqlbmtsq a través de una aplicación existente, el campo **lifeLSN** de la estructura SQLB_TBSPQRY_DATA se devuelve con un valor de LSN no válido. Todos los demás campos de la estructura siguen siendo válidos. Para corregir este comportamiento, la aplicación en uso debe actualizarse para incorporar la versión más reciente de la API sqlbmtsq. En el caso de las demás API, no hay restricción alguna en las funciones de las API independientemente de la versión de las API que se esté utilizando.

Cambios en el número de secuencia de anotaciones cronológicas que afectan al comportamiento de las API y las aplicaciones

API sqlurlog (API de nivel anterior)

El comportamiento de la API sqlurlog cambia en función de la configuración de cliente-servidor.

Éstas son las configuraciones posibles:

- Cliente y servidor más recientes: si se utiliza la API sqlurlog en esta configuración de cliente-servidor, puede devolverse el mensaje de error SQL2650N si alguno de los LSN devueltos tiene un valor mayor que el que pueden contener los campos de LSN de la estructura de salida SQLU_RLOG_INFO. La única manera de evitar este problema sería modificar las aplicaciones que utilizan la API sqlurlog para que utilicen su sustituta, que es la API db2ReadLog.

Nota: Si se utiliza la API sqlurlog en esta configuración, los registros de anotaciones cronológicas devueltos mediante el almacenamiento intermedio de anotaciones cronológicas representan los nuevos registros de anotaciones cronológicas y contienen los LSN del nuevo formato db2LSN.

- Cliente antiguo y servidor más reciente: como ocurre con las API db2ReadLog y db2ReadLogNoConn, las llamadas a la API sqlurlog realizadas desde clientes antiguos hacia servidores con un nivel actualizado “no están soportadas”. Cualquier intento de utilizar la API sqlurlog dará como resultado el mensaje de error SQL1198N. Para evitar este problema, es necesario actualizar el cliente al nivel del servidor.
- Cliente más reciente y servidor antiguo: no hay restricción alguna en las funciones de la API sqlurlog cuando se utiliza en esta configuración de cliente-servidor.

API sqlbftsq, sqlbstsq y sqlbtsq (API de nivel anterior)

El comportamiento de las API sqlbftsq, sqlbstsq y sqlbtsq cambia en función de la configuración de cliente-servidor.

Éstas son las configuraciones posibles:

- Cliente y servidor más recientes: el uso de cualquiera de estas tres API en este tipo de configuración de cliente-servidor puede dar como resultado que el campo **lifelSN** de la estructura SQLB_TBSPQRY_DATA de nivel anterior contenga un valor de LSN no válido. Esto es una consecuencia de la incapacidad del campo **lifelSN** de la estructura SQLB_TBSPQRY_DATA de contener valores grandes de LSN. Para evitar este problema, deben utilizarse las API que son las sucesoras de las API sqlbftsq, sqlbstsq y sqlbtsq, donde las sucesoras son las API sqlbftpq, sqlbstpq y sqlbmtsq.
- Cliente antiguo y servidor más reciente: el comportamiento de las API descrito en la configuración anterior se aplica también a esta configuración.
- Cliente más reciente y servidor antiguo: no hay restricción alguna en las funciones de las API sqlbftsq, sqlbstsq y sqlbtsq cuando se utilizan en esta configuración de cliente-servidor.

Capítulo 4. Cómo están organizadas las descripciones de las API

En esta sección se proporciona información acerca de cómo se organizan las descripciones de 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 base de datos particionada, 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

Uno de los requisitos 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`

Cómo están organizadas las descripciones de las API

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.

Cómo están organizadas las descripciones de las API

Parámetros de la API de REXX

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

Archivos de inclusión para aplicaciones de las API de DB2

En la lista siguiente 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 autenticación 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.

Archivos de inclusión para aplicaciones de las API de DB2

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 5. 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 **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
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 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.

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

iType

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

- DB2CONTACT_EMAIL
- DB2CONTACT_PAGE

piAddress

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

iMaxPageLength

Entrada. La longitud máxima del mensaje para 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 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** se establece en 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

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 nivel de release de la estructura transferida como segundo parámetro **pParmStruct**. Para utilizar la estructura

db2AddSnapshotRequest - Añadir una petición de instantánea

db2AddSnapshotData tal como se ha descrito anteriormente, especifique db2Versio910. Si desea utilizar una versión diferente de esta estructura, vea el archivo de cabecera db2ApiDf del directorio include para obtener una lista completa de versiones soportadas. Debe utilizar la versión de la estructura db2AddSnapshotRequestData correspondiente al número de versión que especifique.

pParmStruct

Entrada o salida, o ambas. 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 este 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.

Tras la llamada a la API, **iRequestType**, **iQualifierType**, **iRequestFlags** y **piQualifierData** se restablecen 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 - Grabar mensajes de anotación cronológica para la función de administración y duplicación

Proporciona un mecanismo para que los usuarios y la función de Duplicación graben información en el archivo de anotaciones cronológicas **db2diag**, y en el archivo de anotaciones cronológicas de notificación 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 el nivel de release de la estructura que se ha pasado 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 cronológicas 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 archivo de anotaciones cronológicas de notificación de administración solamente si el tipo de error especificado es menor o igual que el valor del parámetro de configuración del gestor de bases de datos **notifylevel**. Registrará anotaciones en el archivo de anotaciones cronológicas **db2diag** solamente si el tipo de error especificado es menor o igual que el valor del parámetro de configuración del gestor de bases de datos **diaglevel**. No obstante, toda la información que se graba en las anotaciones cronológicas de notificación de administración se duplica en el archivo de anotaciones cronológicas **db2diag**, a menos que el parámetro de configuración del gestor de bases de datos **diaglevel** esté establecido 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 autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- 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;
```

db2ArchiveLog - Archivar el archivo de anotaciones cronológicas activo

```
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 base de datos particionada. 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.

Ámbito

En un entorno de base de datos particionada, 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 coordinadora. 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;
```

db2AutoConfig - Acceder al Asesor de configuración

```
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 que se ha pasado 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 include) 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 include) 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 (sólo válido con DB2_SG_APPLY y DB2_SG_APPLY_DB) exclusivamente en la partición de base de datos actual. Por omisión, las recomendaciones para las bases 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 el valor de **iApply** se ha establecido 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 el valor de **iApply** se establece 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 el valor de **iApply** se establecer 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 las páginas. 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 pasa a la API. Si la versión es DB2 v9.5 o posterior, no se devolverá **maxagents**, pero sí se devolverá para las versiones anteriores a ésta. 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 que se ha pasado 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 base de datos particionada, 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 específica 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 autorizaciones 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;
```

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

```
    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;
```

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

```
    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.

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

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 emisor, por ejemplo, si se sabe que se han montado todos los soportes de almacenamiento necesarios para la copia de seguridad y no es necesario que aparezcan mensajes de solicitud de los 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

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

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

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

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_DEDUP_DEVICE

Optimiza el formato de la imagen de copia de seguridad de los dispositivos de almacenamiento de destino que dan soporte a la optimización de almacenamiento de datos.

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. Esta opción no es válida para una copia de seguridad fuera de línea ni para una copia de seguridad de múltiples particiones.

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 selectivas en las que INCLUDE es el valor por omisión.

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

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

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 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.

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

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 selectiva.

No puede utilizar SQLU_SNAPSHOT_MEDIA con ninguna de las opciones 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

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

El comportamiento por omisión para una copia de seguridad selectiva 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 duplicación (**INCLUDE LOGS** es el valor por omisión de todas las copias de seguridad selectivas 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.

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

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 selectiva 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 normas de control del acceso basadas en etiquetas (normas 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 base de datos particionada

- 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.

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

- **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;
```

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

```
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 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 include) 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.

dbname

Entrada. Nombre de la base de datos.

Parámetros de la estructura de datos db2CfgParam

símbolo

Entrada. Identificador del parámetro de configuración.

Las entradas y tipos de datos válidos para el elemento de **token** db2CfgParam están listados en "Resumen de parámetros de configuración".

Nota: Se añaden símbolos adicionales de db2Cfg para admitir la obtención (o configuración) de los parámetros de configuración de recurso de almacenamiento en antememoria de clúster y los parámetros de configuración de estructura de recurso de almacenamiento en antememoria de clúster en un entorno DB2 pureScale.

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 include) 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 estas dos palabras clave, 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 peticiones 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 partición de base de datos 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 base de datos, una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT

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;
    SQL_PDB_NODE_TYPE member;
} db2Cfg;

typedef SQL_STRUCTURE db2CfgParam
{
    db2UInt32 token;
    char *ptrvalue;
```


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

```
    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 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 include) 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.

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

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.

db2CfgReset

Restablecer todos los parámetros de configuración del gestor de bases de datos con 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**. Esto solo es aplicable en un entorno de base de datos particionada e ignorada en otros entornos.

db2CfgSingleMember

Para actualizar o restaurar la configuración de la base de datos en un miembro específico, establezca este distintivo y proporcione un valor para miembro. Esto solamente es aplicable en un entorno DB2 pureScale y se ignora en otros entornos.

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. El parámetro **DBPARTITIONNUM** solamente es aplicable en un entorno de base de datos particionada y debe ser 0 en un entorno DB2 pureScale.

member

Entrada. Especifica en qué miembro esta API establecerá el valor de configuración. El parámetro **MEMBER** solamente es aplicable en un entorno DB2 pureScale y se ignora en otros entornos.

Parámetros de la estructura de datos db2CfgParam

símbolo

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

Nota: Se añaden símbolos adicionales de db2Cfg para admitir la obtención (o configuración) de los parámetros de configuración de recurso de almacenamiento en antememoria de clúster y los parámetros de configuración de estructura de recurso de almacenamiento en antememoria de clúster en un entorno DB2 pureScale.

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 include) 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

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

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.

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 */
```

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

```
/* 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 el nivel de release de la estructura que se ha pasado 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). En la sección siguiente 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). En la sección siguiente 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.

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

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 pasa por alto.

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 es responsable de asignar memoria para esta matriz antes de la llamada a 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 si se utiliza desde un cliente DB2 UDB Versión 7 a través de un DB2 Connect Versión 8 con un servidor de bases de datos de sistema principal DB2 conectado.

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 movlizar 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, SYSMANT o SYSCTRL tendrán acceso a la base de datos y a sus objetos.

Si una base de datos está en el estado SUSPEND_WRITE, no se puede poner en la modalidad inmovilizada.

Autorización

Una de las autorizaciones 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 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 base de datos particionada, 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 afecta únicamente a la partición de base de datos en la que se ejecuta la API. En entornos DB2 pureScale, esta API puede ejecutarse desde cualquier miembro y puede desencadenar, cuando sea necesario, una recuperación de anomalía general de grupo, que realiza la recuperación de anomalía general para todos los miembros del grupo, o una recuperación de anomalía general de miembro.

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 el nivel de release de la estructura que se ha pasado 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 de anomalía general.

DB2_DB_RESUME_WRITE

Valor necesario para realizar una recuperación de anomalía general en una base de datos que tiene grabaciones de E/S suspendidas. En entornos DB2 pureScale, esta opción reanuda las operaciones de grabación de E/S para todos los miembros suspendidos.

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

db2DatabaseRestart - Reiniciar base de datos

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 de anotaciones cronológicas circulares, una operación de reinicio de base de datos fallará si existe algún problema con los espacios de tablas, 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 podrá 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.

nombreusuario

Nombre de usuario utilizado para reiniciar la base de datos.

contraseña

Contraseña utilizada para autenticar 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 autorizaciones 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 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.

db2DatabaseUnquiesce - Movilizar base de datos

Parámetros específicos de la estructura de datos db2gDbUnquiesceStruct

iDatabaseNameLen

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

db2DatabaseUpgrade - Actualizar la versión anterior de la base de datos DB2 al release actual

Convierte una base de datos DB2 versión 9.5, versión 9.7 o versión 9.8 al release actual.

Autorización

SYSADM

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
db2DatabaseUpgrade (
    db2UInt32 versionNumber,
    void      *pParmStruct,
    struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2DatabaseUpgradeStruct
{
    char *piDbAlias;
    char *piUserName;
    char *piPassword;
    db2UInt16 iDbAliasLen;
    db2UInt16 iUserNameLen;
    db2UInt16 iPasswordLen;
    db2UInt32 upgradeFlags;
} db2DatabaseUpgradeStruct;
```

Parámetros de la API db2DatabaseUpgrade

versionNumber

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

pParmStruct

Entrada. Puntero a la estructura db2DatabaseUpgradeStruct.

pSqlca Salida. Puntero a la estructura sqlca.

Parámetros de datos db2DatabaseUpgradeStruct

piDbAlias

Entrada. Serie que contiene el alias de la base de datos que está catalogada en el directorio de bases de datos del sistema.

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.

db2DatabaseUpgrade - Actualizar la versión anterior de la base de datos DB2 al release actual

iDbAliasLen

Entrada. Número entero sin signo, de 2 bytes, que representa la longitud, expresada en bytes, del alias de base de datos.

iPasswordLen

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.

iUserNameLen

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.

upgradeFlags

Entrada. Opciones de actualización de base de datos. Los valores válidos (definidos en el archivo de cabecera db2ApiDf que se encuentra en el directorio include) son:

DB2DBUPGRADE_REBINDALL

Volver a vincular los paquetes no válidos durante la actualización de la base de datos.

0 Valor por omisión, que no volverá a vincular todos los paquetes durante la actualización de la base de datos.

Notas de uso

Esta API sólo actualizará una base de datos a una versión más nueva y no se puede utilizar para convertir una base de datos actualizada en su versión anterior.

La base de datos debe catalogarse antes de la actualización.

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 el nivel de release de la estructura que se ha pasado 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).

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

SQL_DCE

La base de datos reside en directorios DCE.

authentication

Tipo de autenticación. Los valores válidos son:

SQL_AUTHENTICATION_SERVER

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

SQL_AUTHENTICATION_CLIENT

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

SQL_AUTHENTICATION_DCE

La autenticación se realiza mediante los Servicios de seguridad de DCE.

SQL_AUTHENTICATION_KERBEROS

La autenticación se realiza mediante el Mecanismo de seguridad de Kerberos.

SQL_AUTHENTICATION_NOT_SPECIFIED

DB2 ya no requiere autenticación 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 autenticación se realiza en el nodo donde reside la base de datos de destino, y que la contraseña de autenticación se debe cifrar.

SQL_AUTHENTICATION_DATAENC

Especifica que la autenticación 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 autenticación 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 autenticación 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.

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

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 rellenos 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 el nivel de release de la estructura que se ha pasado 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. No obstante, 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 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 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 autorizaciones siguientes:

- Autorización DATAACCESS
- 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 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 donde se almacenarán 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/test01/lob/path, el nombre de archivo LOB actual es bar y si se ha

db2Export - Exportar datos de una base de datos

establecido el modificador de tipo de archivo **LOBSINSEPF**ILES, los archivos LOB creados serán /u/test01/LOB/path/bar.001.lob, /u/test01/LOB/path/bar.002.lob, etc. Si el modificador de tipo de archivo **LOBSINSEPF**ILES no se ha establecido, todos los documentos LOB se concatenarán y se incluirán en un archivo /u/test01/lob/path/bar.001.lob.

piDataDescriptor

Entrada. Puntero a una estructura `sqlldcol` que especifica los nombres de columnas del archivo de salida. El valor del campo `dc01meth` 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`, ubicado en el directorio `include`) 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.

Se buscará la coincidencia de las columnas para el archivo externo (de **piDataDescriptor**) y las columnas de base de datos de la sentencia `SELECT` de acuerdo con su posición en la lista o en la estructura. 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_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 `sqlldcol` 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 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 `include`) son:

SQLU_INITIAL

Llamada inicial. Este valor debe utilizarse en la primera llamada a la API. Si se devuelve la llamada inicial o cualquier llamada posterior y requiere que la aplicación que realiza la llamada lleve a cabo alguna acción antes de completar la operación de exportación solicitada, la acción del emisor 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 que había solicitado el programa de utilidad ha finalizado y que el programa de utilidad puede seguir 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 donde se almacenarán 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`

db2Export - Exportar datos de una base de datos

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/test01/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/test01/xml/path/bar.001.xml`, `/u/test01/xml/path/bar.002.xml`, etc. Si el modificador de tipo de archivo `XMLINSEPFILES` no se ha establecido, todos los documentos XML se concatenarán y se incluirán en un archivo `/u/test01/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.

db2Export - Exportar datos de una base de datos

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.

Se emite un mensaje de aviso 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**. 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 dinámico 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.

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 siguiente, 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.

Importante: Este mandato o API ha quedado en desuso y puede que se elimine en un futuro release, ya que el supervisor de salud ha quedado en desuso en la Versión 9.7. No recibe soporte en los entornos DB2 pureScale. Para obtener más información, consulte el tema “El Supervisor de salud ha quedado en desuso” en <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

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;
```

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

```
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 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. El nombre del espacio de tablas o contenedor de espacio de tablas cuando el parámetro de tipo de objeto, **iObjType**, se establece en 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. 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 y DB2ALERTCFG_OBJTYPE_DATABASE.

ioNumIndicators

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

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

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 ni 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 el estado, 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 sólo para DB2ALERTCFG_SCRIPTTYPE_OS.

stmtTermChar

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

pUserID

Cuenta de usuario utilizada para ejecutar el script.

pPassword

Contraseña de la cuenta de usuario **pUserId**.

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

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 establece NULL en **pioIndicators**, 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 desea la configuración. Cuando se utilice de esta manera, establezca **ioNumIndicators** en la longitud de matriz de entrada y el resto de los campos en db2GetAlertCfgInd en 0 o NULL.

Toda la memoria bajo este puntero se asigna mediante el motor y debe liberarse con una llamada a la API db2GetAlertCfgFree siempre que la API db2GetAlertCfg finalice sin ningún error. Consulte 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.

Importante: Este mandato o API ha quedado en desuso y puede que se elimine en un futuro release, ya que el supervisor de salud ha quedado en desuso en la Versión 9.7. No recibe soporte en los entornos DB2 pureScale. Para obtener más información, consulte el tema “El Supervisor de salud ha quedado en desuso” en <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

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 el nivel de release de la estructura que se ha pasado 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 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.

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

numContacts

Número de **pContacts**.

pContacts

Puntero a la estructura **db2ContactTypeData**. Los campos **pGroupName**, **pDescription**, **pContacts** y **pContacts**. El usuario debe asignar previamente **pName**, con sus correspondientes tamaños máximos. Invoque **db2GetContactGroup** con **numContacts=0** y **pContacts=NULL** para que se devuelva la longitud necesaria para **pContacts** en **numContacts**.

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** se establece en 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 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, contiene el número de estructuras db2ContactGroupDesc necesarias en **poGroups**.

poGroups

Salida. Puntero a la estructura db2ContactGroupDesc.

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

Parámetros de la estructura de datos db2ContactGroupDesc

poName

Salida. Nombre del grupo. Este emisor debe asignar previamente este parámetro, con el correspondiente tamaño máximo.

poDescription

Salida. Descripción del grupo. Este emisor debe asignar previamente este parámetro, con el correspondiente tamaño máximo.

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 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 **poContacts**.

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

poContacts

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

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.

db2GetDistMap - Obtener correlación de distribución

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 db2GetRowPartNum 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 DATAACCESS
- 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

```
db2GetDistMap (
    sqluint32 db2VersionNumber,           /* Número versión base datos */
    void * pParmStruct,                   /* Parámetros de entrada/salida */
    struct sqlca * pSqlca);               /* SQLCA */

where
SQL_STRUCTURE db2DistMapStruct
{
    unsigned char          *tname;        /* Nombre de tabla */
                                   /* completo */
    struct db2PartitioningInfo *partinfo; /* Información de */
                                   /* particionamiento */
};

SQL_STRUCTURE db2PartitioningInfo
{
    sqluint32          pmaplen;          /* Longitud de correlación */
                                   /* de particionamiento */
    SQL_PDB_NODE_TYPE *pmap;            /* Correlación particionam. */
    unsigned short     sqld;             /* Número de elementos */
                                   /* db2PartitioningKey */
                                   /* utilizados */
    struct db2PartitioningKey sqlpartkey[SQL_MAX_NUM_PART_KEYS]; /* */
                                   /* KEYS */
};

SQL_STRUCTURE db2PartitioningKey
```

db2GetDistMap - Obtener correlación de distribución

```
{
    unsigned short          sqltype;    /* Tipo clave datos    */
    unsigned short          sqllen;    /* Longitud clave datos */
};
```

Parámetros de la API db2GetDistMap

nombret

Nombre totalmente calificado de la tabla.

pmaplen

Longitud de la correlación de distribución.

pmap

Nombre de la correlación de distribución.

sqlid

Número de elementos utilizados en la estructura **sqlpartkey**.

sqlpartkey

Claves de distribución que se utilizan para la tabla.

Notas de uso

La estructura **pmap** de db2PartitioningInfo debe apuntar a una matriz de SQL_PDB_MAP_SIZE_32K entradas.

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

Devuelve la lista de contactos y de grupos de contactos, o ambas, que han de recibir notificaciones relacionadas con la salud de una instancia.

Una lista de contactos consta de las direcciones de correo electrónico y las direcciones de Internet de buscapersonas de las personas que han de recibir notificaciones cuando existan condiciones de salud que no son 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 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,

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

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** se establece en 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.

Importante: Este mandato o API ha quedado en desuso y puede que se elimine en un futuro release, ya que el supervisor de salud ha quedado en desuso en la Versión 9.7. No recibe soporte en los entornos DB2 pureScale. Para obtener más información, consulte el tema “El Supervisor de salud ha quedado en desuso” en <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

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 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 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 las instancias autónomas, se debe utilizarse `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` en el directorio `include`.

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 el parámetro de tipo de objeto, **iObjType**, se establece en `DB2HEALTH_OBJTYPE_TABLESPACE` o `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-espacio-tablas.nombre-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`. De lo 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.

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

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 se asigna mediante 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 `sql1lib/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.

Importante: Este mandato o API ha quedado en desuso y puede que se elimine en un futuro release, ya que el supervisor de salud ha quedado en desuso en la Versión 9.7. No recibe soporte en los entornos DB2 pureScale. Para obtener más información, consulte el tema “El Supervisor de salud ha quedado en desuso” en <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

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 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 autorizaciones 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);
```

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

```
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 nivel de release de la estructura transferida como segundo parámetro **pParmStruct**. Para utilizar la estructura tal como se ha descrito anteriormente, especifique db2Version810 o una versión más reciente. 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 (área de supervisor) 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 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
- SQLM_DBMON_VERSION9_7 (necesaria para la información relacionada con la reclamación de extensiones de MDC por medio de una reorganización)
- SQLM_DBMON_VERSION10 (necesaria para la información relacionada con la reclamación de extensiones de ITC por medio de una reorganización)

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** se ha establecido en SQLM_DBMON_VERSION7 o una versión más reciente.
- valor de nodo

Nota: Para las instancias autónomas debe utilizarse el valor SQLM_CURRENT_NODE.

poOutputFormat

Formato de la corriente de datos devuelta por el servidor. Será uno de los valores siguientes:

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

- 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, ubicado en el directorio include) son:

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

Nota: SQLM_CLASS_HEALTH y SQLM_CLASS_HEALTH_WITH_DETAIL han quedado en desuso y se pueden eliminar en un futuro release ya que el supervisor de salud ha quedado en desuso en la versión 9.7.

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 salud que incluya información de colección, utilice el campo **AGENT_ID** en 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 autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- 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;
```

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

```
db2UInt32 iVersion;  
db2int32 iNodeNumber;  
db2UInt32 iSnapshotClass;  
} db2gGetSnapshotSizeData;
```

Parámetros de la API db2GetSnapshotSize

versionNumber

Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo parámetro **pParmStruct**. Para utilizar la estructura tal como se ha descrito anteriormente, especifique db2Version810 o una versión más reciente. 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 (área de supervisor) 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 de 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

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

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** se ha establecido en `SQLM_DBMON_VERSION7` o una versión más nueva.
- 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`, ubicado en el directorio `include`) son:

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

Notas de uso

Esta función genera una cantidad significativa de uso de proceso adicional. La asignación y liberación dinámica de memoria para cada llamada a la API `db2GetSnapshot` exige también recursos. 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 el nivel de release de la estructura que se ha pasado 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 autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT

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 nivel de release de la estructura transferida como segundo parámetro **pParmStruct**.

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

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 en espera.

iByForce

Entrada. Este argumento se pasa por alto 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 en 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 en 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 autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT

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 nivel de release de la estructura transferida como segundo parámetro **pParmStruct**.

pParmStruct

Entrada. Puntero a la estructura db2HADRStopStruct.

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

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

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT

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;
```

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

Parámetros de la API db2HADRTakeover

versionNumber

Entrada. Especifica la versión y 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 en espera 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 período de ventana similar (establecido en el parámetro de configuración de base de datos **hadr_peer_window** en un valor distinto de cero). Si no se desactiva la base de datos primaria antes de que caduque la ventana similar, dará lugar a la *división de conocimientos* o *primaria dual*. Si se ejecutan cuando el par HADR no está en estado similar o similar desconectado (la ventana similar ha caducado), se devuelve un error.

No puede utilizar esta opción cuando ha modalidad de sincronización se ha establecido en ASYNC o SUPERASYNC.

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 con una diferencia de 5 segundos entre sí. Es decir, la operación puede ser correcta cuando debería fallar, o fallar cuando debería ser correcta. Debe utilizar un servicio

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

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 de los registros del histórico de la base de datos

Finaliza una exploración de los registros de histórico de base de datos y libera recursos de DB2 necesarios para la exploración. Antes de ejecutar esta API es necesario invocar satisfactoriamente la API db2HistoryOpenScan.

Nota: Esta API solamente recibe soporte en los lenguajes de programación C, C++ o Java. En los lenguajes de programación COBOL, FORTRAN y REXX ya no recibe soporte. Puede emitir una consulta para acceder a registros del histórico de la base de datos utilizando la vista administrativa DB_HISTORY.

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 registros del histórico de la base de datos, consulte la API db2HistoryOpenScan.

db2HistoryGetEntry - Obtener la entrada siguiente de los registros del histórico de la base de datos

Obtiene la entrada siguiente de los registros del histórico de la base de datos. Antes de ejecutar esta API es necesario invocar satisfactoriamente la API db2HistoryOpenScan.

Nota: Esta API solamente recibe soporte en los lenguajes de programación C, C++ o Java. En los lenguajes de programación COBOL, FORTRAN y REXX ya no recibe soporte. Puede emitir una consulta para acceder a registros del histórico de la base de datos utilizando la vista administrativa DB_HISTORY.

Autorización

Ninguna

Conexión necesaria

Instancia. No es necesario invocar sqlcatin 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 el nivel de release de la estructura que se ha pasado 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.

db2HistoryGetEntry - Obtener la entrada siguiente de los registros del histórico de la base de datos

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 registros del histórico de la base de datos, consulte la API db2HistoryOpenScan.

db2HistoryOpenScan - Iniciar una exploración de los registros del histórico de la base de datos

Inicia una exploración de registros de histórico de base de datos.

Nota: Esta API solamente recibe soporte en los lenguajes de programación C, C++ o Java. En los lenguajes de programación COBOL, FORTRAN y REXX ya no recibe soporte. Puede emitir una consulta para acceder a registros del histórico de la base de datos utilizando la vista administrativa DB_HISTORY.

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;
    db2UInt32 oMaxLogStreams;
    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;
    db2UInt32 oMaxLogStreams;
    db2UInt16 iCallerAction;
    db2UInt16 oHandle;
} db2gHistoryOpenStruct;
```

Parámetros de la API db2HistoryOpenScan

versionNumber

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

pParmStruct

Entrada o salida. Un 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 de registros del histórico de la base de datos.

oMaxTbspaces

Salida. Número máximo de nombres de espacios de tablas que se almacenan con cualquier entrada del archivo histórico.

oMaxLogStreams

Salida. El número máximo de corrientes de anotaciones cronológicas almacenadas en 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 include) son:

DB2HISTORY_LIST_HISTORY

Lista todos los sucesos que están actualmente anotados cronológicamente en los registros del histórico de la base de datos.

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 red denominación de espacios de tablas.

DB2HISTORY_LIST_ALT_TABLESPACE

Lista las operaciones de modificar espacios 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 REORGANIZE TABLE. 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 los registros del histórico de la base de datos.
- 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.

db2HistoryOpenScan - Iniciar una exploración de los registros del histórico de la base de datos

Está permitido un máximo de ocho exploraciones de registros del histórico de la base de datos por cada proceso.

Para listar todas las entradas de los registros del histórico de la base de datos, 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. En un bucle, realice las acciones siguientes:
 - Llamar a la API `db2HistoryGetEntry` para obtener entradas de los registros del histórico de la base de datos.
 - 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 un valor `SQL_RC_NOMORE` de `SQLCODE`, significa que se han recuperado todas las entradas de los registros del archivo histórico de la base de datos.
 - 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`.

db2HistoryUpdate - Actualizar una entrada de los registros del histórico de la base de datos

Actualiza la ubicación, el tipo de dispositivo o el comentario en una entrada de los registros del histórico de la base de datos.

Nota: Esta API solamente recibe soporte en los lenguajes de programación C, C++ o Java. En los lenguajes de programación COBOL, FORTRAN y REXX ya no recibe soporte. Puede emitir una consulta para acceder a registros del histórico de la base de datos utilizando la vista administrativa DB_HISTORY.

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- DBADM

Conexión necesaria

Base de datos. Para actualizar entradas de los registros del histórico de base de datos de 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;
```

db2HistoryUpdate - Actualizar una entrada de los registros del histórico de la base de datos

```
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 el nivel de release de la estructura que se ha pasado 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 selectiva
A	Tivoli Storage Manager
U	Salida de usuario
P	Área de conexión 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 al realizar llamadas al mandato **PRUNE HISTORY**, ejecutar el procedimiento **ADMIN_CMD** con **PRUNE HISTORY**, realizar llamadas a la API **db2Prune** ni al ejecutar el recorte de los registros del histórico de la base de datos 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 de los registros del histórico de la base de datos.

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 de los registros de histórico de la base de datos 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 se sustituye y no puede volver a crearse. Estos cambios no se registran en el archivo de anotaciones.

db2HistoryUpdate - Actualizar una entrada de los registros del histórico de la base de datos

El propósito principal de los registros del 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 utilidad de restauración utilizará y hará referencia al histórico de imágenes de copia de seguridad y a sus ubicaciones para satisfacer 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.

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, necesita una de las autorizaciones siguientes:
 - DATAACCESS
 - 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:
 - DATAACCESS
 - 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 las opciones REPLACE o REPLACE_CREATE, necesita una de las autorizaciones siguientes:
 - DATAACCESS
 - Privilegio CONTROL sobre la tabla o vista
 - Privilegio INSERT, SELECT y DELETE sobre la tabla o vista
- IMPORT en una nueva tabla, utilizando la opción CREATE o REPLACE_CREATE, necesita una de las autorizaciones siguientes:
 - 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 para el esquema, si el nombre de esquema de la tabla hace referencia a un esquema existente
- IMPORT en una tabla o jerarquía que no existe, utilizando las opciones CREATE o REPLACE_CREATE, necesita una de las autorizaciones siguientes:
 - 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, necesita una de las autorizaciones siguientes:
 - DATAACCESS
 - 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 *piImportInfoOut;
    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;
```

db2Import - Importar datos a una tabla, jerarquía, apodo o vista

```
    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 db2gImportOut *piImportInfoOut;
    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. Puntero a 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 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 LOB. Este parámetro no es válido cuando se importa a un apodo.

piDataDescriptor

Entrada. Puntero a una estructura `sqldcol` que contiene información sobre las columnas que se seleccionan para importar 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 ha establecido en `SQL_METH_D`, se llevará a cabo la selección de columnas por omisión desde el archivo de entrada externo. En este caso, se hará

db2Import - Importar datos a una tabla, jerarquía, apodo o vista

caso omiso tanto del número de columnas como de la matriz de especificación de columnas. Para los archivos DEL o IXF, 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 {nombret[(lista-columnast)] |
[ALL TABLES | (nombret[(lista-columnast)][, nombret[(lista-columnast)]])]}
[IN] HIERARCHY {STARTING nombret | (nombret[, nombret])}
[UNDER nombre-subtabla | 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.

nombret

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 con una versión anterior del producto DB2 instalado, en que se debe especificar un nombre calificado o no calificado. Si es una vista, no puede ser una vista de sólo lectura.

lista-columnast

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.

nombre-subtabla

Especifica una tabla padre cuando se crean una o más subtablas en la opción CREATE.

ALL TABLES

Palabra clave implícita solo 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.

db2Import - Importar datos a una tabla, jerarquía, apodo o vista

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 **tname** y **tcolumn-list** 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 **tcolumn-list** y las columnas externas (tanto especificadas como implícitas) se comparan de acuerdo con su posición en la lista o en la estructura (los datos de la primera columna especificados en la estructura sqldcol se insertan en el campo de tabla o de vista correspondiente al primer elemento de **tcolumn-list**).

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.

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 se 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 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:

db2Import - Importar datos a una tabla, jerarquía, apodo o vista

SQLU_INITIAL

Llamada inicial. Este valor debe utilizarse en la primera llamada a la API. Si se devuelve la llamada inicial o cualquier llamada posterior y requiere que la aplicación que realiza la llamada lleve a cabo alguna acción antes de completar la operación de importación solicitada, la acción del emisor 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 que había solicitado el programa de utilidad ha finalizado y que el programa de utilidad puede seguir 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 a la estructura db2ImportIn.

poImportInfoOut

Salida. Puntero a 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 ello, 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 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 db2ImportIn

iRowcount

Entrada. Número de registros físicos a cargar. Permite a un usuario cargar solamente las primeras filas de **iRowcount** en un archivo. Si **iRowcount** es 0, el proceso de importación intentará procesar todas las filas del archivo.

db2Import - Importar datos a una tabla, jerarquía, apodo o vista

iRestartcount

Entrada. Número de registros que hay que saltarse antes de insertar o actualizar registros. Es funcionalmente equivalente al parámetro **iSkipcount**. Los parámetros **iRestartcount** y **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 operación de confirmación cada vez que se importan registros **piCommitcount**. El valor NULL especifica el número de confirmación por omisión, que es cero para las importaciones fuera de línea y AUTOMATIC para las importaciones 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 requiere 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 independientemente del 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 ha de 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

db2Import - Importar datos a una tabla, jerarquía, apodo o vista

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 el parámetro **piCommitCount** no se haya utilizado. 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 debe especificarse 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 include 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.

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 include 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".

db2Import - Importar datos a una tabla, jerarquía, apodo o vista

piXdsArgs

Entrada. Puntero a la estructura db2DMUXmlValidateXds, que representa argumentos que se corresponden con la cláusula CLP "XMLVALIDATE USING XDS".

Este parámetro sólo se aplica cuando el parámetro **iUsing** de la misma estructura se ha establecido en DB2DMU_XMLVAL_XDS.

piSchemaArgs

Entrada. Puntero a la estructura db2DMUXmlValidateSchema, que representa argumentos que se corresponden con la cláusula CLP "XMLVALIDATE USING SCHEMA".

Este parámetro sólo se aplica cuando el parámetro **iUsing** de la misma estructura se ha establecido en DB2DMU_XMLVAL_SCHEMA.

Parámetros específicos de la estructura de datos db2glImportStruct

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 hayan suprimido 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,

db2Import - Importar datos a una tabla, jerarquía, apodo o vista

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 utilice INSERT con el parámetro **iRestartcount** establecido en el número de filas importadas satisfactoriamente.

Por omisión, las operaciones COMMIT automáticas no se realizan para las opciones INSERT ni INSERT_UPDATE. No obstante, sí se realizan si el parámetro ***piCommitcount** es distinto de 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 ni REPLACE_CREATE con una tabla de objetos si ésta tiene otros dependientes distintos de ella misma o con una vista de objeto si la tabla base tiene dependientes (incluida ella misma). Para sustituir una tabla o una vista de este tipo, realice las acciones siguientes:

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.

Utilizar valores de **piDataDescriptor** que no sean valores por omisión o especificar una lista explícita de columnas de tabla en **piLongActionString**, hace que la importación en una base de datos remota sea más lenta.

Antes de poder importar datos en los formatos de archivo ASC o DEL debe existir la tabla o la jerarquía de base de datos; sin embargo, si la tabla aún no existe, las

db2Import - Importar datos a una tabla, jerarquía, apodo o vista

opciones IMPORT CREATE o IMPORT REPLACE_CREATE crearán la tabla al importar datos desde un archivo PC/IXF. Para las tablas de tipo, IMPORT CREATE también puede crear 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 se soporta 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, solo 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, a una tabla creada temporal ni 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.

db2Import - Importar datos a una tabla, jerarquía, apodo o vista

- No se soporta la importación de archivos PC/IXF con un 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 sobre el apodo participante. Debe asegurarse de que el apodo que desea utilizar cuando realiza una operación de importación ya exista.

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 base de datos particionada, 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 autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- 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;
```


db2Inspect - Inspeccionar la base de datos para comprobar la integridad de la arquitectura

```
    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 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 include) 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` indicados anteriormente, representan un límite en el número de errores, en lugar del número de páginas con error, de las que se debe 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 - Inspeccionar la base de datos para comprobar la integridad de la arquitectura

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 en 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 include) 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 include) 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. Se aconseja finalizar la unidad de trabajo confirmando los cambios con la sentencia COMMIT o la retrotrayendo los cambios con la sentencia ROLLBACK antes de iniciar la operación de inspección.

El proceso de inspección graba 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 base de datos particionada, 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

db2Inspect - Inspeccionar la base de datos para comprobar la integridad de la arquitectura

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, retrotrae inmediatamente todas las transacciones activas y establece la instancia en 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. Tras completarse las tareas de administración, puede movilizar la instancia utilizando la API db2InstanceUnquiesce. Esta API permite que otros usuarios puedan conectarse con las bases de datos que se encuentran dentro de la instancia sin tener que concluir y volver a iniciar la instancia.

En esta modalidad, solamente los grupos o usuarios con autorización DBADM, SYSADM, SYMAINT o SYSCTRL tendrán acceso a la base de datos y a sus objetos.

Autorización

Una de las autorizaciones 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;
    db2UInt32 iOptions;
} 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;
db2Uint32 iQOptions;
} db2gInsQuiesceStruct;

```

Parámetros de la API db2InstanceQuiesce

versionNumber

Entrada. Especifica la versión y 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 de un usuario a quien se permitirá acceder a la instancia mientras esté 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 base de datos particionada, se utilizará el valor especificado por el parámetro de configuración de gestor de bases de datos **start_stop_time**.

iQOptions

Entrada. Especifica las opciones de inmovilización de instancia. Los valores válidos (definidos en el archivo de cabecera sqlenv, ubicado en el directorio include) son:

db2InstanceQuiesce - Inmovilizar instancia

DB2INSQUIESCE_RESTRICTEDACCESS

La instancia se inmoviliza con la opción **RESTRICTED ACCESS** para impedir que las bases de datos se activen para realizar la comprobación de las autorizaciones.

Parámetros específicos de la estructura de datos db2glnsQuiesceStruct

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 la instancia local o remota de DB2.

Ámbito

Esta API puede emitirse respecto a un único miembro, lista de miembros o, en general, respecto a todos los miembros. En un entorno DB2 pureScale, esta API se puede emitir respecto a un sistema principal específico para iniciar los servicios de instancia en esa máquina.

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT

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;
```

db2InstanceStart - Iniciar instancia

```
        char *piNetName;
        db2Uint32 iTblspaceType;
        db2NodeType iTblspaceNode;
        db2Uint32 iIsComputer;
        char *piComputer;
        char *piUserName;
        char *piPassword;
        db2QuiesceStartStruct iQuiesceOpts;
    } db2StartOptionsStruct;

typedef SQL_STRUCTURE db2QuiesceStartStruct
{
    db2int8 iIsQRequested;
    db2int8 iQOptions;
    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;
    db2int8 iQOptions;
    db2UInt32 iQUsrNameLen;
    char *piQUsrName;
    db2UInt32 iQGrpNameLen;
    char *piQGrpName;
    db2int8 iIsQUsrGrpDef;
} db2gQuiesceStartStruct;
```

Parámetros de la API db2InstanceStart**versionNumber**

Entrada. Especifica la versión y 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. Puntero al 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 TRUE o FALSE. Este parámetro debe establecerse en TRUE en caso de que se trate de un inicio remoto.

piNodeOrHostName

Entrada. Miembro 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 miembro para definir el entorno DB2 (MPP solamente). Este archivo se ejecuta antes de que se inicien los miembros. El valor por omisión es db2profile.

db2InstanceStart - Iniciar instancia

iIsNodeNum

Entrada. Indica si se especifica un número de miembro o CF. Si se especifica, el mandato de inicio sólo afecta al miembro especificado o al CF.

iNodeNum

Entrada. Número de miembro o CF.

iOption

Entrada. Especifica una acción. Los valores válidos para **OPTION** (definidos en el archivo de cabecera `sqlenv`, situado en el directorio `include`) son: are:

SQLLE_NONE

Emite la operación **db2start** normal.

SQLLE_ADDNODE

Emite el mandato **ADD NODE**. Esta opción no está soportada en un entorno DB2 pureScale.

SQLLE_RESTART

Emite el mandato **RESTART DATABASE**. Esta opción no está soportada en un entorno DB2 pureScale.

SQLLE_RESTART_PARALLEL

Emita el mandato **RESTART DATABASE** para la ejecución en paralelo. Esta opción no está soportada en un entorno DB2 pureScale.

SQLLE_STANDALONE

Inicia el nodo en la modalidad **STANDALONE**.

SQLLE_HOST

Inicia la instancia en un sistema principal.

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 o una lista de nombres de red delimitada por comas.

iTblspaceType

Entrada. Este parámetro solo se utiliza para las operaciones **ADD MEMBER**. Especifica el tipo de definiciones de espacios de tablas temporales del sistema que se debe utilizar para el miembro que se está añadiendo. Los valores válidos son:

SQLLE_TABLESPACES_NONE

No crear ningún espacio de tablas temporal del sistema.

SQLLE_TABLESPACES_LIKE_NODE

Los contenedores para los espacios de tablas temporales del sistema deben ser los mismos que los utilizados para el miembro especificado.

SQLLE_TABLESPACES_LIKE_CATALOG

Los contenedores para los espacios de tablas temporales del sistema deben ser los mismos que los utilizados para el miembro de catálogo de cada base de datos.

iTblspaceNode

Entrada. Especifica el número de miembro del que deben obtenerse definiciones de espacios de tablas temporales del sistema. El número de miembro debe existir en el archivo de configuración `db2nodes.cfg`, y solo se utiliza si el campo `tblspace_type` se ha establecido en `SQLLE_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 inicio de sesión. Sólo es válido en el sistema operativo Windows.

piPassword

Entrada. Contraseña que corresponde al nombre de usuario de la cuenta de inicio de sesión.

iQuiesceOpts

Entrada. Puntero a la estructura `db2QuiesceStartStruct`.

Parámetros de la estructura de datos `db2QuiesceStartStruct`

iIsQRequested

Entrada. Indicador establecido en `TRUE` o `FALSE`. Este parámetro debe establecerse en `TRUE` si se solicita la inmovilización.

iQOptions

Entrada. Especifica las opciones de inmovilización de instancia. Los valores válidos (definidos en el archivo de cabecera `sqlenv`, ubicado en el directorio `include`) son:

DB2INSQUIESCE_RESTRICTEDACCESS

La instancia se inicia en `ADMIN MODE` con la opción `RESTRICTED ACCESS` para impedir que las bases de datos se activen para realizar la comprobación de las autorizaciones.

piQUserName

Entrada. Nombre de usuario inmovilizado.

piQGrpName

Entrada. Nombre del grupo inmovilizado.

iIsQUsrGrpDef

Entrada. Indicador establecido en `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 `db2glInstanceStStruct`

iRemoteInstLen

Entrada. Especifica la longitud, en bytes, de `piRemoteInstName`.

db2InstanceStart - Iniciar instancia

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**.

Ejemplos

El siguiente ejemplo muestra el inicio de una instancia:

```
struct sqlca sqlca; // sqlca to carry the sqlcode
struct db2InstanceStartStruct instanceStartStruct;
struct db2StartOptionsStruct startOptions;

instanceStartStruct.iIsRemote = FALSE; // demo local instance
instanceStartStruct.piRemoteInstName = NULL;
instanceStartStruct.piStartOpts = &startOptions;
```

En un entorno DB2 pureScale, puede establecer opciones individuales según el tipo de operaciones que desee ejecutar respecto a la instancia. Por ejemplo:

- Iniciar la instancia en un sistema principal

```
startOptions.iOption = SQLE_HOST;
startOptions.iIsHostName = TRUE;
strcpy(startOptions.piHostName, "ca1");
```

- Iniciar un miembro

```
startOptions.iIsNodeNum = TRUE;
startOptions.iNodeNum = 10;
```

- Iniciar un recurso de almacenamiento en antememoria de clúster

```
startOptions.iIsNodeNum = TRUE;
startOptions.iNodeNum = 40;
```

- Ejecute el mandato global **db2start** respecto a la instancia

```
instanceStartStruct.piStartOpts = NULL;
```

Finalmente, debe invocar a la API `db2InstanceStart` para iniciar la instancia.

```
db2InstanceStart(db2Version1010, &instanceStartStruct, &sqlca);
```


db2InstanceStop - Detener instancia

Detiene la instancia local o remota de DB2.

Ámbito

Esta API puede emitirse respecto a un único miembro, lista de miembros o, en general, respecto a todos los miembros. En un entorno DB2 pureScale, esta API se puede emitir respecto a un sistema principal específico para detener los servicios de instancia en esa máquina.

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT

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 iIsHostName;
    char *piHostName;
    db2UInt32 iStopOption;
    db2UInt32 iCallerac;
    db2Sint32 iQuiesceDeferMinutes;
} db2StopOptionsStruct;
```

db2InstanceStop - Detener instancia

```
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 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 TRUE o FALSE. Este parámetro debe establecerse en TRUE en caso de que se trate de un inicio remoto.

piRemoteInstName

Entrada. Puntero al 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 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 de perfil que se ejecutó en el inicio para definir el entorno DB2. 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 miembro. Los valores posibles son TRUE y FALSE. Si se especifica, el mandato de detención solo afecta al miembro especificado.

iNodeNum

Entrada. Número de miembro.

iIsHostName

Entrada. Indica si se especifica un nombre de sistema principal.

piHostName

Entrada. Puntero al nombre del sistema principal.

iStopOption

Entrada. Opción. Los valores válidos son:

SQLQ_NONE

Emite la operación **db2stop** normal.

SQLQ_FORCE

Emite el mandato **FORCE APPLICATION (ALL)**.

SQLQ_DROP

Descarta el miembro del archivo de configuración db2nodes.cfg.

SQLQ_QUIESCE

Inmoviliza el miembro especificado.

SQLQ_HOST

Cierra la instancia en un sistema principal.

iCallerac

Entrada. Este campo solamente es válido para el valor **SQLQ_DROP** del campo OPTION. Los valores válidos son:

SQLQ_DROP

Llamada inicial. Es el valor por omisión.

SQLQ_CONTINUE

Llamada posterior. Continuar el proceso después de un mensaje de solicitud.

SQLQ_TERMINATE

Llamada posterior. Terminar el proceso después de un mensaje de solicitud.

iQuiesceDeferMinutes

Entrada. Indica el número de minutos tras los que las aplicaciones se ven expulsadas del miembro.

db2InstanceStop - Detener instancia

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**.

Ejemplos

El siguiente ejemplo muestra el cierre de una instancia.

```
struct sqlca sqlca; // sqlca to carry the sqlcode
struct db2InstanceStopStruct instanceStopStruct;
struct db2StopOptionsStruct stopOptions;

instanceStopStruct.iIsRemote = FALSE; // demo local instance
instanceStopStruct.piRemoteInstName = NULL;
instanceStopStruct.piStopOpts = &stopOptions;
```

En un entorno DB2 pureScale, puede establecer opciones individuales según el tipo de operaciones que desee ejecutar respecto a la instancia. Por ejemplo:

- Cierre de una instancia en un sistema principal

```
stopOptions.iOption = SQLE_HOST;
stopOptions.iIsHostName = TRUE;
strcpy(stopOptions.piHostName, "ca1");
```

- Cierre de un sistema principal con la opción force

```
stopOptions.iOption = SQLE_HOST | SQLE_FORCE;
stopOptions.iIsHostName = TRUE;
strcpy(stopOptions.piHostName, "ca1");
```

- Cierre de un miembro

```
stopOptions.iIsNodeNum = TRUE;
stopOptions.iNodeNum = 10;
```

- Cierre de un miembro con la opción force

```
stopOptions.iOption = SQLE_FORCE;
stopOptions.iIsNodeNum = TRUE;
stopOptions.iNodeNum = 10;
```

- Cierre de un recurso de almacenamiento en antememoria de clúster

```
stopOptions.iIsNodeNum = TRUE;
stopOptions.iNodeNum = 40;
```

- Cierre de un recurso de almacenamiento en antememoria de clúster con la opción force

```
stopOptions.iOption = SQLE_FORCE;
stopOptions.iIsNodeNum = TRUE;
stopOptions.iNodeNum = 40;
```

- Ejecute el mandato global **db2stop** respecto a la instancia

```
instanceStopStruct.piStopOpts = NULL;
```

- Ejecute el mandato global **db2stop** con la opción force

db2InstanceStop - Detener instancia

```
stopOptions.iOption = SQLE_FORCE;
```

Finalmente, debe invocar la API db2InstanceStop para cerrar la instancia.

```
db2InstanceStop(db2Version1010, &instanceStopStruct, &sqlca);
```

db2InstanceUnquiesce - Movilizar instancia

Moviliza todas las bases de datos de la instancia.

Autorización

Una de las autorizaciones 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 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 el nivel de release de la estructura que se ha pasado 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.

db2LdapCatalogDatabase - Registrar la base de datos en el servidor LDAP

piDatabaseName

Entrada. Especifique el nombre de la base de datos a catalogar. Este 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 permiten caracteres de retorno de carro ni 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 autenticación de DCE no se puede utilizar. La autenticación 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 autenticación 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 soporta LDAP. En ese caso, el administrador tiene que registrar manualmente cada base de datos en LDAP para permitir a los clientes que soportan LDAP acceder a la base de datos sin tener que catalogarla localmente en cada máquina cliente.
- La aplicación quiere utilizar un nombre diferente para conectarse a la base de datos. En este caso, el administrador tiene que catalogar la base de datos utilizando un nombre de alias diferente.

db2LdapCatalogDatabase - Registrar la base de datos en el servidor LDAP

- Durante la operación **CREATE DATABASE IN LDAP**, el nombre de la 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 * pParamStruct,
    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 el nivel de release de la estructura que se ha pasado 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.

db2LdapCatalogNode - Proporcionar un alias para el nombre de nodo en el servidor LDAP

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 * pParamStruct,
    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 el nivel de release de la estructura que se ha pasado 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 el nivel de release de la estructura que se ha pasado como segundo parámetro, **pParamStruct**.

db2LdapRegister - Registrar el servidor DB2 en el servidor LDAP

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 permiten caracteres de retorno de carro ni 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

db2LdapRegister - Registrar el servidor DB2 en el servidor LDAP

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` (por omisión)
- `SQL_CPIC_SECURITY_SAME`
- `SQL_CPIC_SECURITY_PROGRAM`

db2LdapRegister - Registrar el servidor DB2 en el servidor LDAP

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 el nivel de release de la estructura que se ha pasado 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. Este 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 el nivel de release de la estructura que se ha pasado como segundo parámetro, **pParmStruct**.

pParmStruct

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 el nivel de release de la estructura que se ha pasado como segundo parámetro, **pParamStruct**.

db2LdapUpdate - Actualizar los atributos del servidor DB2 en el servidor LDAP

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 permiten caracteres de retorno de carro ni 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

db2LdapUpdate - Actualizar los atributos del servidor DB2 en el servidor LDAP

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_TCP4 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 (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 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

db2LdapUpdateAlternateServerForDB - Actualizar el servidor alternativo de la base de datos en el servidor LDAP

nombre de nodo debe existir en LDAP. El IBM Data Server Runtime Client 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 disponer 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. 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.

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 de archivo totalmente calificado, cursor o conexión con nombre.

Autorización

Una de las autorizaciones siguientes:

- DATAACCESS
- 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.

Si se especifica la opción FORCE, se necesita la autorización SYSADM.

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);
```

db2Load - Cargar datos en una tabla

```
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;
}

```

db2Load - Cargar datos en una tabla

```
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
originalmente (carga
desde el cursor sólo) */

```

Parámetros de la API db2Load

versionNumber

Entrada. Especifica la versión y 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 conexión de memoria o sentencias de SQL.

La información proporcionada en esta estructura depende del valor del campo **media_type**. Los valores válidos (definidos en el archivo de cabecera sqlutil, ubicado en el directorio include) 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 **sessions** 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_vendedor, donde el nombre de archivo es el

db2Load - Cargar datos en una tabla

identificador exclusivo para los datos que cargar. Solamente deberá haber una entrada `sqlu_vendor`, independientemente del valor de **sessions**. El campo **sessions** indica el número de sesiones TSM que 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 **filename** es el identificador exclusivo para los datos que 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 **sessions** debe establecerse en el valor 1.

piLobPathList

Entrada. Puntero a la estructura `sqlu_media_list`. Para los tipos de archivo IXF, ASC y DEL, una lista de vías de 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`, ubicado en el directorio `include`) 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 **filename** es el identificador exclusivo para los datos que cargar. Solamente deberá haber una entrada `sqlu_vendor`, independientemente del valor de **sessions**. El campo **sessions** indica el número de sesiones TSM que 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 compartido, y **filename** es el identificador exclusivo para los datos que cargar. Solamente deberá haber una entrada `sqlu_vendor`, independientemente del valor de **sessions**. El campo **sessions** indica el número de otras sesiones de proveedor que 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 `sqlldcol` 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 fuente. 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 normas para los archivos `DEL`, excepto que `SQL_METH_N` indica que los nombres de columna de archivo deben proporcionarse en la estructura `sqlldcol`.

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 tdbname [(column_list)]
[FOR EXCEPTION e_tdbname]"
```

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 pendiente de carga, esta opción no afecta al estado de los espacios de tablas.

La opción de terminación de carga no eliminará un estado de pendiente de copia de seguridad de los espacios de tablas.

db2Load - Cargar datos en una tabla

tbname

El nombre de la tabla en la que deben cargarse los datos. La tabla no puede ser una tabla de sistema, una tabla temporal declarada ni una tabla temporal creada. 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 normas 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 se 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 archivo 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 grabarse 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. Puntero 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`, ubicado en el directorio `include`) 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 **sessions** indica 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 **sqlu_vendor**. El campo **sessions** indica 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

db2Load - Cargar datos en una tabla

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`, ubicado en el directorio `include`) 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 posterior y requiere que la aplicación que realiza la llamada lleve a cabo alguna acción antes de completar la operación de carga solicitada, la acción del emisor 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 que había solicitado el programa de utilidad ha finalizado y que el programa de utilidad puede seguir 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 `nombreamArchivo.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 `rowcnt` filas de un archivo.

iRestartcount

Entrada. Reservado para una utilización futura.

piUseTablespace

Entrada. Si se están reconstruyendo los índices, se construye una copia sombra 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

db2Load - Cargar datos en una tabla

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 sombra 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 sombra 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 omite si el usuario no especifica **INDEXING MODE REBUILD** o **INDEXING MODE AUTOSELECT**. Esta opción también se omitirá si se elige **INDEXING MODE AUTOSELECT** y la carga decide actualizar de forma incremental el índice.

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 mediante `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 el almacenamiento dinámico 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 requiere 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 beneficiarse de las ventajas del paralelismo intrapartición. 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`, ubicado en el directorio `include`) son:

SQLU_INX_AUTOSELECT

LOAD elige entre las modalidades de indexado REBUILD e INCREMENTAL.

db2Load - Cargar datos en una tabla

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 carácter 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 reúnen 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.

Durante la carga, no se recopilan estadísticas de distribución para las columnas de tipo XML.

iSetIntegrityPending

Entrada. Especifica que se coloque la tabla en estado pendiente de establecimiento de integridad. Si se especifica el valor `SQLU_SI_PENDING_CASCADE_IMMEDIATE`, el estado pendiente de establecimiento de 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 establecimiento de integridad a las tablas dependientes quedará diferida 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. 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 `include` 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
{
    db2Uuint16                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 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 include) 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, se espera que el nombre del archivo de salida en cada partición de base de datos tenga 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 *nombreamchivo.xxx*, donde *nombreamchivo* 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 grabará 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 *nombreamchivo.xxx*, donde *nombreamchivo* 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

db2Load - Cargar datos en una tabla

se produzcan en particiones de base de datos individuales. Los valores válidos (definidos en el archivo de cabecera db2ApiDf, ubicado en el directorio include) 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 darán lugar a que toda la operación resulte anómala y a la retrotracción 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** se ha 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** se ha 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 será `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 normas 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 `RECLLEN`. 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.

db2Load - Cargar datos en una tabla

iNumNodes

Entrada. El número de nodos en 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 estar 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 emisor 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 que todavía no ha cambiado el estado de la tabla en la partición de base de datos respecto al estado en el que se encontraba antes de la llamada 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, ubicado en el directorio include) 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 `include` 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 requiere una secuencia determinada, los datos deberán clasificarse 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 referencial, 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 comprobación o de referencia se ponen en el estado pendiente de establecimiento de integridad. Las tablas de resumen que se definen con REFRESH IMMEDIATE y que dependen de las tablas que se cargan, también se ponen en el estado pendiente de establecimiento de integridad. Ejecute la sentencia SET INTEGRITY para sacar las tablas del estado pendiente de establecimiento de integridad. Las operaciones de carga no se pueden llevar a cabo en tablas de resumen reproducidas.

Para los índices de clústeres, los datos deben clasificarse en el índice de clústeres 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;
}
```


db2LoadQuery - Obtener el estado de una operación de carga

```
} 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 el nivel de release de la estructura que se ha pasado 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 include) son:

DB2LOADQUERY_TABLENAME

Especifica un nombre de tabla para su utilización por la API db2LoadQuery.

db2LoadQuery - Obtener el estado de una operación de carga

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 include) 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 resumen.

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 confirmado 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).

db2LoadQuery - Obtener el estado de una operación de carga

oCurrentMPPNode

Salida. Indica qué servidor de particiones de base de datos se está consultando (solamente para la modalidad del entorno de base de datos particionada).

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 la sentencia SET INTEGRITY para sacar la tabla del 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

En esta tabla se ha realizado una modificación de reorganización recomendada. Es necesario realizar una reorganización clásica para que se pueda acceder a la tabla.

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 - Obtener el estado de una operación de carga

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 reinicialable. La tabla puede situarse en estado DB2LOADQUERY_NO_LOAD_RESTART durante una operación de avance. Esto puede producirse si realiza una operación de avance hasta un punto en el tiempo anterior al final de una operación de carga, o si realiza una operación de avance mediante 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.

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 confirmado 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 base de datos particionada).

db2LoadQuery - Obtener el estado de una operación de carga

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 include) 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 la sentencia SET INTEGRITY para sacar la tabla del 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

En esta tabla se ha realizado una modificación de reorganización recomendada. Es necesario realizar una reorganización clásica para que se pueda acceder a la tabla.

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 - Obtener el estado de una operación de carga

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 reinicialable. La tabla puede situarse en estado DB2LOADQUERY_NO_LOAD_RESTART durante una operación de avance. Esto puede producirse si realiza una operación de avance hasta un punto en el tiempo anterior al final de una operación de carga, o si realiza una operación de avance mediante 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.

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 include) 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 include) 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 resumen.

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, en bytes, del parámetro **piString**.

iLocalMessageFileLen

Entrada. Especifica la longitud, en bytes, del parámetro **piLocalMessageFile**.

**Parámetros específicos de la estructura de datos
db2gLoadQueryStru64**

iStringLen

Entrada. Especifica la longitud, en bytes, del parámetro **piString**.

iLocalMessageFileLen

Entrada. Especifica la longitud, 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 graba 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 debe recopilar 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 autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- 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 nivel de release de la estructura transferida como segundo parámetro **pParmStruct**. Para utilizar la estructura tal como se ha descrito anteriormente, 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 grabarán los datos de estado del conmutador.

iBufferSize

Entrada. Especifica el tamaño del almacenamiento intermedio de salida.

iReturnData

Entrada. Distintivo que especifica si se deben grabar 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

db2MonitorSwitches - Obtener o actualizar los valores de los conmutadores del supervisor

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 debe utilizarse `SQLM_CURRENT_NODE`.

poOutputFormat

Formato de la corriente de datos devuelta por el servidor. Será uno de los valores siguientes:

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 autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- 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 el nivel de release de la estructura que se ha pasado como segundo parámetro, **pParmStruct**.

db2Prune - Suprimir las entradas del archivo histórico o archivos de anotaciones cronológicas de la vía de acceso de anotación cronológica activa

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.

Nota: Este valor no está soportado en entornos DB2 pureScale.

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.

Nota: Este valor no está soportado en entornos DB2 pureScale.

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, es posible que no se pueda recuperar una base de datos o que se pierdan 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 selectiva 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 selectiva 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

indicación de fecha y hora

Variable del lenguaje principal que contiene una indicación de fecha y hora. Todas las entradas con indicaciones de fecha y hora iguales o anteriores a la indicación de fecha y hora proporcionada se suprimirán 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 el nivel de release de la estructura que se ha pasado 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 include).

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 - Leer registros de anotaciones cronológicas

Lee registros de anotaciones cronológicas de las anotaciones cronológicas de base de datos DB2 o consulta el gestor de anotaciones cronológicas para obtener información sobre el estado actual de la anotación cronológica.

Esta API sólo se puede utilizar con bases de datos recuperables. Una base de datos es recuperable si los parámetros de configuración de base de datos **logarchmeth1** o **logarchmeth2**, o ambos, no se han establecido en OFF.

Autorización

Una de las autorizaciones 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;
    db2LRI *piStartLRI;
    db2LRI *piEndLRI;
    char *poLogBuffer;
    db2UInt32 iLogBufferSize;
    db2UInt32 iFilterOption;
    db2ReadLogInfoStruct *poReadLogInfo;
    db2LRI *piMinRequiredLRI;
} db2ReadLogStruct;

typedef SQL_STRUCTURE db2ReadLogInfoStruct
{
    db2LRI initialLRI;
    db2LRI firstReadLRI;
    db2LRI nextStartLRI;
    db2LRI firstReusedLRI;
    db2UInt32 logRecsWritten;
    db2UInt32 logBytesWritten;
    db2UInt32 timeOfLRIReuse;
    db2TimeOfLog currentTimeValue;
    db2UInt32 futureUse1;
    db2LSN oldestInflightLSN;
} db2ReadLogInfoStruct;

typedef SQL_STRUCTURE db2TimeOfLog
{
    db2UInt32 seconds;
    db2UInt32 accuracy;
} db2TimeOfLog;
```


db2ReadLog - Leer registros de anotaciones cronológicas

```
typedef SQL_STRUCTURE db2ReadLogFilterData
{
    db2LRI          recordLRIType1;
    db2LRI          recordLRIType2;
    db2UInt32      realLogRecLen;
    db2int32       sqlcode;
} db2ReadLogFilterData;
typedef SQL_STRUCTURE db2LRI
{
    db2UInt64      lriType;
    db2UInt64      part1;
    db2UInt64      part2;
} db2LRI;

#define DB2READLOG_LRI_1  1
#define DB2READLOG_LRI_2  2
```

Parámetros de la API db2ReadLog

versionNumber

Entrada. Especifica la versión y el 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.

piStartLRI

Entrada. La LRI de inicio especifica el punto de inicio para la lectura del archivo de anotaciones. Este valor no es necesario que sea el inicio de un registro de anotaciones real.

piEndLRI

Entrada. El LRI de finalización especifica el punto de finalización para la lectura del archivo de anotaciones. Este valor debe ser mayor que el parámetro **piStartLRI** y no es necesario que sea el inicio o 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

db2ReadLog - Leer registros de anotaciones cronológicas

especificado. Este almacenamiento intermedio debe ser lo suficientemente grande para contener un solo registro de anotaciones. Como directriz, este almacenamiento intermedio deberá ser de 88 bytes como mínimo. Su tamaño máximo depende del tamaño del rango solicitado.

- Si la opción **iFilterOption** está configurada en ON, la estructura db2ReadLogFilterData se incluirá como prefijo en todos los registros de anotaciones cronológicas.
- Si la opción **iFilterOption** está configurada en OFF, cada registro de anotaciones cronológicas del almacenamiento intermedio recibirá como prefijo dos estructuras db2LRI de 24 bytes.

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 LRI especificado.

DB2READLOG_FILTER_ON

Sólo lee los registros de anotaciones cronológicas en el rango de LRI 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.

piMinRequiredLRI

Entrada. Este campo es para funciones futuras. La aplicación debe configurar este campo en 0.

Parámetros de la estructura de datos db2ReadLogInfoStruct

initialLRI

Es la primera LRI utilizada o que será utilizada por la base de datos desde que se activó.

firstReadLRI

Es la primera LRI existente en el parámetro **poLogBuffer**.

nextStartLRI

Es el inicio del siguiente registro de anotaciones que el llamador debe leer. Debido a que algunos registros de anotaciones cronológicas pueden filtrarse y no devolverse en el parámetro **poLogBuffer**, la utilización de esta LRI como inicio de la lectura siguiente en lugar del final del último registro de anotaciones cronológicas contenido en el parámetro **poLogBuffer** evita que se exploren de nuevo registros de anotaciones cronológicas que ya se han filtrado.

db2ReadLog - Leer registros de anotaciones cronológicas

firstReusedLRI

Primera LRI que se debe reutilizar debido a una operación de restauración o avance de una base de datos.

logRecsWritten

Número de registros de anotaciones grabados en el parámetro **poLogBuffer**.

logBytesWritten

Número total de bytes de datos grabados en el parámetro **poLogBuffer**.

timeOfLRIReuse

La hora en la que la LRI representada por **firstReusedLRI** se volvió a utilizar. La hora es el número de segundos transcurridos desde el 1 de enero de 1970.

currentTimeValue

La hora actual de acuerdo con la base de datos.

futureUse1

Este campo es para funciones futuras. La versión actual de DB2 siempre devuelve un valor de 0.

oldestInflightLSN

El LSN de la transacción en curso más antigua.

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.

Parámetros de la estructura de datos db2ReadLogFilterData

Salida. La estructura `db2ReadLogFilterData` retiene los metadatos del registro de anotaciones cronológicas, de la forma siguiente:

recordLRI

LRI del registro de anotaciones cronológicas siguiente. Hay dos LRI en la estructura `db2ReadLogFilterData`:

- LRI `recordLRIType1` del registro de anotaciones cronológicas. Esta es un LRI del tipo 1.
- LRI `recordLRIType2` del registro de anotaciones cronológicas. Esta es un LRI del tipo 2.

realLogRecLen

La longitud del registro físico de anotaciones cronológicas en los archivos de anotaciones cronológicas de DB2.

sqlcode

El valor de este campo será distinto de cero cuando se produzca un error al intentar descomprimir la imagen de fila comprimida del registro de anotaciones cronológicas. Si se produce un error, contendrá un entero que representa el código de SQL asociado con el error. Para los errores permanentes, se devolverá probablemente `SQL0204N`. Si se vuelve a enviar la petición API, puede devolverse el mismo error. Para los errores

db2ReadLog - Leer registros de anotaciones cronológicas

transitorios, el código de SQL devuelto se corresponderá con la causa del error, que podría o no requerir acciones de rectificación por parte del usuario.

Parámetros de la estructura de datos db2LRI

db2LRI es una estructura identificadora del registro de anotaciones cronológicas que se utiliza para describir qué registros de anotaciones cronológicas desea leer el usuario y qué registros de anotaciones cronológicas se han leído, de este modo:

IriType

Existen dos tipos de LRI en la estructura db2LRI para la configuración de **IriType**:

- DB2READLOG_LRI_1 se utiliza al leer de forma secuencial en los archivos de anotaciones.
- DB2READLOG_LRI_2 se utiliza para identificar de forma exclusiva el registro de anotaciones cronológicas en la corriente de anotaciones cronológicas al que pertenece.

Notas de uso

Si la acción solicitada es leer el archivo de anotaciones, debe proporcionar un rango de identificadores de registros de anotaciones cronológicas 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 LRI solicitado, y devuelve registros de anotaciones cronológicas asociados a tablas definidas con la cláusula DATA CAPTURE CHANGE, 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.

Nota: La API db2ReadLog no puede utilizarse para leer anotaciones cronológicas grabadas antes de una operación de descarte de miembro. El intento de leer estas anotaciones cronológicas provocará un evento de SQL1273N.

Para utilizar el Lector de anotaciones asíncronas, primero consulte las anotaciones de base de datos para obtener un LRI inicial válido. Después de la consulta de la estructura de información de anotación cronológica de lectura (db2ReadLogInfoStruct) incluirá un LRI de inicio válido (en el miembro **initialLRI**), para utilizarlo en una llamada de lectura. El valor utilizado como LRI final en una lectura puede ser uno de los valores siguientes:

- Un valor mayor que initialLRI
- (0000 0000 0000 0001, FFFF FFFF FFFF FFFF, FFFF FFFF FFFF FFFF) (IriType, part1, part2) que se interpreta por parte del lector de anotaciones cronológicas asíncronas como el final de la anotación cronológica actual.

Los registros de anotaciones propagables que se leen dentro del rango comprendido entre el LRI inicial y el LRI final se devuelven en el almacenamiento intermedio de anotaciones. Si la opción **ifilterOption** está establecida en DB2READLOG_FILTER_ON, el LRI se sustituye por la estructura de datos db2ReadLogFilterData, en el almacenamiento intermedio. Encontrará descripciones de los diversos registros de anotaciones cronológicas de DB2 devueltos por db2ReadLog en el tema "Registros de anotaciones cronológicas de DB2".

db2ReadLog - Leer registros de anotaciones cronológicas

Para leer el siguiente registro de anotaciones secuencial después de la lectura inicial, utilice el campo **nextStartLRI** devuelto en la estructura `db2ReadLogStruct`. Emita de nuevo la llamada con este nuevo LRI inicial y un LRI 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` funciona 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 a la API `db2ReadLog` desde una aplicación puede producirse un error cuando la aplicación se desconecta de la base de datos si no se confirma o retrotrae antes de la desconexión:

- Se puede producir un error `CLI0116E` si se llama a la API `db2ReadLog` desde una aplicación de CLI.
- Se puede producir un error `SQL0428N` si se llama a la API `db2ReadLog` 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 llamar a la API `db2ReadLog`.

Solución 2: emita una sentencia `COMMIT` o `ROLLBACK` después de llamar a la API `db2ReadLog` y antes de desconectarla de la base de datos.

En un entorno DB2 pureScale, la secuencia de llamadas para la API `db2ReadLog` permanece sin cambios desde las versiones anteriores.

Para comenzar a leer los registros de anotaciones cronológicas, el emisor puede especificar `DB2READLOG_READ` o `DB2READLOG_READ_SINGLE`. El LRI aprobado por el emisor no tiene por qué existir en una corriente de anotaciones cronológicas, en cuyo caso el gestor de bases de datos devuelve el siguiente registro de anotaciones cronológicas.

La API `db2ReadLog` devuelve dos estructuras LRI en el almacenamiento intermedio de salida. La primera estructura LRI se utiliza para leer las anotaciones cronológicas secuencialmente y la segunda estructura LRI identifica el registro de anotaciones cronológicas en la corriente de anotaciones cronológicas a la que pertenece.

En un entorno DB2 pureScale, la base de datos tiene una corriente de anotaciones cronológicas para cada miembro. La API `db2ReadLog` admite la lectura de todas las corrientes de anotaciones cronológicas y la combinación de estas para devolver los registros de anotaciones cronológicas al emisor. Esto provoca la devolución de los cambios en los datos en el orden en el que se aplicaron.

El servidor de datos de DB2 aún puede leer los archivos de anotaciones cronológicas de miembros sin actividad o miembros fuera de línea. La API

db2ReadLog - Leer registros de anotaciones cronológicas

db2ReadLog puede esperar a que se complete la recuperación de anomalía general de miembro en un miembro antes de la devolución con más registros de anotaciones cronológicas. En el caso de que la recuperación de anomalía general de miembro tarde mucho tiempo en completarse, se devolverá SQL1015N. Los usuarios deben volver a enviar la llamada db2ReadLog en este caso.

En situaciones específicas, el tiempo de registro de anotaciones cronológicas de confirmación podría ser ligeramente anterior a la confirmación producida en la aplicación.

Compatibilidad con versiones anteriores de la API db2ReadLog

Cualquier aplicación que pase una versión de la API db2ReadLogStruct, que precede a DB2 versión 9.8, a un servidor DB2 versión 9.8 o posterior fallará con SQL2032N, que indica que el parámetro db2ReadLogStruct no es válido. Debido a las grandes diferencias en la estructura del registro de anotaciones cronológicas y la introducción de db2LRI, las aplicaciones anteriores deben volver a escribirse y a compilarse mediante las nuevas estructuras db2ReadLog.

Los clientes de DB2 pueden utilizar la API db2ReadLogStruct anterior de DB2 versión 9.7 y pasarla a un servidor DB2 V9.7, o bien utilizar la API db2ReadLogStruct para DB2 versión 9.8 y pasarla a un servidor DB2 V9.8 o DB2 V10.

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 o 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;
    db2LSN *piStartLSN;
    db2LSN *piEndLSN;
    char *poLogBuffer;
    db2UInt32 iLogBufferSize;
    char *piReadLogMemPtr;
    db2ReadLogNoConnInfoStruct *poReadLogInfo;
} db2ReadLogNoConnStruct;

typedef SQL_STRUCTURE db2ReadLogNoConnInfoStruct
{
    db2LSN firstAvailableLSN;
    db2LSN firstReadLSN;
    db2LSN nextStartLSN;
    db2UInt32 logRecsWritten;
    db2UInt32 logBytesWritten;
    db2UInt32 lastLogFullyRead;
    db2TimeOfLog currentTimeValue;
} db2ReadLogNoConnInfoStruct;
```

Parámetros de la API db2ReadLogNoConn

versionNumber

Entrada. Especifica la versión y el nivel de release de la estructura transferida como segundo parámetro, **pDB2ReadLogNoConnStruct**.

pDB2ReadLogNoConnStruct

Entrada. Puntero a la estructura db2ReadLogNoConnStruct.

db2ReadLogNoConn - Leer las anotaciones cronológicas de la base de datos sin una conexión de base de datos

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 48 bytes como mínimo. Su tamaño máximo depende del tamaño del rango solicitado.

Cada registro de anotaciones cronológicas del almacenamiento intermedio lleva como prefijo un número de secuencia de anotaciones cronológicas (LSN) de ocho bytes que representa el LSN del siguiente registro de anotaciones cronológicas.

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 a 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 cronológicas grabados en el campo del almacenamiento intermedio de anotaciones, **poLogBuffer**.

logBytesWritten

Número de bytes grabados 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 no está soportada en entornos DB2 pureScale. Cualquier intento de invocarla fallará con SQL1419N.

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 se debe modificar.

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 los valores siguientes:

- Un valor mayor que el LSN inicial especificado por el llamador.
- FFFF FFFF FFFF, que el lector de archivos de anotaciones cronológicas asíncrono interpreta como el final de las anotaciones cronológicas 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

db2ReadLogNoConn - Leer las anotaciones cronológicas de la base de datos sin una conexión de base de datos

db2ReadLogNoConnInfoStruct. Emita de nuevo la llamada con este nuevo LSN inicial y un LSN final válido; se leerá el siguiente bloque de registros. Un código de sqlca de `SQLU_RLOG_READ_TO_CURRENT` significa que el lector de anotaciones cronológicas ha leído hasta el final de los archivos de anotaciones cronológicas 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.

Nota: Esta API no admite el formato de imágenes comprimidas RAW en registros de anotaciones cronológicas, que requiere una conexión a la base de datos. Para ello, utilice la API `db2ReadLog` en su lugar. Dado que no se soporta el formato de las imágenes de fila comprimidas, esta API devuelve la imagen de fila comprimida sin modificarla.

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 *piDbPartitionName;
    db2LogStreamIDType *piLogStreamNum;
    db2UInt32 iReadLogMemoryLimit;
    char **poReadLogMemPtr;
} db2ReadLogNoConnInitStruct;
```

Parámetros de la API db2ReadLogNoConnInit

versionNumber

Entrada. Especifica la versión y nivel de release de la estructura transferida como segundo 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:

db2ReadLogNoConnInit - Inicializar la lectura de las anotaciones cronológicas de la base de datos sin una conexión de base de datos

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.

piDbPartitionName

Entrada. Nombre de la partición de la base de datos a la que pertenecen los archivos de anotaciones de recuperación que se están leyendo. Es necesario si la opción de recuperación indicada anteriormente se especifica fuera de un entorno DB2 pureScale.

piLogStreamNum

Entrada. Número de corriente de anotaciones cronológicas propietaria de los archivos de anotaciones cronológicas de recuperación que se leen. Es necesario si la opción de recuperación indicada anteriormente se especifica en un entorno DB2 pureScale.

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

db2ReadLogNoConnInit - Inicializar la lectura de las anotaciones cronológicas de la base de datos sin una conexión de base de datos

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 modificar.

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 API 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 base de datos particionada, 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.

En un entorno DB2 pureScale, la API `db2Recover` se puede invocar desde cualquier miembro.

Autorización

Para recuperar una base de datos existente, una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT

Para recuperar una base de datos nueva, una de las autorizaciones 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;
```

db2Recover - Restaurar y avanzar una base de datos

```
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 * pDB2RecoverStruct,
    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 emisor, por ejemplo, si se sabe que se han montado todos los soportes de almacenamiento necesarios para la recuperación y no es necesario que aparezcan mensajes de solicitud de los 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 - Restaurar y avanzar una base de datos

- 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. 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 Greenwich).

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 de recuperación de carga (`DB2ROLLFORWARD_LOADREC_`).

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 **logpath** 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 base de datos particionada, 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.

db2Recover - Restaurar y avanzar una base de datos

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 y pasará estos nuevos valores a la rutina de inicialización.

iComprOptionsSize

Entrada. Representa el tamaño del bloque de datos pasado como **piComprOptions**. Si **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 autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- DBADM
- SQLADM
- 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;
    char *pPartitionName;
} db2ReorgTable;

typedef SQL_STRUCTURE db2ReorgIndexesAll
{
    char *pTableName;
    char *pIndexName;
    char *pPartitionName;
```

db2Reorg - Reorganizar un índice o una tabla

```
} 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;
    db2UInt32 partitionNameLen;
    char *pPartitionName;
} db2gReorgTable;

typedef SQL_STRUCTURE db2gReorgIndexesAll
{
    db2UInt32 tableNameLen;
    char *pTableName;
    db2UInt32 indexNameLen;
    char *pIndexName;
    db2UInt32 partitionNameLen;
    char *pPartitionName;
} 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.

La reorganización de INPLACE (en línea) no está soportada en entornos DB2 pureScale. Cualquier intento de realizar una reorganización in situ en entornos DB2 pureScale fallará con SQL1419N.

DB2REORG_OBJ_INDEXESALL

Reorganizar todos los índices.

DB2REORG_OBJ_INDEX

Reorganizar un índice.

DB2REORG_RECLAIM_EXTENTS

Reorganizar una tabla de clúster multidimensional (MDC) o de agrupación en clústeres de tiempo de inserción (ITC) para reclamar extensiones vacías para el espacio de tablas.

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**. Si también se ha especificado la opción DB2REORG_RESETDICTIONARY o DB2REORG_KEEPMONITOR, las opciones se aplican al objeto de almacenamiento XML de la tabla, además de al objeto de tabla.

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**. Este parámetro no está soportado en entornos DB2 pureScale.

DB2REORG_PAUSE_ONLINE

Pausar una reorganización en línea, utilizado cuando se especifica DB2REORG_OBJ_TABLE_INPLACE como **reorgType**. Este parámetro no está soportado en entornos DB2 pureScale.

DB2REORG_STOP_ONLINE

Detener una reorganización en línea, utilizado cuando se especifica DB2REORG_OBJ_TABLE_INPLACE como **reorgType**. Este parámetro no está soportado en entornos DB2 pureScale.

DB2REORG_RESUME_ONLINE

Reanudar una reorganización en línea que está en pausa, utilizado cuando

db2Reorg - Reorganizar un índice o una tabla

se especifica DB2REORG_OBJ_TABLE_INPLACE como **reorgType**. Este parámetro no está soportado en entornos DB2 pureScale.

DB2REORG_NOTRUNCATE_ONLINE

No realizar truncamiento de tabla, utilizado cuando se especifica DB2REORG_OBJ_TABLE_INPLACE como **reorgType**. Este parámetro no está soportado en entornos DB2 pureScale.

DB2REORG_ALLOW_NONE

Sin acceso de lectura o grabación 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 grabación a la tabla. Este parámetro no está soportado cuando se especifica DB2REORG_OBJ_TABLE_OFFLINE como **reorgType**. Este parámetro recibe soporte en los entornos DB2 pureScale sólo cuando también se especifica la opción **DB2REORG_INDEX_RECLAIM_EXTENTS, DB2REORG_CLEANUP_ALL** o **DB2REORG_CLEANUP_PAGES**.

DB2REORG_ALLOW_READ

Permitir sólo el acceso de lectura a la tabla. Este parámetro recibe soporte en los entornos DB2 pureScale sólo cuando también se especifica la opción **DB2REORG_INDEX_RECLAIM_EXTENTS, DB2REORG_CLEANUP_ALL** o **DB2REORG_CLEANUP_PAGES**.

DB2REORG_CLEANUP_NONE

Este valor está en desuso en los Versión 10.1. Indica que no se necesita limpieza alguna si reorgType se ha establecido en DB2REORG_OBJ_INDEXESALL o DB2REORG_OBJ_INDEX. Si no se especifica este valor se consigue el mismo efecto, por lo que especificarlo es redundante.

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**.

Si se especifica DB2REORG_CLEANUP_ALL con DB2REORG_INDEX_REBUILD, se devuelve el error SQL2218N.

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**.

Si se especifica DB2REORG_CLEANUP_PAGES con DB2REORG_INDEX_REBUILD, se devuelve el error SQL2218N.

DB2REORG_CONVERT_NONE

Este valor está en desuso en los Versión 10.1. En anteriores releases, este valor indicaba que no se necesita conversión de índice si reorgType se ha establecido en DB2REORG_OBJ_INDEXESALL o DB2REORG_OBJ_INDEX. Este valor ha quedado obsoleto, puesto que los índices de tipo 1 dejaron de mantenerse en la Versión 9.7.

DB2REORG_RESET_DICTIONARY

Si también se ha especificado la opción DB2REORG_LONGLOB, DB2REORG_RESETDICTIONARY se aplica también al objeto de almacenamiento

db2Reorg - Reorganizar un índice o una tabla

XML de la tabla. Si el atributo COMPRESS para la tabla es YES, se crea un nuevo diccionario de compresión. Todas las filas procesadas durante la reorganización están sujetas a compresión utilizando este diccionario nuevo. Este diccionario sustituye a cualquier diccionario anterior del objeto. Si el atributo COMPRESS para la tabla es NO y el objeto de tabla o el objeto de almacenamiento XML 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 también se ha especificado la palabra clave DB2REORG_LONGLOB, DB2REORG_KEEPPDICTIONARY se aplica al objeto de tabla y al objeto de almacenamiento XML de la tabla. Si no se ha especificado DB2REORG_LONGLOB, lo siguiente se aplica únicamente al objeto de tabla.

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.

DB2REORG_INDEX_RECLAIM_EXTENTS

Reclamar extensiones del objeto de índice al espacio de tablas, utilizadas cuando se ha configurado **reorgType** en DB2REORG_OBJ_INDEXESALL o DB2REORG_OBJ_INDEX.

Si se especifica DB2REORG_INDEX_RECLAIM_EXTENTS con DB2REORG_INDEX_REBUILD, se devuelve el error SQL2218N.

DB2REORG_INDEX_REBUILD

Reconstruir los datos de índice, utilizados cuando se especifica DB2REORG_OBJ_INDEXESALL o DB2REORG_OBJ_INDEX como **reorgType**.

Si se especifica DB2REORG_INDEX_REBUILD con DB2REORG_INDEX_RECLAIM_EXTENTS, DB2REORG_CLEANUP_ALL o DB2REORG_CLEANUP_PAGES se devuelve un error, SQL2218N.

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.

db2Reorg - Reorganizar un índice o una tabla

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 aquellos casos en los que se añade una columna a una tabla (como desde ALTER TABLE ADD COLUMN) y las filas se han insertado 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 el espacio de tablas para que contenga la mayor partición de datos de la tabla. Cuando se especifica el parámetro **pPartitionName**, el espacio de tablas temporal tiene que poder contener la partición especificada.

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.

pPartitionName

Entrada. Especifica el nombre de la partición de datos que debe reorganizarse.

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

db2Reorg - Reorganizar un índice o una tabla

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.

pPartitionName

Entrada. Especifica el nombre de la partición de datos cuyos índices deben reorganizarse.

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 **pLongTempSpace**.

partitionNameLen

Entrada. Especifica la longitud, en bytes, de **pPartitionName**.

pPartitionName

Entrada. Especifica el nombre de la partición de datos que debe reorganizarse.

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**.

partitionNameLen

Entrada. Especifica la longitud, en bytes, de **pPartitionName**.

pPartitionName

Entrada. Especifica el nombre de la partición de datos para el índice.

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. Si el objetivo es reclamar espacio, la salida de RECLAIMABLE_SPACE de las funciones **ADMIN_GET_INDEX_INFO** y **ADMIN_GET_TAB_INFO** muestra cuánto espacio se puede reclamar, en kilobytes. Después puede utilizar la opción **RECLAIM_EXTENTS** de **reorgType** o **reorgFlags** para reclamar espacio en las tablas y los índices. Todo el trabajo se confirmará y todos los cursores abiertos se cerrarán durante el proceso de reorganización.

db2Reorg - Reorganizar un índice o una tabla

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 clústeres, los datos se ordenarán de acuerdo al índice de clústeres.
- 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 que puede afectar al acceso a tabla por omisión: reorgType	Tipo de distintivos aplicables que pueden afectar al acceso a tabla por omisión: reorgFlags	Modalidad de acceso seleccionada para la tabla no particionada	Modalidad de acceso seleccionada para la tabla particionada
DB2REORG_OBJ_TABLE_OFFLINE		DB2REORG_ALLOW_READ	DB2REORG_ALLOW_NONE ¹
DB2REORG_OBJ_TABLE_INPLACE		DB2REORG_ALLOW_WRITE	N/D
DB2REORG_RECLAIM_EXTENTS		DB2REORG_ALLOW_WRITE	DB2REORG_ALLOW_WRITE
DB2REORG_OBJ_INDEXESALL ²		DB2REORG_ALLOW_READ	DB2REORG_ALLOW_NONE ¹
DB2REORG_OBJ_INDEXESALL	DB2REORG_CLEANUP_ALL DB2REORG_CLEANUP_PAGES DB2REORG_INDEX_RECLAIM_EXTENTS	DB2REORG_ALLOW_READ	DB2REORG_ALLOW_READ
DB2REORG_OBJ_INDEX ²	DB2REORG_CLEANUP_ALL DB2REORG_CLEANUP_PAGES DB2REORG_INDEX_REBUILD DB2REORG_INDEX_RECLAIM_EXTENTS	N/D	DB2REORG_ALLOW_READ

Nota:

1: Si **pPartitionName** no se especifica, DB2REORG_ALLOW_NONE es el valor por omisión. Para obtener información sobre las modalidades de acceso cuando **pPartitionName** especifica un nombre de partición, consulte el mandato **REORG INDEXES/TABLE**.

2: A menos que se especifique reorgFlags para limpieza o reclamación, el valor por omisión es DB2REORG_INDEX_REBUILD.

N/D: no disponible en este momento porque no recibe soporte.

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 a una modalidad menos restrictiva. 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.
- Si se intenta realizar una reorganización de índice con acceso de lectura o de grabación en una tabla no particionada cuando es necesario reconstruir los índices, la tabla se pondrá fuera de línea mientras los índices se reconstruyen. Se graba un mensaje en el registro de notificaciones de administración y en las anotaciones cronológicas de diagnóstico acerca del cambio. En el caso de una tabla no particionada, no hay nada que reorganizar una vez que se han reconstruido los índices. Cuando se especifique DB2REORG_OBJ_INDEX para una tabla particionada, los índices de la tabla que se deban reconstruir se reconstruirán fuera de línea. Suponiendo que no se haya reconstruido ya, se reorganiza el índice especificado. Esta reorganización emplea la modalidad de acceso especificada; dicha modalidad de acceso no cambiará durante el proceso. Se graba un mensaje en el registro de notificaciones de administración y en las anotaciones cronológicas de diagnóstico sobre la reconstrucción fuera de línea de los índices.
- Para la reorganización de tablas clásica, si no se especifica DB2REORG_RESET_DICTIONARY ni DB2REORG_KEEP_DICTIONARY, el valor por omisión es DB2REORG_KEEP_DICTIONARY.
- Si la reconstrucción de una reorganización de índice sin acceso falla, alguno de los índices o todos ellos estarán no 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;
 - tablas temporales declaradas;
 - tablas temporales creadas.
- Con DB2 Versión 9.7 Fixpack 1 y releases posteriores, **pPartitionName** puede especificar un nombre de partición de datos para reorganizar una partición de datos específica de una tabla particionada de datos o los índices particionados de una partición de datos específica de una tabla particionada.

Los elementos siguientes son aplicables para una tabla particionada de datos cuando se utiliza **pPartitionName** para reorganizar los índices particionados en una partición de datos específica de una tabla particionada:

- Sólo la partición de datos especificada está restringida al nivel de modalidad de acceso. Los usuarios pueden leer las demás particiones de la tabla y grabar datos en ellas mientras los índices particionados de una partición especificada se están reorganizando.
 - Sólo se reorganizan los índices particionados para la partición especificada. Los índices no particionados de la tabla particionada no se reorganizan.
- Si hay algún índice no particionado en la tabla marcado como "no válido" o "para reconstruir", todos los índices marcados como "no válido" o "para reconstruir" se reconstruyen antes de la reorganización. De lo contrario, sólo

db2Reorg - Reorganizar un índice o una tabla

se reorganizan o reconstruyen índices particionados si el objeto de índice se marca como "no válido" o "para reconstruir".

- Cuando se limpian índices, sólo se limpian los índices particionados para la partición especificada.

Cuando se utiliza **pPartitionName** para realizar una reorganización de tabla en una partición de datos de una tabla particionada de datos, los índices no particionados afectan el acceso a la tabla:

- Si no hay índices no particionados (excepto índices de vía de acceso XML generados por el sistema) definidos en la tabla, sólo se reorganiza la partición especificada. La modalidad de acceso sólo es aplicable a la partición especificada; los usuarios pueden leer datos y grabarlos en las demás particiones de la tabla.
- Si hay índices no particionados definidos en la tabla (excluyendo los índices de vía de acceso XML generados por el sistema), la modalidad ALLOW NONE es el valor por omisión y la única modalidad de acceso soportada. En este caso, la tabla se sitúa en modalidad ALLOW NONE. Si se especifica ALLOW READ ACCESS, se devuelve SQL1548N (SQLSTATE 5U047).
- Para una tabla particionada de datos, una reorganización de tabla reconstruye los índices no particionados y los índices particionados en la tabla después de reorganizar la tabla. Si se utiliza **pPartitionName** para reorganizar una partición de datos específica de una tabla particionada de datos, una reorganización de tabla reconstruye los índices no particionados y los índices particionados únicamente para la partición especificada.

Cuando se utiliza **pPartitionName** para realizar una reorganización de tabla o de índice, si se dan las condiciones siguientes se devuelve un error:

- Si el nombre de la partición de datos no existe en la tabla indicada cuando se efectúa una reorganización de tabla de una partición de datos específica, la reorganización falla y devuelve SQL2222N con el código de razón 1.
 - Si se especifica un nombre de partición de datos al efectuar una reorganización de un índice no particionado específico definido en una tabla particionada, la reorganización falla y devuelve SQL2222N con el código de razón 2.
 - Si el nombre de partición de datos especificado sigue en estado enlazado o desenlazado cuando se efectúa una reorganización de tabla de una partición de datos, la reorganización falla y devuelve SQL2222N con el código de razón 3.
 - Si el nombre de partición de datos especificado aún está en estado enlazado o desenlazado cuando se efectúa una reorganización de índice en los índices particionados de una partición de datos, la reorganización falla y devuelve SQL2222N con el código de error 3.
- Si DB2REORG_OBJ_INDEXESALL o DB2REORG_OBJ_INDEX se ha especificado como **reorgType** y no se especifica ninguno de los siguientes **reorgFlags**:
 - DB2REORG_CLEANUP_ALL
 - DB2REORG_CLEANUP_PAGES
 - DB2REORG_INDEX_RECLAIM_EXTENTS
 - DB2REORG_INDEX_REBUILD

DB2REORG_INDEX_REBUILD se toma como valor por omisión para **reorgFlag** y los índices se reconstruyen.

db2ResetAlertCfg - Restablecer la configuración de alertas de los indicadores de salud

Restablece los valores del indicador de salud para objetos específicos a 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.

Importante: Este mandato o API ha quedado en desuso y puede que se elimine en un futuro release, ya que el supervisor de salud ha quedado en desuso en la Versión 9.7. No recibe soporte en los entornos DB2 pureScale. Para obtener más información, consulte el tema “El Supervisor de salud ha quedado en desuso” en <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

Autorización

Una de las autorizaciones 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 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 include) 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**, se establece en 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**, se establece en 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 restablece cuando **iObjType** es DB2ALERTCFG_OBJTYPE_DBM, DB2ALERTCFG_OBJTYPE_DATABASES, DB2ALERTCFG_OBJTYPE_TABLESPACES, DB2ALERTCFG_OBJTYPE_TS_CONTAINERS o cuando **piObjName** y **piDbName** son NULL. Si **iObjType** es DB2ALERTCFG_OBJTYPE_DATABASE, DB2ALERTCFG_OBJTYPE_TABLESPACE, DB2ALERTCFG_OBJTYPE_TS_CONTAINER, y se especifican **piDbName** y **piObjName** (no necesarios para la base de datos), se restablecerán los ajustes actuales para ese objeto en concreto.

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 autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- 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;
```

db2ResetMonitor - Restaurar los datos del supervisor del sistema de base de datos

```
db2UInt32 iDbAliasLength;  
db2UInt32 iVersion;  
db2int32 iNodeNumber;  
} db2gResetMonitorData;
```

Parámetros de la API db2ResetMonitor

versionNumber

Entrada. Especifica la versión y 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 debe utilizarse 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 elementos 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:

- Realizar una restauración a una base de datos con un nombre distinto al nombre de base de datos que hay en la imagen de copia de seguridad (además de poder realizar una restauración a una base de datos nueva), a excepción de una restauración instantánea donde el nombre de base de datos de la imagen de copia de seguridad debe ser el mismo.
- Restaurar bases de datos DB2 creadas en los dos releases anteriores.
- Realizar una restauración a partir de una copia de seguridad de nivel de espacio de tablas o una restauración de los espacios de tablas a partir de una imagen de copia de seguridad de base de datos.
- Transportar un conjunto de esquemas y espacios de tablas desde una imagen de copia de seguridad de base de datos hasta una base de datos (a partir de DB2 Versión 9.7 Fixpack 2).

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 autorizaciones:

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

Para restaurar en una base de datos nueva, una de las siguientes autorizaciones:

- *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;
    char *piStagingDBAlias;
    struct db2SchemaStruct *piSchemaList;
    char *piStogroup
} 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);
```

db2Restore - Restaurar una base de datos o un espacio de tablas

```
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;
    struct db2gSchemaStruct *piSchemaList;
    char *piStagingDBAlias;
    db2UInt32 iStagingDBAliasLen;
    char *piStogroup;
    db2UInt32 iStogroupLen;
} 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
{

```

db2Restore - Restaurar una base de datos o un espacio de tablas

```
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**.

piStagingDBAlias

Entrada. Serie que contiene el nombre de la base de datos de etapas que se utilizará con la opción TRANSPORT. Si este parámetro es NULL, el nombre de la base de datos de etapas se generará internamente y la base de datos se descartará una vez finalizado el transporte. Si se indica un nombre, no se descartará la base de datos de etapas.

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 la fuente especificada.

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

db2Restore - Restaurar una base de datos o un espacio de tablas

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 tablas.
- El programa de utilidad de avance asegurará que los espacios de tablas restaurados en un entorno de base de datos particionada estén sincronizados con las demás particiones de base de datos que contengan los mismos espacios de tablas. 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`.

piSchemalist

Entrada. Lista de los esquemas que se transportarán. Se utiliza con `piTablespaceList` y para definir un conjunto transportable válido.

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 cronológicas actualmente definida. Si los números generadores de base de datos no coinciden, en su lugar se utilizará la vía de acceso de anotaciones cronológicas de las imágenes de copia de seguridad, GLFH. Si los números generadores de base de datos coinciden, se utilizará la vía de acceso de anotaciones cronológicas del GLFH que actualmente está en el disco.

- `DB2RESTORE_LOGTARGET_DEFAULT`

Restaurar los archivos de anotaciones cronológicas a partir de la imagen de copia de seguridad en el directorio de anotaciones cronológicas por omisión de la base de datos, por ejemplo: `/home/dbuser/db2inst/NODE0000/SQL00001/LOGSTREAM0000`

piVendorOptions

Entrada. Se utiliza para pasar información desde la aplicación a las

db2Restore - Restaurar una base de datos o un espacio de tablas

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 emisor, por ejemplo, si se sabe que se han montado todos los soportes de almacenamiento necesarios para la restauración y no es necesario que aparezcan mensajes de solicitud de los 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

db2Restore - Restaurar una base de datos o un espacio de tablas

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, tras una restauración satisfactoria, la base de datos se encuentra en estado de avance pendiente, debe llamarse a la API db2Rollforward para poder utilizar la base de datos.
- 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.

db2Restore - Restaurar una base de datos o un espacio de tablas

- DB2RESTORE_TRANSPORT - Transportar un conjunto de espacios de tablas y esquemas de SQL. Los espacios de tablas y los esquemas que se indican en los parámetros **piTablespaceList** y **piSchemalist** deben definir un conjunto transportable válido (a partir de DB2 Versión 9.7 Fixpack 2).

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 sobregaba 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 sobregaba 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 sobregaba 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 sobregaba 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 sobregaba 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 enumeradas anteriormente.

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.

db2Restore - Restaurar una base de datos o un espacio de tablas

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 restauraciones no de instantánea:

- DB2RESTORE_LOGTARGET_DEFAULT
Restaurar los archivos de anotaciones cronológicas a partir de la imagen de copia de seguridad en el directorio de anotaciones cronológicas por omisión de la base de datos, por ejemplo: /home/dbuser/db2inst/NODE0000/SQL00001/LOGSTREAM0000

Para la restauración instantánea, se debe proporcionar 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

db2Restore - Restaurar una base de datos o un espacio de tablas

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. Configure este valor en `NULL` si no se han creado grupos de almacenamiento.

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 conjunto de bits `DB2RESTORE_GENERATE_SCRIPT`.

piStogroup

Entrada. Nombre del grupo de almacenamiento de destino de todos los espacios de tablas que se transportan. Este grupo de almacenamiento debe existir ya en la base de datos de destino. Solo tiene validez durante una operación de transporte de esquema.

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`.

db2Restore - Restaurar una base de datos o un espacio de tablas

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 selectiva.

No puede utilizar `SQLU_SNAPSHOT_MEDIA` con ninguna de las opciones siguientes:

- Acciones de 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 NULL 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 tablas.

El comportamiento por omisión al restaurar datos de una imagen de copia de seguridad selectiva 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

db2Restore - Restaurar una base de datos o un espacio de tablas

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 duplicación 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 selectivas 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 selectiva.

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**.

iStagingDBAliasLen

Entrada. Número entero sin signo, de 4 bytes, que representa la longitud en bytes del parámetro **piStagingDBAlias**.

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**.

db2Restore - Restaurar una base de datos o un espacio de tablas

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, que ya esté 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 - Restaurar una base de datos o un espacio de tablas

(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 que no hay más, vuelve a la API con el valor de acción de llamador DB2RESTORE_DEVICE_TERMINATE para indicar el fin del dispositivo de medios. 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 existe un archivo de configuración actual utilizable con cualquiera de estos parámetros habilitados, 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 cronológicas y de salida de usuario antes de conectarse con 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, la base de datos puede establecerse en el estado en el que se encontraba antes de que se produjera el daño emitiendo db2Rollforward tras la correcta ejecución de db2Restore. Si la base de datos puede recuperarse, tomará por omisión el estado de avance pendiente 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.

db2Restore - Restaurar una base de datos o un espacio de tablas

- 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 tablas. 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** cuando se emite el mandato **RESTORE DATABASE**, 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 de restauración 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 y el mismo sistema operativo 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, las vías de acceso de almacenamiento asociadas a esta base de datos pueden volver a definirse o permanecer como estaban. Para

db2Restore - Restaurar una base de datos o un espacio de tablas

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. Sin embargo, si se creó la base de datos con la cláusula `AUTOMATIC STORAGE NO`, las vías de acceso de almacenamiento asociadas con la base de datos no pueden volver a definirse.

Restauración instantánea

Igual que una restauración tradicional (no instantánea), el comportamiento por omisión al restaurar una imagen de copia de seguridad selectiva 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 de 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 anotaciones cronológicas reside bajo el directorio de bases de datos (por ejemplo, `/NODExxxx/SQLxxxx/LOGSTREAMxxxx/`). 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 base de datos particionada, se debe llamar a esta API desde la partición de catálogo. Las particiones se avanzan según lo que especifique en la cláusula **T0**:

- 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 de tablas, no se tienen que avanzar todas las tablas de ese espacio de tablas.

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYCTRL
- SYMAINT

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;
}

```

db2Rollforward - Avanzar una base de datos

```
    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 mediante la utilización de los archivos de anotaciones cronológicas disponibles y, a continuación, retrotrayendo ésta. 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 en el tiempo que el parámetro **piStopTime** especifica 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 - Avanzar una base de datos

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 cualquiera de las acciones de llamador de recuperación de carga (`DB2ROLLFORWARD_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 base de datos particionada, 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 la base 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 las tablas particionadas, el avance de punto en el tiempo (PIT) de un espacio de tablas que contenga cualquier parte de una tabla particionada también debe avanzar todos los demás espacios de tablas en los que resida esa tabla hasta el mismo punto en el 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 PIT también debe incluir 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 grabará 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.

db2Rollforward - Avanzar una base de datos

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** se debe establecer 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** se debe establecer 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 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.

piDroppedTblID

Entrada. Serie que contiene el ID de la tabla descartada cuya recuperación se está intentando. Para las tablas particionadas, el ID de tabla descartada (`drop-table-id`) identifica la tabla en su conjunto para 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.

db2Rollforward - Avanzar una base de datos

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 que se realiza cuando se llama a esta API depende del distintivo `rollforward_pending` que tiene 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 se encuentra en estado de avance pendiente. 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 avance pendiente cuando se llama a esta API, la base de datos se avanzará. Los 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 se establezcan en estado 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 avance pendiente 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 avance pendiente y no se especifica ningún punto en el tiempo, los espacios de tablas que se encuentren en estado de avance en proceso se avanzarán hasta el final de las anotaciones cronológicas. 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.

db2Rollforward - Avanzar una base de datos

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 sucesos:

- 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 recuperarse. 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 sólo se aplica a la recuperación en avance de bases de datos completa y al avance de espacios de tablas hasta un punto en el 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.

Cuando el distintivo `rollforward_recovery` se ha establecido en `DATABASE`, la base de datos no estará disponible para utilizarse hasta que termina la recuperación en avance. La finalización se realiza llamando a la API con la acción de emisor `DB2ROLLFORWARD_STOP` o `DB2ROLLFORWARD_RFWRD_STOP` para que la base de datos abandone el estado de avance pendiente. 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 los estados `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 base de datos particionada, 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. 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 o cualquier índice o vista de estadísticas asociados. Este mandato también puede actualizar estadísticas sobre las características de una tabla y de 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 de 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 en tablas, una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- DBADM
- SQLADM
- Privilegio CONTROL sobre la tabla
- LOAD

db2Runstats - Actualizar estadísticas para tablas e índices

Cuando se utiliza en vistas de estadísticas, una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- DBADM
- SQLADM
- 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:

- DATAACCESS
- 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;
    double iIndexSamplingOption;
} db2RunstatsData;

typedef SQL_STRUCTURE db2ColumnData
{
    unsigned char *piColumnName;
    db2int16 iColumnFlags;
} db2ColumnData;

typedef SQL_STRUCTURE db2ColumnDistData
{
```


db2Runstats - Actualizar estadísticas para tablas e índices

```
    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.

datos 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 término medio se recopilarían datos en una 1 fila de cada 10.000. DB2 tratará un valor de 0 o 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 de estadísticas 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 es necesario recopilar estadísticas de distribución para una columna o columnas en particular. 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

db2Runstats - Actualizar estadísticas para tablas e índices

piColumnList. Esto produce el mismo efecto que si esas columnas se hubieran incluido en **piColumnList** en primer lugar. 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.

Los campos de enteros segundo y tercero representan el número de valores frecuentes y el número de cuantiles al recopilar estadísticas de distribución para grupos de columnas. Esto no recibe soporte 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** en primer lugar. Si **iNumColGroups** es cero, se omite **piColumnGroupList**.

piIndexList

Entrada. Matriz de series. Cada serie contiene un nombre de índice totalmente calificado. Si **NumIndexes** es cero, se omite **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 include) son:

DB2RUNSTATS_ALL_COLUMNS

Recopilar estadísticas para todas las columnas de la tabla o vista de estadísticas. 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 de 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

db2Runstats - Actualizar estadísticas para tablas e índices

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 de estadísticas. 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 de 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 de 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 de 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 grabación.

DB2RUNSTATS_SAMPLING_SYSTEM

Recopilar estadísticas para un porcentaje de las páginas de datos según lo especificado por el usuario mediante el parámetro **iSamplingOption**. El muestreo de SYSTEM considera cada página individualmente, incluyendo la página con la probabilidad $P/100$ (donde P es el valor de **iSamplingOption**) y excluyendo la página con la probabilidad $1-P/100$. De esta manera, si **iSamplingOption** 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.

En las vistas de estadísticas, el muestreo de SYSTEM está restringido a las vistas cuyas definiciones son una selección en una sola tabla base. Si la vista contiene varias tablas, el muestreo de SYSTEM también es posible si:

db2Runstats - Actualizar estadísticas para tablas e índices

- Las tablas se unen utilizando predicados de igualdad en todas las columnas de clave primaria y clave foránea incluidas en una restricción de integridad referencial definida entre las tablas.
- Ninguna condición de búsqueda filtra las filas en las tablas padre de la relación; y
- Puede identificarse entre todas las tablas una única tabla hija que a su vez no sea una tabla padre.

Si la vista de estadísticas no cumple esas condiciones, en su lugar se utilizará el muestreo BERNOULLI y se devolverá un mensaje de aviso (SQL2317W).

Si no se especifica DB2RUNSTATS_SAMPLING_SYSTEM, DB2 presupone que se va a utilizar el muestreo BERNOULLI como método de muestreo. El muestreo de BERNOULLI considera cada página individualmente, incluyendo la fila con la probabilidad $P/100$ (donde P es el valor de **iSamplingOption**) y excluyendo la página 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 de estadísticas.

DB2RUNSTATS_SAMPLING_REPEAT

Especifica que se ha pasado un generador a través del parámetro **iSamplingRepeatable**. El valor **iSamplingRepeatable** se utilizará como generador para generar la muestra de datos. El parámetro **iSamplingOption** debe también especificarse para indicar la tasa de muestreo.

DB2RUNSTATS_USE_PROFILE

Recopilar estadísticas para una tabla o vista de estadísticas 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 - Actualizar estadísticas para tablas e índices

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.

DB2RUNSTATS_INDEX_SYSTEM

Recopilar estadísticas para un porcentaje de las páginas de índice según lo especificado por el usuario mediante el parámetro **iIndexSamplingOption**. El muestreo de SYSTEM considera cada página individualmente, incluyendo las páginas de índice con la probabilidad $P/100$ (donde P es el valor de **iIndexSamplingOption**) y excluyendo la página con la probabilidad $1-P/100$. De esta manera, si **iIndexSamplingOption** 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. Si no se especifica DB2RUNSTATS_INDEX_SYSTEM, DB2 presupone que se va a utilizar el muestreo BERNOULLI como método de muestreo. El muestreo de BERNOULLI considera cada página individualmente, incluyendo la fila con la probabilidad $P/100$ (donde P es el valor de **iIndexSamplingOption**) y excluyendo la página con la probabilidad $1-P/100$.

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 por omisión 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:

db2Runstats - Actualizar estadísticas para tablas e índices

- 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 frecuentes 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 de estadísticas.

iIndexSamplingOption

Entrada. Indica que deben recopilarse estadísticas sobre una muestra de datos de índice. **iIndexSamplingOption** 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 0 y 1. Por ejemplo, el valor 0,01 representa un porcentaje de una centésima, de forma que por término medio se tome de muestra una página de índice por cada 10.000. DB2 trata un valor de 0 o 100 como si no se especificara el muestreo, independientemente de si se ha especificado DB2RUNSTATS_INDEX_SYSTEM. DB2 trata un valor superior a 100 o inferior a 0 como 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_INDEX_SYSTEM en **iRunstatsFlags**. Si no se ha establecido el distintivo, se supone el muestreo BERNOULLI.

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 de base de datos **num_freqvalues**.

iNumQuantiles

Entrada. Número de cuantiles que se deben recopilar para la columna. Los valores válidos son:

- n** Recopilar *n* cuantiles en la columna.
- 1** Utilizar el número predefinido de tabla de cuantiles, **iTableDefaultQuantiles**, si está definido, o el parámetro de configuración de base de datos **num_quantiles**.

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 descartan 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.

En DB2 V9.7 Fixpack 1 y releases posteriores, a la recopilación de estadísticas de distribución para una columna de tipo XML se aplican los elementos siguientes:

- Se recopilan estadísticas de distribución para cada índice sobre los datos XML que se han especificado en una columna XML.
- Para recopilar estadísticas de distribución para un índice sobre datos XML, deben recopilarse estadísticas de distribución y estadísticas de tabla, ambas. Las

db2Runstats - Actualizar estadísticas para tablas e índices

estadísticas de tabla deben recopilarse con el fin de que puedan recopilarse las estadísticas de distribución, ya que las estadísticas de distribución XML se almacenan con las estadísticas de tabla.

No es necesario recopilar estadísticas de índice para poder recopilar estadísticas de distribución XML. La recopilación de estadísticas de índice sin recopilar estadísticas de distribución no recopila estadísticas de distribución XML.

Por omisión, las estadísticas de distribución XML utilizan un máximo de 250 cuantiles.

- Se recopilan estadísticas de distribución para los índices sobre los datos XML de tipo VARCHAR, DOUBLE, TIMESTAMP y DATE. No se recopilan estadísticas de distribución para los índices sobre los datos XML de tipo VARCHAR HASHED.
- No se recopilan estadísticas de distribución para los índices particionados sobre los datos XML que se han definido en una tabla particionada.

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 de 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.

No se recopilan estadísticas para columnas con tipos estructurados. Si se especifican, las columnas con estos tipos de datos se ignoran.

Sólo se recopilan AVGCOLLEN y NUMNULLS para las columnas con tipos de datos LOB o LONG.

AVGCOLLEN representa el espacio promedio en bytes si la columna se guarda en la memoria de base de datos o en una tabla temporal. Este valor representa la

db2Runstats - Actualizar estadísticas para tablas e índices

longitud del descriptor de datos para tipos de datos LOB o LONG, excepto cuando los datos LOB están en línea en la página de datos.

Nota: El espacio promedio necesario para almacenar la columna en el disco puede ser diferente del valor representado por esta estadística.

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.

db2SelectDB2Copy - Seleccionar la copia de DB2 que la aplicación utiliza

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 el nivel de release de la estructura que se ha pasado 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 grabaciones de E/S para la base de datos

Detiene o reanuda las operaciones de grabación de E/S para una base de datos. Las operaciones de grabación de E/S deben detenerse antes de que pueda tomarse una imagen duplicada dividida de una base de datos.

Ámbito

Esta API afecta únicamente a la partición de base de datos en la que se ejecuta la API. En entornos DB2 pureScale, esta API puede ejecutarse desde cualquier miembro para suspender operaciones de grabación de E/S para todos los miembros o para reanudar operaciones de grabación de E/S para todos los miembros suspendidos.

Autorización

Una de las autorizaciones 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 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:

db2SetWriteForDB - Suspender o reanudar las grabaciones de E/S para la base de datos

- DB2_DB_SUSPEND_WRITE

Detiene las operaciones de grabación de E/S en disco.

- DB2_DB_RESUME_WRITE

Reanuda las operaciones de grabación 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 **piIndoubtData**.

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 **piIndoubtData** establecido en NULL. A continuación, este valor se puede utilizar para asignar el almacenamiento intermedio y la API se puede emitir con **piIndoubtData** 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.

db2SpmListIndTrans - Listar transacciones dudosas SPM

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 retroacción 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 no sea suficientemente grande si hay más transacciones dudosas adicionales 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.

db2SpmListIndTrans - Listar transacciones dudosas SPM

* Para conectar con el Gestor de puntos síncronos, determine el nombre del Gestor de puntos síncronos 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 el nivel de release de la estructura que se ha pasado como segundo parámetro, **pParmStruct**.

pParmStruct

Entrada. Establecer 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 el nivel de release de la estructura que se ha pasado como segundo parámetro, **pParmStruct**.

pParmStruct

Entrada. Establecer 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 el nivel de release de la estructura que se ha pasado como segundo parámetro, **pParmStruct**.

pParmStruct

Entrada. Establecer 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.

Importante: Este mandato o API ha quedado en desuso y puede que se elimine en un futuro release, ya que el supervisor de salud ha quedado en desuso en la Versión 9.7. No recibe soporte en los entornos DB2 pureScale. Para obtener más información, consulte el tema “El Supervisor de salud ha quedado en desuso” en <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT

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;
```


db2UpdateAlertCfg - Actualizar los valores de configuración de alertas para los indicadores de salud

```
    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 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

db2UpdateAlertCfg - Actualizar los valores de configuración de alertas para los indicadores de salud

- 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**, se establece en DB2ALERTCFG_OBJTYPE_TABLESPACE o DB2ALERTCFG_OBJTYPE_TS_CONTAINER; de lo contrario, se establece en NULL.

piDbName

Entrada. 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 y DB2ALERTCFG_OBJTYPE_DATABASE; de lo contrario se establece en 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 matriz de estructura db2AlertAttrib.

iNumActionUpdates

Entrada. Número de acciones de alerta que se deben actualizar para el indicador de salud **iIndicatorID**.

piActionUpdates

Entrada. Puntero a la matriz de estructura db2AlertActionUpdate.

iNumActionDeletes

Entrada. Número de acciones de alerta que se debe suprimir del indicador de salud **iIndicatorID**.

piActionDeletes

Entrada. Puntero a la matriz de estructura db2AlertActionDelete.

iNumNewActions

Entrada. Número de nuevas acciones de alerta que se deben añadir al indicador de salud **iIndicatorID**.

piNewActions

Entrada. Puntero a la matriz de 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

db2UpdateAlertCfg - Actualizar los valores de configuración de alertas para los indicadores de salud

- 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 el estado, 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 el estado, 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 el estado, 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 sólo para DB2ALERTCFG_SCRIPTTYPE_OS.

stmtTermChar

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

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 autorizaciones 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 nivel de release de la estructura transferida como segundo parámetro **pParmStruct**.

db2UpdateAlternateServerForDB - Actualizar el servidor alternativo para un alias de base de datos en el directorio de bases de datos del sistema

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.

Esta API sólo se debe utilizar en un servidor. Si se emite en un cliente, se pasará por alto 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 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.

db2UpdateContact - Actualizar los atributos de un contacto

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 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.

db2UpdateContactGroup - Actualizar los atributos de un grupo de contactos

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** se establece en 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 salud emitidas por una instancia.

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT

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 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.

db2UpdateHealthNotificationList - Actualizar la lista de contactos a los que se puedan enviar notificaciones de alerta de salud

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** se establece en 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 autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT

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 que se ha pasado 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 include) 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 autorizaciones siguientes:

- Autorización DBADM
- Si se especifica `EXPLAIN ONLY`, basta con la autorización `EXPLAIN` o una autorización que incluya implícitamente `EXPLAIN`.
- Si no existe un paquete, la autorización `BINDADD`; y:
 - Si el nombre de esquema del paquete no existe, la autorización `IMPLICIT_SCHEMA` en la base de datos.
 - Si el nombre de esquema del paquete existe, el privilegio `CREATEIN` para el esquema.
- Si el paquete existe, uno de los privilegios siguientes:
 - Privilegio `ALTERIN` para el esquema
 - Privilegio `BIND` en el paquete

Además, si se captura información de Explain con la cláusula `EXPLAIN` o `EXPLSNAP`, es necesaria una de las autorizaciones siguientes:

- Privilegio `INSERT` para las tablas de Explain
- Autorización `DATAACCESS`

El usuario también necesita todos los privilegios necesarios para compilar las sentencias de SQL estático en la aplicación. Los privilegios otorgados a grupos no se utilizan para la comprobación de autorización de las sentencias estáticas.

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
```


sqlgndx - Programa de aplicación de vinculación para crear un paquete

```
sqlgndx (  
    unsigned short MsgFileNameLen,  
    unsigned short BindFileNameLen,  
    struct sqlca * pSqlca,  
    struct sqlopt * pBindOptions,  
    _SQLOLDCHAR * pMsgFileName,  
    _SQLOLDCHAR * pBindFileName);
```

Parámetros de la API sqlgndx

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 sobregrebará. 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 sqlgndx

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 sobregrebará. 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 `myapp.sqc` genera un archivo de vinculación por omisión llamado `myapp.bnd` y un nombre de paquete por omisión `MYAPP`. (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 prerequisites 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)).

sqlaintp - Obtener mensaje de error

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 línea para cada línea del mensaje de texto. Las líneas se dividen a nivel de palabras. Si no se especifica un valor para el ancho o se establece en 0, se devuelve el texto de 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 autorizaciones siguientes:

- Autorización DBADM
- Si se especifica EXPLAIN ONLY, basta con la autorización EXPLAIN o una autorización que incluya implícitamente EXPLAIN.
- Si se especifica SQLERROR CHECK o EXPLAIN ONLY, es suficiente una autorización EXPLAIN o SQLADM.
- Si no existe un paquete, la autorización BINDADD; y:
 - Si el nombre de esquema del paquete no existe, la autorización IMPLICIT_SCHEMA en la base de datos.
 - Si el nombre de esquema del paquete existe, el privilegio CREATEIN para el esquema.
- Si el paquete existe, uno de los privilegios siguientes:
 - Privilegio ALTERIN para el esquema
 - Privilegio BIND en el paquete

Además, si se captura información de Explain con la cláusula EXPLAIN o EXPLSNAP, es necesaria una de las autorizaciones siguientes:

- Privilegio INSERT para las tablas de Explain
- Autorización DATAACCESS

El usuario también necesita todos los privilegios necesarios para compilar las sentencias de SQL estático en la aplicación. Los privilegios otorgados a grupos no se utilizan para la comprobación de autorización de las sentencias estáticas.

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);
```

sqlaprep - Precompilar programa de aplicación

```
SQL_API_RC SQL_API_FN
sqlgprep (
    unsigned short MsgFileNameLen,
    unsigned short ProgramNameLen,
    struct sqlca * pSqlca,
    struct sqlopt * pPrepOptions,
    _SQLLDCHAR * pMsgFileName,
    _SQLLDCHAR * 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 sobregrebará. 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.

sqlprep - Precompilar programa de aplicación

A continuación de la conexión a una base de datos, sqlprep se ejecuta bajo la transacción que se ha iniciado. A continuación, PRECOMPILE emite una operación COMMIT o ROLLBACK para terminar 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 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 sqlprep, 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.

sqlarbind - 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 autorizaciones siguientes:

- Autorización DBADM
- Privilegio ALTERIN sobre 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
sqlarbind (
    char * pPackageName,
    struct sqlca * pSqlca,
    struct sqlopt * pRebindOptions);
```

```
SQL_API_RC SQL_API_FN
sqlgrbind (
    unsigned short PackageNameLen,
    char * pPackageName,
    struct sqlca * pSqlca,
    struct sqlopt * pRebindOptions);
```

Parámetros de la API sqlarbind

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 sqlgrbind

PackageNameLen

Entrada. Longitud, en bytes, del parámetro **pPackageName**.

Notas de uso

REBIND no confirma automáticamente la transacción tras 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 método 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 recientemente creado, puede utilizarse el mandato **REBIND** para volver a crear el paquete. **REBIND** también se puede utilizar para volver a crear paquetes después de haber ejecutado el mandato **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, tras la actualización de la base de datos, todos los paquetes almacenados en la base de datos se invalidarán con el mandato **UPGRADE 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** or **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 da soporte a ninguna opción de

sqlarbnd - Volver a vincular paquete

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.

REBIND es soportado por DB2 Connect.

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 vuelve a crear el paquete desde las sentencias de SQL almacenadas en la tabla de catálogo 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 **VERSION** coincida con la **VERSION** 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 (**REBIND**) 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 Explain utilizadas son las del solicitante de **REBIND**, no del vinculator 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 autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON
- 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.

Importante: Este mandato o API ha quedado en desuso en la Versión 9.7 y puede que se elimine en un futuro release. En su lugar, pueden utilizarse las funciones de tabla `MON_GET_TABLESPACE` y `MON_GET_CONTAINER`, que devuelven más información. Para obtener más información, consulte el tema “Los mandatos `LIST TABLESPACES` y `LIST TABLESPACE CONTAINERS` han quedado en desuso” en <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055001.html>.

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON
- 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 base de datos particionada, solamente se listan los espacios de tablas pertenecientes a la partición de base de datos actual.

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON
- 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 estas estructuras indicado por **MaxContainers**, y establecer

sqlbftcq - Captar los datos de la consulta para filas de un contenedor de espacio de tablas

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.

Importante: Este mandato o API ha quedado en desuso en la Versión 9.7 y puede que se elimine en un futuro release. En su lugar, pueden utilizarse las funciones de tabla MON_GET_TABLESPACE y MON_GET_CONTAINER, que devuelven más información. Para obtener más información, consulte el tema “Los mandatos LIST TABLESPACES y LIST TABLESPACE CONTAINERS han quedado en desuso” en <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055001.html>.

Ámbito

En un entorno de base de datos particionada, solamente se listan los espacios de tablas pertenecientes a la partición de base de datos actual.

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- sysmaint
- SYSMON
- 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.

sqlbftpq - Captar los datos de la consulta para filas de un espacio de tablas

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 de espacio de un espacio de tablas.

Importante: Este mandato o API ha quedado en desuso en la Versión 9.7 y puede que se elimine en un futuro release. En su lugar, pueden utilizarse las funciones de tabla `MON_GET_TABLESPACE` y `MON_GET_CONTAINER`, que devuelven más información. Para obtener más información, consulte el tema “Los mandatos `LIST TABLESPACES` y `LIST TABLESPACE CONTAINERS` han quedado en desuso” en <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055001.html>.

Ámbito

En un entorno de base de datos particionada, solamente se listan los espacios de tablas pertenecientes a la partición de base de datos actual.

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON
- 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.

sqlbgtss - Obtener estadísticas de utilización del espacio de tablas

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.

Importante: Este mandato o API ha quedado en desuso en la Versión 9.7 y puede que se elimine en un futuro release. En su lugar, pueden utilizarse las funciones de tabla `MON_GET_TABLESPACE` y `MON_GET_CONTAINER`, que devuelven más información. Para obtener más información, consulte el tema “Los mandatos `LIST TABLESPACES` y `LIST TABLESPACE CONTAINERS` han quedado en desuso” en <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055001.html>.

Ámbito

En un entorno de base de datos particionada, solamente se listan los espacios de tablas pertenecientes a la partición de base de datos actual.

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON
- 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);
```

sqlbmtsq - Obtener los datos de la consulta para todos los espacios de tablas

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 un puntero a ese espacio al que realiza la llamada. 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 SQLB_RESERVED1.

reserved2

Entrada. Siempre 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 sqlcfmem.

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 autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON
- 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 tablas:

- sqlbctcq

sqlbotcq - Abrir una consulta de contenedor de espacio de tablas

Busca una lista completa de la información del contenedor. La API asigna el espacio necesario para contener la información de todos los contenedores y devuelve un puntero a esta información. Utilice esta API para explorar la lista de contenedores cuando busque información específica. El uso de esta API es equivalente a llamar a las tres API `sqlbotcq`, `sqlbftcq` y `sqlbctcq`, con la diferencia de que esta API asigna automáticamente la memoria para la información de salida. Una llamada a esta API debe ir seguida de una llamada a `sqlfmem` para liberar memoria.

- `sqlbotcq`
- `sqlbftcq`
- `sqlbctcq`

Estas tres API funcionan como un cursor SQL en tanto que utilizan el paradigma OPEN/FETCH/CLOSE. El llamante debe proporcionar el área de salida para la búsqueda. A diferencia del cursor SQL, sólo puede haber una consulta de contenedor de espacio de tablas activa a la vez. Utilice este conjunto de API para explorar la lista de contenedores de espacios de tablas cuando busque información específica. Estas API permiten al usuario controlar los requisitos de memoria de una aplicación (comparado con `sqlbctcq`).

Cuando se llama a `sqlbotcq`, se genera una instantánea de la información del contenedor de espacio de tablas actual en el agente que presta servicio a la aplicación. Si la aplicación emite una segunda llamada de consulta del contenedor de espacio de tablas (`sqlbctcq` 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 generar 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.

Importante: Este mandato o API ha quedado en desuso en la Versión 9.7 y puede que se elimine en un futuro release. En su lugar, pueden utilizarse las funciones de tabla `MON_GET_TABLESPACE` y `MON_GET_CONTAINER`, que devuelven más información. Para obtener más información, consulte el tema “Los mandatos `LIST TABLESPACES` y `LIST TABLESPACE CONTAINERS` han quedado en desuso” en <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055001.html>.

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMAINT
- SYSMON
- 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.

sqlbotsq - Abrir una consulta de espacio de tablas

pNumTablespaces

Salida. Número de espacios de tablas contenidos en la base de datos conectada.

Notas de uso

Esta API va normalmente seguida de una o más llamadas a sqlbftpq y luego de 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
Busca información sobre todos los espacios de tablas. La API asigna el espacio necesario para contener la información de todos los espacios de tablas y devuelve un puntero a esta información. Utilice esta API para explorar la lista de espacios de tablas cuando busque información específica. El uso de esta API es equivalente a llamar a las tres API que se indican en la lista siguiente, excepto que esta API asigna automáticamente la memoria para la información de salida. Una llamada a esta API debe ir seguida de una llamada a sqlfmem para liberar memoria.
- sqlbotsq
- sqlbftpq
- sqlbctsq
Estas tres API funcionan como un cursor SQL en tanto que utilizan el paradigma OPEN/FETCH/CLOSE. El llamante debe proporcionar el área de salida para la búsqueda. A diferencia del cursor SQL, solo puede haber una consulta de espacio de tablas activa a la vez. Utilice este conjunto de API para explorar la lista de espacios de tablas cuando busque información específica. Este conjunto de API permite al usuario controlar los requisitos de memoria de una aplicación (comparado con sqlbmtsq).

Cuando se llama a sqlbotsq, se guarda una instantánea de la información del espacio de tablas actual en el almacenamiento intermedio del 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 por la 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.

Importante: Este mandato o API ha quedado en desuso en la Versión 9.7 y puede que se elimine en un futuro release. En su lugar, pueden utilizarse las funciones de tabla `MON_GET_TABLESPACE` y `MON_GET_CONTAINER`, que devuelven más información. Para obtener más información, consulte el tema “Los mandatos `LIST TABLESPACES` y `LIST TABLESPACE CONTAINERS` han quedado en desuso” en <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055001.html>.

Ámbito

En un entorno de base de datos particionada, solamente se listan los espacios de tablas pertenecientes a la partición de base de datos actual.

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON
- 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.

sqlbstpq - Obtener información sobre un espacio de tablas individual

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).

reservada

Entrada. Siempre 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 combinación de las API OPEN TABLESPACE QUERY, FETCH y CLOSE que requieren más recursos, que deben utilizarse para explorar el espacio de tablas necesario 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

Proporciona un mecanismo de restauración redirigida, en el que el usuario restaura una base de datos y, si es necesario, un conjunto diferente de contenedores de almacenamiento del sistema operativo.

Utilice esta API cuando el espacio de tablas se encuentre en un estado definición de almacenamiento pendiente o en un estado de definición de almacenamiento permitido. Estos estados son posibles durante una operación de restauración, inmediatamente con anterioridad a la restauración de las páginas de base de datos.

Autorización

Una de las autorizaciones 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.

sqlbstsc - Definir contenedores de espacios de tablas

NumContainers

Entrada. Número de filas contenidas en la estructura apuntada por **pContainerData**. Si se proporciona un valor de 0 con un puntero NULL para **pContainerData** quiere decir que la gestión del espacio de tablas se realizará mediante almacenamiento automático.

pContainerData

Entrada. Especificaciones del contenedor. Aunque se utilice la estructura `SQLB_TBSCONTQRY_DATA`, solo se utilizan los campos **contType**, **totalPages**, **name** y **nameLen** (para lenguajes que no sean C); los demás campos no se tienen en cuenta. Un valor NULL con un valor 0 para **NumContainers** indica que la gestión del espacio de tablas se realizará mediante almacenamiento automático. Esta opción también se puede utilizar para ofrecer a los espacios de tablas habilitados para el almacenamiento automático existentes una distribución en bandas mejorada en las vías de acceso de almacenamiento existentes, mediante la redefinición de los contenedores.

Nota: el espacio de tablas permanecerá fuera de línea mientras se esté ejecutando la restauración.

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 tablas, 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 tablas. 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 establecer 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`.

Esta secuencia de acciones 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.

El uso de esta API para habilitar el almacenamiento automático para espacios de tablas obligará a que se vuelvan a definir todos los contenedores actuales de forma que utilicen las vías de acceso de almacenamiento proporcionadas para la base de datos.

Los espacios de tablas gestionados por el sistema (SMS) existentes no se pueden convertir al uso del almacenamiento automático.

Se omita `SetContainerOptions` cuando se está convirtiendo un espacio de tablas al uso del almacenamiento automático (`NumContainers` es 0 y `pContainerData` es NULL).

Si se realiza una restauración redirigida de un espacio de tablas en un entorno de múltiples particiones, utilizando para ello la opción **USING AUTOMATIC STORAGE** del mandato **SET TABLESPACE CONTAINERS**, sólo se convertirá al almacenamiento automático el espacio de tablas de la partición que se está restaurando. No se redefinen los contenedores de las otras particiones de base de datos.

Nota: La conversión a almacenamiento automático del espacio de tablas correspondiente a solamente una de las particiones como parte de una operación de restauración redirigida provoca incoherencias en la definición del espacio de tablas. También se podrían generar errores inesperados al añadir particiones de base de datos nuevas al sistema o al grupo de particiones de base de datos. Por ejemplo, si todas las particiones de base de datos se sometieran a una restauración redirigida y, a continuación, se utilizara la opción **USING AUTOMATIC STORAGE** del mandato **SET TABLESPACE CONTAINERS**, el espacio de tablas se convertiría al almacenamiento automático en todas las particiones de base de datos. Si se añadiera otra partición de base de datos posteriormente, ésta tendría la misma definición de espacio de tablas que las otras particiones de base de datos.

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.

Importante: Este mandato o API ha quedado en desuso en la Versión 9.7 y puede que se elimine en un futuro release. En su lugar, pueden utilizarse las funciones de tabla `MON_GET_TABLESPACE` y `MON_GET_CONTAINER`, que devuelven más información. Para obtener más información, consulte el tema “Los mandatos `LIST TABLESPACES` y `LIST TABLESPACE CONTAINERS` han quedado en desuso” en <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055001.html>.

Ámbito

En un entorno de base de datos particionada, solamente se listan los espacios de tablas pertenecientes a la partición de base de datos actual.

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON
- 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 tablas.

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 `sqlfmem`. 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 la 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 de gestor de bases de datos **nombre_spm** como *aliasbd* 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 asociados, 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 miembros. Si uno o varios de estos miembros encuentran un error durante la activación de la base de datos, se devuelve un aviso. La base de datos permanece activada en todos los miembros en los que la API ha sido satisfactoria.

Nota: Si se ha encontrado un error, la base de datos puede permanecer desactivada en esos miembros donde la API ha fallado.

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT

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
db2ActivateDb (
    db2UInt32 versionNumber,
    void * pDB2ActivateDbStruct,
    struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ActivateDbStruct
{
    char * piDbAlias;
    char * piUserName;
    char * piPassword;
    db2UInt32 iOptions;
} db2ActivateDbStruct;
```

Parámetros de la API db2ActivateDb

versionNumber

Entrada. Especifica la versión y el nivel de release de la estructura transferida como los datos del segundo parámetro.

pDB2ActivateDbStruct

Entrada. Puntero a la estructura db2ActivateDbStruct.

pSqlca Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2ActivateDbStruct

piDbAlias

Entrada. Puntero al nombre de alias de base de datos.

piUserName

Entrada. Puntero al ID de usuario que empieza la base de datos. Puede ser NULL.

piPassword

Entrada. Puntero a la contraseña para el nombre de usuario. Puede ser NULL, pero debe especificarse si se especifica un nombre de usuario.

iOptions

Entrada. Reservado para una utilización futura.

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.

Ejemplos

En este apartado se proporciona un ejemplo de un caso de activación de base de datos. El bloque de código siguiente es un programa de cuerpo común que se utilizará para casos de activación de bases de datos.

```
struct sqlca sqlca;           // sqlca to carry the sqlcode
struct db2ActivateDbStruct activateDbStruct;
struct db2ActDeactMemberStruct memberList;

strcpy(activateDbStruct->piUserName, "USER1"); //local instance
```

La siguiente API de activación se muestra como habitual en los casos que se enumeran en los ejemplos siguientes:

```
db2ActivateDb( db2Version, &activateDbStruct, &sqlca );
```

- Active una base de datos globalmente en todos los miembros

```
activateDbStruct.iOptions = DB2_ACTDEACT_DB_GLOBAL;  
activateDbStruct->piMemberList = NULL;
```

- Active una base de datos en un miembro individual 20

```
activateDbStruct.iOptions = DB2_ACTDEACT_DB_MEMBER;  
activateDbStruct->piMemberList->numMembers = 1;  
activateDbStruct->piMemberList->pMember[0] = 20;
```

Notas de uso

Los administradores de bases de datos pueden utilizar **ACTIVATE DATABASE** para iniciar 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** solo se pueden concluir mediante `sqlc_deactivate_db` o mediante `db2InstanceStop`. Para obtener una lista de las bases de datos activadas, puede llamar `db2GetSnapshot`. API

Si una base de datos se ha iniciado mediante un mandato **DB2 CONNECT TO** (o mediante una conexión implícita) y posteriormente se emite **ACTIVATE DATABASE** para esa misma base de datos, también debe utilizarse **DEACTIVATE DATABASE** para concluir esa base de datos.

Consulte el parámetro de configuración **autorestart** para conocer el comportamiento de conexión de la base de datos cuando la base de datos de destino tenga un estado incoherente.

Sintaxis de la API de REXX

Esta API se puede invocar desde REXX mediante la interfaz SQLDB2.

sqle_deactivate_db - Desactivar base de datos

Desactiva la base de datos especificada y detiene todos los servicios de base de datos asociados.

Ámbito

Esta API desactiva la base de datos especificada en todos los miembros. Si uno o varios de estos miembros encuentran un error durante la desactivación de la base de datos, se devuelve un aviso. La base de datos se desactivará en todos los miembros donde se haya ejecutado la API satisfactoriamente.

Nota: Si se ha encontrado un error, la base de datos puede permanecer activada en esos miembros donde la API ha fallado.

Autorización

Una de las autorizaciones 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
    db2DeactivateDb(
        db2UInt32 versionNumber,
        void * pDB2DeactivateDbStruct,
        struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2DeactivateDbStruct
{
    char * piDbAlias;
    char * piUserName;
    char * piPassword;
    db2UInt32 iOptions;
    db2UInt32 iNumMembers;
    SQL_PDB_NODE_TYPE * piMemberList;
} db2DeactivateDbStruct;
```

Parámetros de la API db2DeactivateDb

versionNumber

Entrada. Especifica la versión y el nivel de release de la estructura transferida como los datos del segundo parámetro.

pDB2DeactivateDbStruct

Entrada. Puntero a la estructura db2DeactivateDbStruct.

pSqlca Salida. Puntero a la estructura sqlca.

Parámetros de la estructura de datos db2DeactivateDbStruct

piDbAlias

Entrada. Puntero al nombre de alias de base de datos.

piUserName

Entrada. Puntero al ID de usuario que empieza la base de datos. Puede ser NULL.

piPassword

Entrada. Puntero a la contraseña para el nombre de usuario. Puede ser NULL, pero debe especificarse si se especifica un nombre de usuario.

iOptions

Entrada. Valor predefinido que se utiliza explícitamente para forzar la desactivación de la base de datos y para denotar que el mandato de desactivación se está emitiendo en el contenido de una lista de miembros especificada.

iNumMembers

Entrada. Número de miembros enumerados en **iMemberList**.

piMemberList

Entrada. Puntero a la lista de miembros.

Parámetros de la API sqlc_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.

Ejemplos

En este apartado se proporcionan ejemplos de distintos casos de desactivación de bases de datos. El bloque de código siguiente es un programa de cuerpo común que se utilizará para casos de desactivación de bases de datos.

```
struct sqlca sqlca; // sqlca to carry the sqlcode
struct db2DeactivateDbStruct deactivateDbStruct;

strcpy(deactivateDbStruct->piUserName, "USER1"); //local instance
```

La siguiente API de desactivación se muestra como habitual en los casos que se enumeran en los ejemplos siguientes:

```
db2DeactivateDb( db2Version, &deactivateDbStruct, &sqlca );
```

- Desactive una base de datos globalmente en todos los miembros

```
deactivateDbStruct.iOptions = DB2_ACTDEACT_DB_GLOBAL;
deactivateDbStruct->piMemberList = NULL;
```

- Desactive una base de datos en un miembro

```
deactivateDbStruct.iOptions = DB2_ACTDEACT_DB_MEMBER;
deactivateDbStruct->piMemberList->numMembers = 1;
deactivateDbStruct->piMemberList->pMember[0] = 20;
```

- Desactive una base de datos en un miembro específico 30, mediante la opción FORCE

```
deactivateDbStruct.iOptions = DB2_ACTDEACT_DB_MEMBER;
deactivateDbStruct.iOptions |= DB2_DEACTIVATE_FORCE;
deactivateDbStruct->piMemberList->numMembers = 1;
deactivateDbStruct->piMemberList->pMember[0] = 30;
```

Notas de uso

Las bases de datos inicializadas por **ACTIVATE DATABASE** solo 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 aplicación de usuario que se desconecte no concluirá la base de datos; también se deberá utilizar **DEACTIVATE DATABASE** en ese caso.

Sintaxis de la API de REXX

Esta API se puede invocar desde REXX mediante la interfaz SQLDB2.

sqlleadn - Añadir una partición de base de datos al entorno de base de datos particionado

Añade una partición de base de datos a un servidor de particiones de bases 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 autorizaciones 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
sqlleadn (
    void * pAddNodeOptions,
    struct sqlca * pSqlca);
```

```
SQL_API_RC SQL_API_FN
sqlgaddn (
    unsigned short addnOptionsLen,
    struct sqlca * pSqlca,
    void * pAddNodeOptions);
```

Parámetros de la API sqlleadn

pAddNodeOptions

Entrada. Puntero a la estructura opcional `sqlc_addn_options`. Esta estructura se utiliza para especificar el servidor de particiones de base de datos fuente, si existe, de las definiciones del espacio de tablas temporal del sistema para todas las particiones de base de datos que se van a crear. 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 `sqlc_addn_options` en bytes.

Notas de uso

Esta API solo debe utilizarse si se añade un servidor de particiones de base de datos a un entorno que tenga una base de datos y ésta no se ha catalogado en el momento en que se realizó la operación de añadir la partición. En esta situación, puesto que la base de datos no está catalogada, la operación de añadir la partición no reconoce la base de datos y no crea una partición de base de datos correspondiente a la base de datos del nuevo servidor de particiones de base de datos. Cualquier intento de conectarse a la partición de base de datos en el nuevo servidor de particiones de base de datos dará como resultado un error. En primer lugar, la base de datos debe catalogarse para que pueda utilizarse la API sqleaddn para crear la partición de base de datos correspondiente a la base de datos del nuevo servidor de particiones de base de datos.

Esta API no deberá utilizarse si el entorno tiene más de una base de datos y al menos una de esas bases de datos se catalogó en el momento de realizar la operación de añadir la partición. En esta situación, utilice la API sqlecran para crear una partición de base de datos para cada base de datos que no se catalogó en el momento de realizar la operación de añadir partición. En primer lugar, cada base de datos no catalogada deberá catalogarse para que pueda utilizarse la API sqlecran para crear la partición de base de datos correspondiente a la base de datos del nuevo servidor de particiones de base de datos.

Antes de añadir una nueva partición de base de datos, asegúrese de que hay suficiente almacenamiento para los contenedores que deben crearse.

La operación de añadir nodo crea una partición de base de datos vacía en el nuevo servidor de partición de base de datos para cada base de datos que exista en la instancia. Los parámetros de configuración de las nuevas particiones de base de datos se establecen con el valor por omisión.

Nota: Las bases de datos no catalogadas no se reconocen cuando se añade una nueva partición de base de datos. Las bases de datos no catalogadas no estarán presentes en la nueva partición de base de datos. Cualquier intento de conectarse a la base de datos de la nueva partición de base de datos devolverá el mensaje de error SQL1013N.

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 fallará si hay una operación de crear base de datos o descartar base de datos en curso. Se puede volver a llamar a la API cuando la operación haya finalizado.

sqleaddn - Añadir una partición de base de datos al entorno de base de datos particionado

Las definiciones de vía de acceso de almacenamiento de los grupos de almacenamiento se recuperan cuando la API sqleaddn tiene que comunicarse con la partición de catálogo de cada una de las bases de datos en la instancia. 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 base de datos particionada a fin de recuperar las definiciones de espacios de tablas. Se utiliza el parámetro de configuración **start_stop_time** del gestor de bases de datos para especificar la hora, en minutos, a la que el otro servidor de particiones de base de datos debe responder con las definiciones de espacio de tablas y almacenamiento automático. Si se excede este tiempo, la API fallará. Aumente el valor de **start_stop_time**, y vuelva a invocar a la API.

Sintaxis de la API de REXX

Esta API se puede invocar desde REXX mediante la interfaz SQLDB2.

sqlcatcp - 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 sqlcatin 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
sqlcatcp (
    char * pNodeName,
    char * pUserName,
    char * pPassword,
    char * pNewPassword,
    struct sqlca * pSqlca);
```

Parámetros de la API sqlcatcp

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 autenticar 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. En un entorno DB2 pureScale, tal como indica el establecimiento de SQLWARN0 en 'W' y de SQLWARN4 en 'S', este valor representa el número de miembro. En un entorno de base de datos particionada, tal como indica el símbolo 10, este valor representa el número de partición de la base de datos. Si la base de datos no está en un entorno particionado ni en un entorno DB2 pureScale, este valor siempre será 0, en caso de estar presente.
9. En un entorno DB2 pureScale, tal como indica el establecimiento de SQLWARN0 en 'W' y de SQLWARN4 en 'S', este valor representa el número de miembros en una instancia de DB2 pureScale. Si este valor es distinto de cero, representa el número de particiones de bases de datos en un entorno de base de datos particionada. Si la base de datos no está en un entorno particionado ni en un entorno DB2 pureScale, este valor siempre será 0. Este símbolo está presente solamente con la versión 5 o posterior.

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 ha descrito 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 autentifican el nombre de usuario y la contraseña, y el lugar donde se cambia la contraseña dependen del tipo de autenticación de la instancia seleccionada.

El nodo con el que se debe establecer una conexión también se puede especificar mediante una llamada a la API sqlsetc.

Sintaxis de la API de REXX

Desde REXX no se puede llamar directamente a esta API. No obstante, los programadores de REXX pueden utilizar esta función llamando al procesador de línea de mandatos de DB2 para ejecutar el mandato **ATTACH**.

sqlcatin - 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 `sqlcatcp` en lugar de la API `sqlcatin`.

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
sqlcatin (
    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 `sqlcatin`

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 autenticar la conexión. Puede ser NULL.

pPassword

Entrada. Serie que contiene la contraseña del nombre de usuario especificado. Puede ser NULL.

sqlcain - Conectar a instancia

pSqlca

Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlcain

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. Si la instancia de servidor funciona en un entorno DB2 pureScale, tal como indica el establecimiento de SQLWARN0 en 'W' y de SQLWARN4 en 'S', este valor representa al número de miembro. Si, tal como indica el símbolo 9, la instancia de servidor funciona en un entorno particionado, este símbolo representa el número de nodo. Si la instancia de servidor funciona en un entorno no particionado y fuera de un entorno DB2 pureScale, este valor no es aplicable y siempre es 0.
9. Si este valor es cero, la instancia de servidor funciona en un entorno no particionado y fuera de un entorno DB2 pureScale. De otro modo, este valor distinto de cero representa el número de miembros en una instancia de DB2 pureScale, si SQLWARN0 se ha establecido en 'W' y SQLWARN4 se ha establecido en 'S'. Si este valor no es cero, pero no se han establecido los valores de SQLWARN0 ni SQLWARN4, representa el número de nodos en un entorno particionado. Este símbolo está presente sólo con la Versión 5 o posterior.

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 ha descrito 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 autentifica el nombre de usuario y la contraseña depende del tipo de autenticación 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.

nombreusuario

Nombre utilizado por el usuario para conectar con la instancia.

contraseña

Contraseña utilizada para autenticar el nombre de usuario.

sqlcadb - 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 base de datos particionada, 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 autorizaciones 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
sqlcadb (
    _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 de la 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 **dfldbpath** 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 **dfldbpath**.

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 autenticación especificado para la base de datos. La autenticación es un proceso que verifica que el usuario sea quien dice ser. El acceso a objetos de base de datos depende de la autenticación del usuario. Los valores válidos (definidos en `sqlenv.h`) son:

SQL_AUTHENTICATION_SERVER

Especifica que la autenticación 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 autenticación se realiza en el servidor de particiones de base de datos donde se invoca la aplicación.

sqlcadb - Catalogar una base de datos del directorio de bases de datos del sistema

SQL_AUTHENTICATION_KERBEROS

Especifica que la autenticación se realiza utilizando el mecanismo de seguridad de Kerberos.

SQL_AUTHENTICATION_NOT_SPECIFIED

Autenticación no especificada.

SQL_AUTHENTICATION_SVR_ENCRYPT

Especifica que la autenticación se realiza en el servidor de particiones de base de datos donde reside la base de datos de destino, y que la contraseña de autenticación se debe cifrar.

SQL_AUTHENTICATION_DATAENC

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

SQL_AUTHENTICATION_SVRENC_AES0

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 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 autenticación 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 sqlcadb

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.

sqlecadb - Catalogar una base de datos del directorio de bases de datos del sistema

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 autenticación
- 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 **Type** establecido en `SQL_INDIRECT`, no se tendrá en cuenta el valor del parámetro **Authentication** proporcionado, y la autenticación se establecerá en `SQL_AUTHENTICATION_NOT_SPECIFIED` en el directorio.

sqlecadb - Catalogar una base de datos del directorio de bases de datos del sistema

Si el almacenamiento en antememoria de directorios está habilitado, 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 (sólo servidor) en DB2, detenga (**db2stop**) el gestor de bases de datos y, a continuación, vuelva a iniciar (**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 autenticación. Los valores válidos son:

SERVER La autenticación 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 autenticación se realiza en el servidor de particiones de base de datos donde se invoca la aplicación.

KERBEROS

Especifica que la autenticación se realiza utilizando el mecanismo de seguridad de Kerberos.

NOT_SPECIFIED

Autenticación no especificada.

SVR_ENCRYPT

Especifica que la autenticación 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 autenticación deben cifrarse.

DATAENC

Especifica que la autenticación tiene lugar en el servidor de

sqlcadb - Catalogar una base de datos del directorio de bases de datos del sistema

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 autenticación se realiza utilizando un mecanismo de seguridad externo basado en un plugin de la API de GSS.

SQL_AUTHENTICATION_SVRENC_AES0

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 permiten caracteres de retorno de carro ni 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 /.../cell11/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 ha de volver a crearse. 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 de este servidor de particiones de base de datos se encontrará en estado de restauración pendiente. 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 autorizaciones 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);
```

sqlecran - Crear una base de datos en un servidor de particiones de base de datos

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.

sqlcrea - 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 base de datos particionada, 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 autorizaciones 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
sqlcrea (
    char * pDbName,
    char * pLocalDbAlias,
    char * pPath,
    struct sqlbdbdesc * 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 sqlbdbdesc * pDbDescriptor,
    char * pPath,
    char * pLocalDbAlias,
    char * pDbName);
```

Parámetros de la API sqlcrea

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 base de datos no se establezca en una vía de acceso montada con NFS (por ejemplo, en sistemas Linux y UNIX, no deberá especificar el directorio \$HOME del propietario de la instancia). La vía de acceso especificada para esta API en un entorno de base de datos particionada 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 necesitan un conjunto de códigos y un territorio determinados para una base de datos, el conjunto de códigos y el territorio necesarios deben especificarse mediante la estructura `sqledbterritoryinfo`. Si este campo es NULL, se permite una de las opciones siguientes como valor de clasificación para la base de datos (sqlcode 1083): NULL, SQL_CS_SYSTEM, SQL_CS_IDENTITY_16BIT o SQL_CS_UNICODE.

Importante: Las clasificaciones basadas en el Algoritmo de clasificación Unicode del estándar Unicode versión 4.0.0 han quedado en desuso en la Versión 10.1 y podrían eliminarse en un release futuro. Para obtener más información, consulte el tema “Las clasificaciones basadas en el Algoritmo de clasificación Unicode del estándar Unicode versión 4.0.0 han quedado en desuso” en *Novedades en DB2 Versión 10.1*

sqlcrea - Crear una base de datos

Reserved2

Entrada. Reservado para una utilización futura.

pDbDescriptorExt

Entrada. Este parámetro hace referencia a un bloque de descripción de base de datos ampliado (sqllebdbdescext) 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 4.096 bytes para la base de datos y no se crea el grupo de almacenamiento por omisión, IBMSTOGROUP.

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 base de datos particionada, crea la base de datos en todos los servidores de particiones de base de datos listados en db2nodes.cfg y crea un directorio \$DB2INSTANCE/NODExxx en el subdirectorio especificado de cada servidor de partición de base de datos, donde xxx 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, sqlcrea 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 las autorizaciones siguientes:
 - 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 DATABASE** .

Con la autorización DBADM, se pueden otorgar estos privilegios (y revocarlos) a otros usuarios o a PUBLIC. Si otro administrador con autorización SYSADM o DBAD 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 base de datos particionada, el gestor de bases de datos crea un subdirectorio, \$DB2INSTANCE/NODExxxx, bajo la vía de acceso especificado o por

sqlcrea - Crear una base de datos

omisión en todos los servidores de particiones de base de datos. *xxxx* es el número de nodo 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 `sqlcrea` 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` necesaria 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.

sql819a

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.

sql819b

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.

sql850a

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.

sql850b

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.

sql932a

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.

sql932b

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 sqlcadb 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 nombrebd [ON víaAcceso] [ALIAS aliasbd]
[USING CODESET conjuntocódigos TERRITORY territorio]
[COLLATE USING {SYSTEM | IDENTITY | USER :udcs}]
[NUMSEGS númsegs] [DFT_EXTENT_SZ tamañoext_omisión]
[CATALOG TABLESPACE <definición_espaciotablas>]
[USER TABLESPACE <definición_espaciotablas>]
[TEMPORARY TABLESPACE <definición_espaciotablas>]
[WITH comment]
```

Donde <definición_espaciotablas> indica:
 MANAGED BY {
 SYSTEM USING :SMS_string |
 DATABASE USING :DMS_string }
 [EXTENTSIZE número_de_páginas]
 [PREFETCHSIZE número_de_páginas]
 [OVERHEAD número_de_milisegundos]
 [TRANSFERRATE número_de_milisegundos]

Parámetros de la API de REXX

nombrebd

Nombre de la base de datos.

aliasbd

Alias de la base de datos.

víaAcceso

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 **dfbdbpath**).

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 base de datos no se establezca en una vía de acceso montada con NFS (por ejemplo, en sistemas operativos Linux y UNIX, no debería especificar el directorio \$HOME del propietario de la instancia). La vía de acceso especificada para esta API en un entorno de base de datos particionada no puede ser una vía de acceso relativa.

conjuntocódigos

Conjunto de códigos que se debe utilizar para los datos entrados en la base de datos.

territorio

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 las bases de datos Unicode.

USER *udcs*

La aplicación que llama especifica el orden de clasificación en una variable del lenguaje principal que contiene una serie de 256 bytes que define el orden de clasificación.

númsegs

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.

tamañoext_omisión

Especifica el tamaño de extensión por omisión de los espacios de tablas de la base de datos.

SMS_string

Variable del 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 del 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 del 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 del 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 grabarán 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 el tiempo de latencia, búsqueda de disco y uso de controlador de E/S 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 autorizaciones 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 operativos Linux y UNIX, se almacena en el directorio de instalación DB2 (sql11ib, por ejemplo).

sqlectnd - Catalogar una entrada en el directorio de nodos

Si el almacenamiento en antememoria de directorios está habilitado, 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 (sólo servidor) en DB2, detenga (mandato **db2stop**) y vuelva a iniciar (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 nombre_instancia [WITH comment]
```

Parámetros de la API de REXX, opción 1

nodename

Alias del nodo que se debe catalogar.

nombre_instancia

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 nombre_sistema INSTANCE nombre_instancia
```

Parámetros de la API de REXX, opción 2

nodename

Alias del nodo que se debe catalogar.

nombre_sistema

Nombre de sistema del nodo donde reside la base de datos de destino.

nombre_instancia

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.

nombresistpral

Nombre de sistema principal o dirección IPv4 o dirección IPv6 del nodo donde reside la base de datos de destino.

nombreservicio

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

sqlectnd - Catalogar una entrada en el directorio de nodos

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.

nombresistpral

Nombre de sistema principal o dirección IPv4 o dirección IPv6 del nodo donde reside la base de datos de destino.

nombreservicio

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.

nombresistpral

Nombre de sistema principal o dirección IPv4 o dirección IPv6 del nodo donde reside la base de datos de destino.

nombreservicio

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 autorizaciones 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 nulo, se utiliza el directorio de bases de datos del sistema.

El comentario sólo se cambia en el directorio de bases de datos locales o del sistema del servidor de particiones de base de datos donde se ejecuta

sqlcdcgd - Cambiar un comentario de base de datos en el directorio de bases de datos locales o del sistema

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 sqlcdcgd

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 debe estar catalogado en el directorio de base de datos local. Si no se especifica la vía de acceso, el alias de base de datos debe estar catalogado 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.

vía de acceso

Vía de acceso donde reside la base de datos.

sqledcgd - Cambiar un comentario de base de datos en el directorio de bases de datos locales o del sistema

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 permiten caracteres de retorno de carro ni 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 base de datos particionada.

Ámbito

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

Autorización

Una de las autorizaciones 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.

sqledpan - Descartar una base de datos de un servidor de particiones de base de datos

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 autorizaciones 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 (
    _SQLLDCHAR * pDbAlias,
    _SQLLDCHAR * pReserved2,
    struct sqlca * pSqlca);
```

```
SQL_API_RC SQL_API_FN
sqlgdrpd (
    unsigned short Reserved1,
    unsigned short DbAliasLen,
    struct sqlca * pSqlca,
    _SQLLDCHAR * pReserved2,
    _SQLLDCHAR * 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 cronológicas se necesitan para una recuperación en avance tras una operación de restauración, los archivos de anotaciones cronológicas deben guardarse antes de llamar a 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 base de datos del sistema del cliente. El nombre de la base de datos correspondiente se elimina del directorio de base de datos del sistema del servidor.

Sintaxis de la API de REXX

```
DROP DATABASE dbalias
```

Parámetros de la API de REXX

aliasbd

Alias de la base de datos que se debe descartar.

sqlldrpn - 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 autorizaciones 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
sqlldrpn (
    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 sqlldrpn

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 NULL 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

sqledrpn - Comprobar si se puede descartar un servidor de particiones de base de datos

entrada del servidor de particiones de base de datos en el archivo `db2nodes.cfg`, con lo cual se elimina el servidor del entorno de base de datos particionada.

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 eliminar una partición de base de datos de un grupo de particiones de base de datos, puede utilizar la opción de la API `sqludrtdt` 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

sqlfmem - 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
sqlfmem (
    struct sqlca * pSqlca,
    void * pBuffer);
```

```
SQL_API_RC SQL_API_FN
sqlgfmem (
    struct sqlca * pSqlca,
    void * pBuffer);
```

Parámetros de la API sqlfmem

pSqlca

Salida. Puntero a la estructura sqlca

pBuffer

Entrada. Puntero a la memoria que se debe liberar.

sqlfrce - 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 base de datos particionada, 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 base de datos particionada.

Autorización

Una de las autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT

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
sqlfrce (
    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 sqlfrce

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.

sqlfrce - Desconectar usuarios y aplicaciones del sistema

Si este parámetro se establece en `SQL_ALL_USERS` (definido en `sqlenv`), se forzarán todas las aplicaciones que tengan conexiones de base de datos o adjunciones de instancia. Si se establece en cero, se devolverá 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 `sqlfrce`. 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_ASYNC` (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 haber ejecutado un mandato `force`, la base de datos seguirá aceptando peticiones de conexión. Es posible que se necesiten operaciones de forzar adicionales para forzar a todos los usuarios a que salgan. Las funciones de supervisión del sistema de base de datos se utilizan para obtener los ID de agente de los usuarios a los que se aplicará la operación de forzar.

Cuando la modalidad de desconexión se establece en `SQL_ASYNC` (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 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 ejecutará una operación de forzar en todos los usuarios a los que se puede aplicar. Si no se puede encontrar uno o varios de los ID de agente especificados, **sql code** en la estructura `sqlca` se establecerá en 1230. Por ejemplo, puede que no se encuentre un ID de agente si el usuario finaliza la sesión en el periodo de tiempo comprendido entre que se recopila el ID de agente y se llama a `sqlfrce`. El usuario que llama a esta API nunca se ve forzado a salir.

sqlefrce - Desconectar usuarios y aplicaciones del sistema

Los ID de agente se reciclan, y se utilizan para forzar las aplicaciones algún tiempo después de que el supervisor del sistema de base de datos haya obtenido los ID de agente. Cuando un usuario finaliza sesión, otro usuario puede iniciar sesión y obtener mediante ese proceso de reciclaje el mismo ID de agente, con lo que es posible que se aplique la operación de forzar en 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 las aplicaciones que tienen conexiones de base de datos y las que tienen conexiones de instancias.

agentidarray

Variable compuesta de lenguaje principal de REXX que contiene la lista de ID de agentes que se terminarán. En la información siguiente, XXX es el nombre de la variable del 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.

sqlgdad - 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 autorizaciones 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
sqlgdad (
    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 sqlgdad

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 servidores Power System

sqlegdad - Catalogar una base de datos en el directorio de DCS (Database Connection Services)

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 el almacenamiento en antememoria de directorios está habilitado, 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 (sólo servidor) en DB2, detenga (**db2stop**) el gestor de bases de datos y, a continuación, vuelva a iniciar (**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 nombrebd [AS nombrebd_destino]  
[AR nombrear] [PARMS paráms] [WITH comentario]
```

Parámetros de la API de REXX

dbname

Nombre de la base de datos local de la entrada de directorio que se debe añadir.

nombrebd_destino

Nombre de la base de datos de destino.

nombrear

Nombre del cliente de aplicaciones.

paráms

Serie de parámetros. Si se especifica, la serie debe escribirse entre comillas dobles.

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  
sqlgdccl (  
    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 de bases de datos (DCS).

Autorización

Una de las autorizaciones 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. Especifique el campo **ldb** de esta estructura incluyendo el nombre local de la base de datos que ha de suprimirse. 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 u nodo, utilice las API sqlegdsc, sqlegdgt y sqlegdcl.

Si se habilita el almacenamiento en antememoria de directorios 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 (sólo servidor) en DB2, detenga (**db2stop**) el gestor de bases de datos y, a continuación, vuelva a iniciar (**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 del 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

sqlgdge - 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
sqlgdge (
    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 sqlgdge

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 del 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

sqlegdgc - Obtener una entrada específica del directorio de DCS (Database Connection Services)

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

Debe llamarse a 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 del 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 mismo directorio cambia después de una llamada a esta API. Utilice la API sqlegdgt y la API sqlegdcl para liberar los recursos asociados a la llamada a 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 llamador de la exploración utiliza el valor devuelto **pNumEntries** para asignar suficiente memoria para recibir las entradas. Si se recibe una llamada de exploración mientras se está reteniendo una copia, se liberará la copia anterior y se recogerá la nueva copia.

Sintaxis de la API de REXX

```
OPEN DCS DIRECTORY
```

sqlgins - 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
sqlgins (
    _SQLOLDCHAR * pInstance,
    struct sqlca * pSqlca);
```

```
SQL_API_RC SQL_API_FN
sqlggins (
    struct sqlca * pSqlca,
    _SQLOLDCHAR * pInstance);
```

Parámetros de la API sqlgins

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 `sqlcatin - 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 del lenguaje principal de REXX en la que se debe colocar el nombre de la instancia del gestor de bases de datos.

sqlintr - Interrumpir peticiones de aplicaciones

Detiene una petición. Se llama a esta API desde un manejador de señales Control-Inter de una aplicación.

El manejador de señales Control-Inter puede ser el manejador 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
    sqlintr ( void );
```

```
SQL_API_RC SQL_API_FN
    sqlgintr (
        void);
```

Parámetros de la API `sqlintr`

Ninguna

Notas de uso

No se debe invocar ninguna API del gestor de bases de datos desde un manejador de interrupciones, excepto `sqleisig`. Sin embargo, el sistema no impedirá que se realice esa acción.

Las transacciones de base de datos que estén en estado de confirmación o retroacción 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 que existía con anterioridad a la invocación de la sentencia de SQL.
Otras sentencias de SQL	Las transacciones de base de datos están establecidas en el estado que existía con anterioridad 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 (por lo general, Control-C o Control-Inter, o ambos). Cuando este manejador por omisión detecta una señal de interrupción, restablece la señal y llama a 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 no tiene manejador de señales y se recibe una interrupción, se termina la aplicación. Esta API proporciona una gestión sencilla de señales y se puede utilizar si una aplicación no tiene muchos requisitos de 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 manejadores de señales instalados previamente funcionan debidamente.

Sintaxis de la API de REXX

```
INSTALL SIGNAL HANDLER
```

sqlcngdb - Actualizar la versión anterior de la base de datos DB2 a la versión actual

Convierte una versión anterior (versión 8 o superior) de una base de datos DB2 al release actual.

Las API sqlcngdb y sqlmgdb han quedado en desuso y dejarán de mantenerse en futuros releases. En su lugar deberá utilizar la nueva API db2DatabaseUpgrade.

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
sqlcngdb (
    _SQLOLDCHAR * pDbAlias,
    _SQLOLDCHAR * pUserName,
    _SQLOLDCHAR * pPassword,
    struct sqlca * pSqlca);
```

```
SQL_API_RC SQL_API_FN
sqlmgdb (
    unsigned short PasswordLen,
    unsigned short UserNameLen,
    unsigned short DbAliasLen,
    struct sqlca * pSqlca,
    _SQLOLDCHAR * pPassword,
    _SQLOLDCHAR * pUserName,
    _SQLOLDCHAR * pDbAlias);
```

Parámetros de la API sqlcngdb

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 sqlmgdb

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 actualizará una base de datos a una versión más reciente, y no puede utilizarse para la conversión de una base de datos actualizada en su versión anterior.

La base de datos debe catalogarse antes de la migración.

Sintaxis de la API de REXX

```
MIGRATE DATABASE aliasbd [USER nombreusuario USING contraseña]
```

Parámetros de la API de REXX

aliasbd

Alias de la base de datos que ha de actualizarse.

nombreusuario

Nombre de usuario utilizado para reiniciar la base de datos.

contraseña

Contraseña utilizada para autenticar 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 del 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 rellenos con blancos por la derecha.

El valor **sqlcode** de 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 del lenguaje principal de REXX que contiene el identificador devuelto por la API Abrir exploración de directorio de nodos.
- value** Variable del lenguaje principal de REXX compuesta a la que se devuelve información sobre la entrada de nodo. Si no se suministra ningún nombre, se utiliza el nombre SQLEINFO. En la información que sigue a continuación, XXX representa el nombre de la variable del 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 a 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. No obstante, 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 del lenguaje principal.

XXX.0 Número de elementos de la variable (siempre 2)

XXX.1 Especifica una variable del lenguaje principal de REXX que contiene un número para el ID de exploración

XXX.2 Número de entradas contenidas en el directorio.

sqlqryc - 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
sqlqryc (
    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 `sqlqryc`

`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 de opciones de conexión que se deben devolver.

`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 del lenguaje principal.

XXX.1 Valor de Currentconnection para el tipo CONNECTION

XXX.2 Valor de Currentconnection para SQLRULES

XXX.3 Valor de Currentconnection que indica las conexiones que se liberarán cuando se emita COMMIT.

XXX.4 Valor de Currentconnection de la opción **SYNCPPOINT**. La opción **SYNCPPOINT** 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 Currentconnection para una sentencia PREPARE aplazada.

sqlqryi - Consultar información sobre el cliente

Devuelve información de cliente existente rellenando los campos de la estructura de datos `sqle_client_info`.

Dado que esta API permite especificar un alias de base de datos, una aplicación puede consultar información de cliente asociada a una conexión específica. Devuelve un valor nulo si la API `sqleseti` no ha establecido anteriormente un valor.

Si se solicita una conexión específica, esta API devuelve los últimos valores de la conexión. Si se especifican todas las conexiones, la API devuelve los valores que se asociarán a todas las conexiones; es decir, los valores que se pasaron en la última llamada a `sqleseti` (especificando 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
sqlqryi (
    unsigned short DbAliasLen,
    char * pDbAlias,
    unsigned short NumItems,
    struct sqle_client_info* pClient_Info,
    struct sqlca * pSqlca);
```

Parámetros de la API `sqlqryi`

`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 correcta, los valores actuales se devolverán 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`.

sqlsact - 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
sqlsact (
    char * pAccountingString,
    struct sqlca * pSqlca);
```

```
SQL_API_RC SQL_API_FN
sqlgsact (
    unsigned short AccountingStringLength,
    char * pAccountingString,
    struct sqlca * pSqlca);
```

Parámetros de la API sqlsact

pAccountingString

Entrada. Serie que contiene los datos contables.

pSqlca

Salida. Puntero a la estructura sqlca.

Parámetros específicos de la API sqlgsact

AccountingStringLength

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 sólo los caracteres A a la Z, 0 al 9, y el carácter de subrayado (_).

sqlesdeg - Establecer el nivel o grado máximo de paralelismo intrapartición para la ejecución de sentencias de SQL

Establecer el nivel o grado máximo de paralelismo intrapartición para la ejecución de sentencias de SQL para las 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 autorizaciones 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 se aplicará a todas las aplicaciones activas. Si se establece en cero, se devolverá 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`.

sqlsdeg - Establecer el nivel o grado máximo de paralelismo intrapartición para la ejecución de sentencias de SQL

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 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, puede que no se encuentre un ID de agente si el usuario finaliza la sesión en el periodo de tiempo comprendido entre que se recupila el ID de agente y se llama a 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.

sqlsetc - 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 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
sqlsetc (
    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 sqlsetc

pConnectionSettings

Entrada. Un puntero para la estructura `sqle_conn_setting`, que especifica los 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 necesario para la opción.

NumSettings

Entrada. Cualquier valor entero (comprendido entre 0 y 7) que representa el número de valores de opciones de conexión que se deben establecer.

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`).

Cuando la API de `SET CLIENT` se haya ejecutado correctamente, los valores de conexión se fijarán y sólo se podrán cambiar volviendo a ejecutar la misma API.

sqlsetc - Establecer valores de conexión del cliente

Todas las opciones precompiladas correspondientes de los módulos de aplicación se alterarán temporalmente.

Sintaxis de la API de REXX

```
SET CLIENT USING :values
```

Parámetros de la API de REXX

valores

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 del lenguaje principal.

XXX.0 Número de valores de conexión que se deben definir.

XXX.1 Especifica cómo configurar el tipo CONNECTION. 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 configurar el ámbito de desconexión en bases de datos en la confirmación. Los valores válidos son:

EXPLICIT

Desconectar solamente las marcadas con la sentencia RELEASE de SQL

sqlsetc - Establecer valores de conexión del cliente

CONDITIONAL

Desconectar solamente las que no tengan cursores WITH HOLD abiertos

AUTOMATIC

Desconectar todas las conexiones

XXX.4 Especifica cómo configurar 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 sentencia OPEN, DESCRIBE o EXECUTE correspondiente. No obstante, no se diferirá la sentencia PREPARE INTO

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 `sqlc_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 retroacción. 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 de la conexión con una base de datos, o puede utilizarse para definir o modificar los valores después de haberse establecido 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 sqlc_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.

sqlesei - Establecer información sobre el cliente

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 Universal Database 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 `sqlesei` en un programa CLI antes de establecer una conexión no funcionará. La llamada a la API `sqlesei` 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()`.

Si la API establece la serie de contabilidad, cuando se hacen cambios en el nombre de la aplicación o el ID de usuario cliente, la serie de contabilidad no se actualiza. Sin embargo, si la serie de contabilidad se ha establecido mediante el

sqleseti - Establecer información sobre el cliente

procedimiento `wlm_set_client_info` y se cambia el ID de usuario cliente o el nombre de la aplicación, se actualizará la serie de contabilidad.

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 autorizaciones 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 sqledrpd.

Para listar las bases de datos que están catalogadas en un nodo, utilice las API db2DbDirOpenScan, db2DbDirGetNextEntry y db2DbDirCloseScan.

sqleuncd - Descatalogar una base de datos del directorio de bases de datos del sistema

El tipo de autenticación 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 el almacenamiento en antememoria de los directorios 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 (sólo servidor) en DB2, detenga (**db2stop**) el gestor de bases de datos y, a continuación, vuelva a iniciar (**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.

sqlleuncn - Descatalogar una entrada del directorio de nodos

Suprime una entrada del directorio de nodos.

Autorización

Una de las autorizaciones 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
sqlleuncn (
    _SQLLOLDCHAR * pNodeName,
    struct sqlca * pSqlca);
```

```
SQL_API_RC SQL_API_FN
sqlguncn (
    unsigned short NodeNameLen,
    struct sqlca * pSqlca,
    _SQLLOLDCHAR * pNodeName);
```

Parámetros de la API sqlleuncn

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 el almacenamiento en antememoria de los directorios 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 (sólo servidor) en DB2, detenga (**db2stop**)

sqleuncn - Descatalogar una entrada del directorio de nodos

el gestor de bases de datos y, a continuación, vuelva a iniciar (**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 principales, 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 de un almacenamiento intermedio definido por un puntero a una variable a la que la aplicación puede acceder directamente.

Esta API se utiliza en lenguajes de programación principales, como FORTRAN y COBOL, que no proporcionan mecanismos para el manejo de punteros. Esta API puede utilizarse para obtener resultados de las API que devuelven un puntero a los datos necesarios.

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. Un puntero al área que contiene los datos necesarios.

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 principales (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 sin signo que representa el número de bytes que se debe transferir.

sqllogstt - 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
sqllogstt (
    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 sqllogstt

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.

sqllogstt - Obtener el mensaje de SQLSTATE

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 a nivel de palabras. Si no se especifica un valor o este parámetro se establece en 0, se devuelve el texto del mensaje sin divisiones de línea.

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 partición de base de datos es la partición del catálogo para 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 autorizaciones siguientes:

- SYSADM
- SYSCTRL
- DBADM

Asimismo, también es necesario uno de los grupos de autorizaciones siguientes:

- Privilegios DELETE, INSERT y SELECT en todas las tablas del grupo de particiones de base de datos que se está redistribuyendo
- Autorización DATAACCESS

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,
```

sqlldrdt: redistribuir datos a través de un grupo de particiones de base de datos

```
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 sqlldrdt

pNodeGroupName

Nombre del grupo de particiones de base de datos que se va a redistribuir.

pTargetPMapFileName

Nombre del archivo que contiene la correlación 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

sqlldr: redistribuir datos a través de un grupo de particiones de base de datos

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 sqlldr

NodeGroupNameLen

Longitud del nombre del grupo de particiones de base de datos.

TargetPMapFileNameLen

Longitud del nombre del archivo de la correlación de distribución de destino.

DataDistFileNameLen

Longitud del nombre del archivo de distribución de datos.

Notas de uso

Cuando finaliza una operación de redistribución, se graba un archivo de mensaje en:

- El directorio `$HOME/sql/lib/redist` de los sistemas operativos Linux y UNIX, mediante la utilización del siguiente formato para los subdirectorios y el nombre de archivo: *nombre-base-datos.nombre-grupo de nodos.indicación-fecha-hora*.
- El directorio `$HOME\sql\lib\redist\` del sistema operativo Windows, mediante la utilización del siguiente formato para los subdirectorios y el nombre de archivo: *nombre-base-datos\primeros-ocho-caracteres-de-nombre-grupo-nodos\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.

sqludrdt: redistribuir datos a través de un 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 mediante la emisión de 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

A partir de la Versión 9.7, esta API está en desuso. Utilice la API `db2GetRowPartNum` (Obtener el número de servidor de particiones de base de datos de una fila) para que devuelva el número de partición de base de datos y el número del servidor de particiones de base de datos de una fila.

Si llama a la API `sqlugrpn` y la variable de registro `DB2_PMAP_COMPATIBILITY` se ha establecido en `OFF`, se devolverá el mensaje de error `SQL2768N`.

Devuelve el número de partición de base de datos y el número de servidor de particiones de base de datos en función de los valores de clave de distribución. Una aplicación puede utilizar esta información para determinar en qué servidor de particiones de base de datos está almacenada una fila específica de la tabla.

La estructura de datos de particionamiento, `sqlupi`, es la entrada de esta API. La API `sqlugtpi` puede devolver la estructura. Otra entrada posible es la representación de caracteres de los valores de clave de distribución correspondientes. La salida es un número de partición de base de datos que se genera a través de la estrategia de distribución y el número de servidor de particiones de base de datos correspondiente a partir de la correlación de distribución. Si no se proporciona la información de la correlación de distribución, solo se devuelve el número de partición de base de datos. Esto puede resultar de utilidad cuando se analiza la distribución de datos.

No es necesario que el gestor de bases de datos esté en ejecución cuando se invoca esta API.

Ámbito

Esta API debe invocarse desde un servidor de particiones de base de datos en el archivo `db2nodes.cfg`. Esta API no se debe invocar desde un cliente, pues podría devolverse información errónea sobre el particionamiento de la base de datos, debido a las diferencias existentes en la página de códigos y en el formato endian 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_ctypecode,
    unsigned short codepage,
    struct sqlupi * part_info,
    short * part_num,
    SQL_PDB_NODE_TYPE * node_num,
    unsigned short chklvl,
    struct sqlca * sqlca,
```


sqlugrpn - Obtener el número de servidor de particiones de base de datos para una fila

```
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->sqlid**.

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_ctypecode

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 comprobación realizada sobre los parámetros de entrada. Si el valor especificado es cero, no se

sqlugrpn - Obtener el número de servidor de particiones de base de datos para una fila

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 numéricos y de fecha y hora, las representaciones de caracteres deben encontrarse en la página de códigos del correspondiente sistema desde el que se invoca a la API.

Si **node_num** no es nulo, se debe proporcionar la correlación 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 **sqlld** del parámetro **part_info** (**part_info->sqlld**) 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.

A partir de la versión DB2 9.7, esta API está en desuso. Utilice la API `db2GetDistMap` (obtener correlación de distribución) para que se devuelva la información de distribución. Si llama a la API `sqlugtpi` y la variable de registro **DB2_PMAP_COMPATIBILITY** se ha establecido en `OFF`, se devolverá el mensaje de error `SQL2768N`.

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 a la que se hace referencia, el usuario debe tener, como mínimo, una de las autorizaciones siguientes:

- Autorización `DATAACCESS`
- 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.

qlugtpi - Obtener información de distribución de tablas

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 base de datos particionada, 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 autorizaciones siguientes:

- SYSADM
- SYSCTRL
- SYSMANT
- 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

sqluvqdp - Inmovilizar espacios de tablas para una tabla

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

Para restablecer el estado de los espacios de tablas en el estado normal si se cumple cualquiera de las condiciones siguientes:

- 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 está soportada en entornos DB2 pureScale.

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.

sqluvqdp - Inmovilizar espacios de tablas para una tabla

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 tablas. 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 modificar el estado de un espacio de tablas de un estado menos restrictivo a otro más restrictivo (por ejemplo, S a U o bien 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).

sqluvqdp - Inmovilizar espacios de tablas para una tabla

Capítulo 6. Llamada a las API de DB2 en REXX

Sintaxis para llamar 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.

Nota: Para obtener información acerca de las versiones soportadas del lenguaje de programación REXX, consulte “Soporte para el desarrollo de aplicaciones de bases de datos en REXX” en *Getting Started with Database Application Development*.

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).

La llamada a una API de DB2 mediante SQLDB2 equivale a llamar directamente al CLP, excepto por las diferencias 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

Llamada a las API de DB2 en REXX

- 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 enumeradas 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 repetitiva.
- CS** Estabilidad del cursor. Es el valor por omisión.
- UR** Lectura no confirmada.
- RS** Estabilidad de lectura.
- NC** Sin confirmación.

Cambiar el nivel de aislamiento

Capítulo 7. Las API de gestión de transacciones dudosas

Es probable que haya veces en las que sea útil poder realizar una consulta, confirmación o retrotracción de transacciones dudosas sin esperar a que el gestor de transacciones realice una acción de *resincronización*. Esta situación podría producirse si, por ejemplo, se interrumpe la línea de comunicaciones y una transacción dudosa está ocupando recursos necesarios.

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 bases de datos, no puede esperar a que el gestor de transacciones efectúe la operación *re-sync*. 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.

Las API de gestión de transacciones dudosas

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 el nivel de release de la estructura que se ha pasado 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 rnm;
    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 el nivel de release de la estructura que se ha pasado 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 se encuentran en el formato db2XaRecoverStruct. La aplicación puede recorrer la lista de transacciones dudosas mediante el tamaño de la estructura db2XaRecoverStruct, empezando por la dirección suministrada 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 **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.

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 **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.

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

db2XaListIndTrans - Listar transacciones dudosas

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

db2XaListIndTrans - Listar transacciones dudosas

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 **originator** 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 `include`) 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` situado en el directorio `include`) 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 podido confirmar o retrotraer una transacción.

db2XaListIndTrans - Listar transacciones dudosas

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

nombre Salida. Nombre de una fuente de datos federada.

pxid 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 las transacciones federadas participan menos de 10 fuentes de datos. Si más de 16 fuentes de datos federadas no pueden confirmarse ni retrotraerse en una transacción, la API db2XaListIndTrans sólo devolverá 16 de éstas para esta transacción dudosa. Para una transacción dudosa no federada, el parámetro **rmn** 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 la API db2XaListIndTrans 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 no sea suficientemente grande si hay más transacciones dudosas adicionales 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 estos pasos.

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 ser 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 8. Aplicaciones por hebras con acceso simultáneo

sqliteAttachToCtx - 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
sqliteAttachToCtx (
    void * pCtx,
    void * reserved,
    struct sqlca * pSqlca);
```

Parámetros de la API sqliteAttachToCtx

pCtx Entrada. Contexto válido asignado previamente por sqliteBeginCtx.

reserved

Reservado para una utilización futura. Su valor se debe establecer en NULL.

pSqlca

Salida. Puntero a la estructura sqlca.

sqlBeginCtx - Crear y conectar a un contexto de aplicación

Creación de un contexto de aplicación, o bien creación y conexión a un contexto de aplicación.

Se puede crear más de un contexto de aplicación. Cada contexto tiene su propio ámbito de confirmación. Hebras diferentes se pueden enlazar a contextos diferentes (vea la API `sqlAttachToCtx`). 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
sqlBeginCtx (
    void ** ppCtx,
    sqlint32 lOptions,
    void * reserved,
    struct sqlca * pSqlca);
```

Parámetros de la API `sqlBeginCtx`

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 `sqlAttachToCtx` para la hebra actual. Si se utiliza esta opción, el parámetro **ppCtx** puede ser NULL. 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`.

sqlDetachFromCtx - Desconectar de contexto

Desconecta el contexto que está utilizando la hebra actual.

El contexto sólo se desconecta 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
sqlDetachFromCtx (
    void * pCtx,
    void * reserved,
    struct sqlca * pSqlca);
```

Parámetros de la API sqlDetachFromCtx

pCtx Entrada. Contexto válido asignado previamente por sqlBeginCtx.

reserved

Reservado para una utilización futura. Su valor se debe establecer en NULL.

pSqlca

Salida. Puntero a la estructura sqlca.

sqlEndCtx - 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
sqlEndCtx (
    void ** ppCtx,
    sqlint32 lOptions,
    void * reserved,
    struct sqlca * pSqlca);
```

Parámetros de la API sqlEndCtx

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 `sqlBeginCtx`.

SQL_CTX_END_ALL

Si es necesario, se realiza una llamada a `sqlDetachFromCtx` 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 NULL, pero si se pasa el parámetro, debe ser un contexto válido asignado previamente por `sqlBeginCtx`. Se realiza una llamada a `sqlGetCurrentCtx`, 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`.

sqlEndCtx - Desconectar y liberar la memoria asociada a un contexto de aplicación

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 (por ejemplo mediante sqlEndCtx antes de invocar sqlEndCtx.

sqlGetCurrentCtx - 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
sqlGetCurrentCtx (
    void ** ppCtx,
    void * reserved,
    struct sqlca * pSqlca);
```

Parámetros de la API sqlGetCurrentCtx

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.

sqlInterruptCtx - 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  
sqlInterruptCtx(  
    void * pCtx,  
    void * reserved,  
    struct sqlca * pSqlca);
```

Parámetros de la API sqlInterruptCtx

pCtx Entrada. Contexto válido asignado previamente por sqlBeginCtx.

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.

sqlSetTypeCtx - Definir el tipo de contexto de aplicación

Establece el tipo de contexto de la aplicación.

Esta API debería ser la primera API de la base de datos a la que se llame 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
sqlSetTypeCtx (
    sqlint32 IOptions);
```

Parámetros de la API sqlSetTypeCtx

IOptions

Entrada. Los valores válidos son:

SQL_CTX_ORIGINAL

Todas las hebras utilizarán el mismo contexto y se bloqueará el acceso simultáneo. Es el valor por omisión si no se llama a 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 sqlBeginCtx
- API sqlAttachToCtx
- API sqlDetachFromCtx
- API sqlEndCtx

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 sqlintr interrumpe todos los contextos. Para interrumpir un contexto determinado, utilice sqlInterruptCtx.

sqlSetTypeCtx - Definir el tipo de contexto de aplicación

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 9. 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 estos tres plugins 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.

Despliegue de un plugin de recuperación de grupos

Antes de empezar

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.

Procedimiento

- 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 autenticación basado en un ID de usuario/contraseña del sistema de seguridad de DB2, puede desarrollar sus propios plugins de autenticación por ID de usuario/contraseña o adquirirlos de terceros.

Antes de empezar

Dependiendo de la utilización prevista, todos los plugins de autenticación 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 autenticación 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 autenticación por ID de usuario/contraseña que sean apropiados para su sistema de gestión de bases de datos, puede desplegar los plugin.

Procedimiento

- Para desplegar un plugin de autenticación 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 autenticación 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. Uno de los siguientes:
 - Establezca el parámetro de configuración del gestor de bases de datos **srvcon_auth** en el tipo de autenticación CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT o DATA_ENCRYPT_CMP. O bien:
 - Establezca el parámetro de configuración del gestor de bases de datos **srvcon_auth** en NOT_SPECIFIED y establezca **authentication** en el tipo de autenticación CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT o DATA_ENCRYPT_CMP.
- Para desplegar un plugin de autenticación por ID de usuario/contraseña en clientes de bases de datos, siga estos pasos en cada cliente:
 1. Copie la biblioteca de plugins de autenticación 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 autenticación, 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 autenticación por ID de usuario/contraseña, siga estos pasos en cada cliente, servidor o pasarela:
 1. Copie la biblioteca del plugin de autenticación 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. Establezca el parámetro de configuración del gestor de bases de datos **authentication** en el tipo de autenticación CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT o DATA_ENCRYPT_CMP.

Despliegue de un plugin de GSS-API

Si desea personalizar el mecanismo de autenticación del sistema de seguridad de DB2, puede desarrollar sus propios plugins de autenticación utilizando GSS-API o adquirirlos de terceros.

Antes de empezar

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

Despliegue de un plugin de GSS-API

DB2. Dependiendo de la utilización prevista, todos los plugins de autenticación 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 autenticación de GSS-API que sean apropiados para su sistema de gestión de bases de datos, puede desplegar los plugin.

Procedimiento

- Para desplegar un plugin de autenticación de GSS-API en el servidor de bases de datos, siga estos pasos en el servidor:
 1. Copie la biblioteca de plugins de autenticación 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. Uno de los siguientes:
 - Establecer el parámetro de configuración **srvcon_auth** del gestor de bases de datos en GSSPLUGIN o GSS_SERVER_ENCRYPT es un modo de habilitar el servidor para que utilice el método de autenticación GSSAPI PLUGIN. O bien:
 - Establecer el parámetro de configuración **srvcon_auth** del gestor de bases de datos en NOT_SPECIFIED y el valor **authentication** en GSSPLUGIN o GSS_SERVER_ENCRYPT es un modo de habilitar el servidor para que utilice el método de autenticación GSSAPI PLUGIN.
- Para desplegar un plugin de autenticación de GSS-API en clientes de bases de datos, siga estos pasos en cada cliente:
 1. Copie la biblioteca de plugins de autenticación 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 autenticación 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 autenticación un plugin de autenticación 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 autenticación de GSS-API, siga estos pasos:

1. Copie la biblioteca de plugins de autenticación 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. Establezca el parámetro de configuración **authentication** del gestor de bases de datos en GSSPLUGIN o GSS_SERVER_ENCRYPT.

Despliegue de un plugin de Kerberos

Para personalizar el mecanismo de autenticación Kerberos del sistema de seguridad de DB2, puede desarrollar sus propios plugins de autenticación Kerberos o adquirir uno de terceros.

Antes de empezar

Si desea desplegar una nueva versión de un plugin existente, desde detener el servidor de DB2 y las aplicaciones que utilizan el plugin. Pueden producirse comportamientos indefinidos, incluidas interrupciones, si un proceso utiliza un plugin al desplegar una nueva versión de ese plugin (con el mismo nombre).

Acerca de esta tarea

El plugin de autenticación Kerberos puede desplegarse en un servidor de bases de datos o un cliente de base de datos.

Procedimiento

- Para desplegar un plugin de autenticación de Kerberos en el servidor de bases de datos, siga estos pasos en el servidor:
 1. Copie la biblioteca de plugins de autenticación de Kerberos en el directorio de plugins del servidor.
 2. Actualice el valor del parámetro de configuración del gestor de bases de datos **srvcon_gssplugin_list**, que es una lista ordenada y delimitada por comas, para incluir el nombre del plugin del servidor Kerberos. Un solo plugin de esta lista puede ser un plugin Kerberos. Si no hay ningún plugin de Kerberos en la lista, se devuelve un error. Si hay más de un plugin de Kerberos en la lista, se devuelve un error. Si el valor del parámetro de configuración está en blanco y el parámetro de configuración **authentication** se establece en KERBEROS o KRB_SVR_ENCRYPT, se utiliza el plugin de Kerberos por omisión, DB2 Kerberos plug-in, IBMkrb5.
 3. Si es necesario, establezca el valor del parámetro de configuración **srvcon_auth** del gestor de bases de datos. Si desea desplegar un plugin de Kerberos, los valores aceptables para el parámetro de configuración **srvcon_auth** del gestor de bases de datos son los siguientes:
 - KERBEROS
 - KRB_SERVER_ENCRYPT
 - GSSPLUGIN
 - GSS_SERVER_ENCRYPT
 - En blanco, pero sólo si el parámetro de configuración **authentication** se establece en uno de los valores anteriores de esta lista.
- Para desplegar un plugin de autenticación Kerberos en un cliente de base de datos, lleve a cabo los pasos siguientes en el cliente:
 1. Copie la biblioteca de plugins de autenticación de Kerberos en el directorio de plugins del cliente.

Despliegue de un plugin de Kerberos

2. Establezca el parámetro de configuración `clnt_krb_plugin` del gestor de base de datos en el nombre del plugin de Kerberos. Si el valor del parámetro de configuración `clnt_krb_plugin` está en blanco, el cliente no puede utilizarla autenticación Kerberos. En Windows, el valor por omisión es `IBMkrb5`. Sólo debe modificarse para un plugin de Kerberos personalizado. En UNIX, debe establecerse el valor, porque el valor por omisión está en blanco. Para la autorización local en un cliente, servidor o pasarela, que haga uso de un plugin de autenticación Kerberos, siga estos pasos:
 - a. Copie la biblioteca de plugins de autenticación Kerberos en el directorio de plugins de cliente del cliente, servidor o pasarela.
 - b. Establezca el parámetro de configuración `clnt_krb_plugin` del gestor de base de datos en el nombre del plugin.
 - c. Establezca el parámetro de configuración `authentication` del gestor de bases de datos en `KERBEROS` o `KRB_SERVER_ENCRYPT`.
3. Opcional: catalogue las bases de datos a las que accederá el cliente, e indique que el cliente solo utilizará un plugin de autenticación Kerberos. En el ejemplo siguiente se cataloga la base de datos `testdb`:

```
CATALOG DB testdb AT NODE testnode AUTHENTICATION KERBEROS
TARGET PRINCIPAL service/host@REALM
```

Escritura de plugins de seguridad

Cómo carga DB2 los plugins de seguridad

Para que el sistema de base de datos DB2 tenga la información necesaria para llamar a las funciones de los plugins de seguridad, un plugin de seguridad debe tener la función de inicialización correctamente configurada.

Cada biblioteca de plugins debe contener una función de inicialización junto con un nombre específico determinado por el tipo de plugin:

- Plugin de autenticación del lado del servidor: `db2secServerAuthPluginInit()`
- Plugin de autenticación del lado del 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 añada mensajes de anotaciones a los archivos de anotaciones cronológicas **db2diag**
- 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 **errmsg,  
    db2int32 *errmsglen);
```

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 autenticación 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

Existen ciertas restricciones que influyen en el desarrollo de bibliotecas de plugins.

En la lista siguiente se describen las restricciones para desarrollar 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 utilizado en 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, como en las situaciones 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)

Restricciones en el desarrollo de bibliotecas de plugins de seguridad

- 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_USERNAME_SPACE_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 autenticación 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 anteriormente.

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

Restricciones en el desarrollo de bibliotecas de plugins de seguridad

shr64.o (64 bits). Un archivero individual puede contener miembros de 32 bits y de 64 bits, lo que permite desplegar el archivero en ambos tipos de plataformas.

Por ejemplo, para crear una biblioteca de plugins para archiveros de 32 bits:

```
xlc_r -qmkshrojb -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 -qmkshrojb -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.

Bifurcación

Las bibliotecas de plugins no deberían bifurcarse ya que se duplicarán los sockets y los descriptores en el proceso hijo, lo que puede provocar bloqueos o comportamientos incorrectos. En concreto, puede causar un conflicto de bloqueo de archivo falso si se bifurcó el hijo mientras había un descriptor de archivo abierto en el archivo. También puede suceder que la bifurcación herede muchos otros recursos como semáforos.

Restricciones para plugins de seguridad

Existen ciertas restricciones 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 autenticar 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 autenticar 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 autenticación 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 autenticación 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 autenticación a la que el usuario puede conectarse.

Restricciones de soporte de InfoSphere 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 autenticación 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

Restricciones para plugins de seguridad

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.	Todos
-1	DB2SEC_PLUGIN_UNKNOWNERROR	La API de plugins encontró un error inesperado.	Todos
-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

Códigos de retorno para plugins de seguridad

Tabla 10. Códigos de retorno de plugins de seguridad (continuación)

Código de retorno	Valor de sentencia define	Significado	API aplicable
-8	DB2SEC_PLUGIN_USER_REVOKED	Se ha revocado al usuario.	db2secValidatePassword db2GetGroupsForUser
-9	DB2SEC_PLUGIN_USER_SUSPENDED	Se ha suspendido al usuario.	db2secValidatePassword db2GetGroupsForUser
-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.	Todos
-14	DB2SEC_PLUGIN_DISKERROR	El plugin ha encontrado un error de disco.	Todos
-15	DB2SEC_PLUGIN_NOPERM	El plugin no ha podido acceder a un archivo debido a que no tiene los permisos correctos para el archivo.	Todos
-16	DB2SEC_PLUGIN_NETWORKERROR	El plugin ha encontrado un error de red.	Todos
-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.	Todos
-19	DB2SEC_PLUGIN_FILENOTFOUND	El plugin no puede abrir y leer un archivo porque el archivo falta en el sistema de archivos.	Todos
-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

Códigos de retorno para plugins de seguridad

Tabla 10. Códigos de retorno de plugins de seguridad (continuación)

Código de retorno	Valor de sentencia define	Significado	API aplicable
-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
-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.	Todos
-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.	Todos
-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.	Todos
-29	DB2SEC_PLUGIN_ROOT_NEEDED	El plugin está intentando ejecutar una aplicación que requiere privilegios root.	Todos
-30	DB2SEC_PLUGIN_UNEXPECTED_SYSTEM_ERROR	El plugin ha detectado un error inesperado del sistema. Existe la posibilidad de que la configuración del sistema actual no esté soportada.	Todos

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 `errmsg` para proporcionar una descripción del problema más específica que el código de retorno.

Por ejemplo, la serie de texto proporcionada en `errmsg` 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 `errmsg` 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 `errmsg` 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 los archivos de anotaciones cronológicas **db2diag** 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

La secuencia con la que el gestor de bases de datos DB2 llama a las API de plugins de seguridad es diferente en función del escenario en el que se llama a las API de plugins de seguridad.

Estas son las situaciones principales en las que el gestor de bases de datos DB2 llama 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 autenticación 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 autenticación 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 autenticación implícita si es necesario.

CLIENT - explícita

En una autenticación 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 llama 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 autenticación implícita, cuando el cliente y el servidor negocian la autenticación por ID de usuario/contraseña (por ejemplo, cuando se establece que el parámetro **srvcon_auth** en el servidor sea igual a SERVER;

Secuencias de llamada para las API de plugins de seguridad

SERVER_ENCRYPT, DATA_ENCRYPT o DATA_ENCRYPT_CMP), la aplicación cliente llama a las siguientes API de plugins de seguridad:

- db2secGetDefaultLoginContext();
- db2secFreeToken();

Basado en servidor (SERVER, SERVER_ENCRYPT, DATA_ENCRYPT) - explícita

En una autenticación explícita, cuando el cliente y el servidor negocian la autenticación 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 llama a las siguientes API de plugins de seguridad:

- db2secRemapUserid();

GSSAPI y Kerberos - implícita

En una autenticación implícita, cuando el cliente y el servidor negocian la autenticación 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 llama 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 autenticación 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 autenticación implícita y para la explícita, a menos que se indique lo contrario.

- db2secProcessServerPrincipalName();
- db2secGenerateInitialCred(); (para la autenticación 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 autenticación mutua desde el servidor y así lo exige la implementación.

Secuencias de llamada para las API de plugins de seguridad

Desde un cliente, servidor o pasarela para una autorización local

Para una autorización local, el mandato de DB2 utilizado llama a las siguientes API de plugins de seguridad:

- db2secGetDefaultLoginContext();
- db2secGetGroupsForUser();
- db2secFreeToken();
- db2secFreeGroupList();

Se llama a estas API para los mecanismos de autenticación basados en un ID de usuario/contraseña y para los mecanismos de autenticación 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 autenticación 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 llama a las siguientes API de plugins de seguridad en el caso del mecanismo de autenticación 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 llama 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 su servidor de bases de datos, el proceso agente o hebra de DB2 llama a la siguiente API de plugins de seguridad con sólo el ID de autorización como entrada:

- db2secGetGroupsForUser();

No se obtendrán símbolos procedentes de otros plugins de seguridad.

Plugins de seguridad

Para el sistema de base de datos DB2 la autenticación se realiza utilizando *plugins de seguridad*. Un plugin de seguridad es una biblioteca de carga dinámica que proporciona servicios de seguridad de autenticación.

Plugin de recuperación de grupos

Recupera información sobre pertenencia a grupos para un determinado usuario.

Plugin de autenticación por ID de usuario/contraseña

Los tipos siguientes de autenticación se implementan mediante un plugin de autenticación por ID de usuario y contraseña:

- CLIENT
- SERVER
- SERVER_ENCRYPT
- DATA_ENCRYPT
- DATA_ENCRYPT_CMP

Estos tipos de autenticación determinan cómo y dónde se autentifica un usuario. El tipo de autenticación que se utiliza se determina mediante el método siguiente:

- Para operaciones de conexión o unión, si especifica un valor para el parámetro de configuración **srvcon_auth**, ese valor tiene preferencia sobre el valor del parámetro de configuración **authentication**.
- En todos los demás casos, se utiliza el valor del parámetro de configuración **authentication**.

Plugin de autenticación GSS-API

La autenticación 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). El protocolo Kerberos es el medio predominante para implementar el mecanismo de autenticación GSS-API. Los plugins de autenticación de GSS-API aplican los tipos de autenticación siguientes:

- KERBEROS
- GSSPLUGIN
- KRB_SERVER_ENCRYPT
- GSS_SERVER_ENCRYPT

KRB_SERVER_ENCRYPT y GSS_SERVER_ENCRYPT admiten la autenticación GSS-API y de ID de usuario y contraseña. Sin embargo, la autenticación GSS-API es el tipo preferido de autenticación. Se proporciona soporte Kerberos para el cliente en los sistemas operativos Solaris, AIX, HP-UX (sólo 64 bits), Windows y Linux. Para los sistemas operativos Windows, el soporte de Kerberos está habilitado por omisión.

El gestor de bases de datos DB2 admite estos plugin en el cliente y el servidor.

Nota: Los tipos de autenticación determinan cómo y dónde se autentifica un usuario. Para utilizar un tipo de autenticación determinado, establezca el valor del parámetro de configuración **authentication** del gestor de bases de datos.

Puede utilizar cada uno de los plugins de forma independiente, o en combinación con los otros plugins. Por ejemplo, puede utilizar un plugin de autenticación del lado del servidor específico, pero aceptar los valores por omisión de DB2 para la

autenticación de clientes y grupos. Como alternativa, puede tener sólo un plugin de recuperación de grupo o de autenticación de clientes, pero sin un plugin del lado del servidor.

Si desea utilizar la autenticación GSS-API, son necesarios plugins tanto en el cliente como en el servidor.

El comportamiento por omisión para la autenticación es utilizar un plugin de ID de usuario/contraseña que implemente el mecanismo de nivel del sistema operativo para la autenticación.

El producto de base de datos DB2 incluye plugins para la recuperación de grupos, la autenticación por ID de usuario/contraseña y la autenticación GSS-API. Puede personalizar todavía más el comportamiento de autenticación 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 admitir un plugin de recuperación de grupos y un plugin de autenticación por ID de usuario/contraseña.

Como alternativa, los usuarios que utilizan el plugin de autenticación GSS-API determinan qué plugin debe utilizarse explorando la lista de plugins GSS-API implementados en el servidor de DB2. Se elige el primer nombre de plugin de autenticación que coincide con un plugin de autenticación GSS-API implementado en el cliente. Puede especificar la lista de plugins GSS-API de servidor implementados utilizando el parámetro de configuración del gestor de bases de datos `srvcon_gssplugin_list`. La figura siguiente muestra la infraestructura del plugin de seguridad en un cliente DB2:

Plugins de seguridad

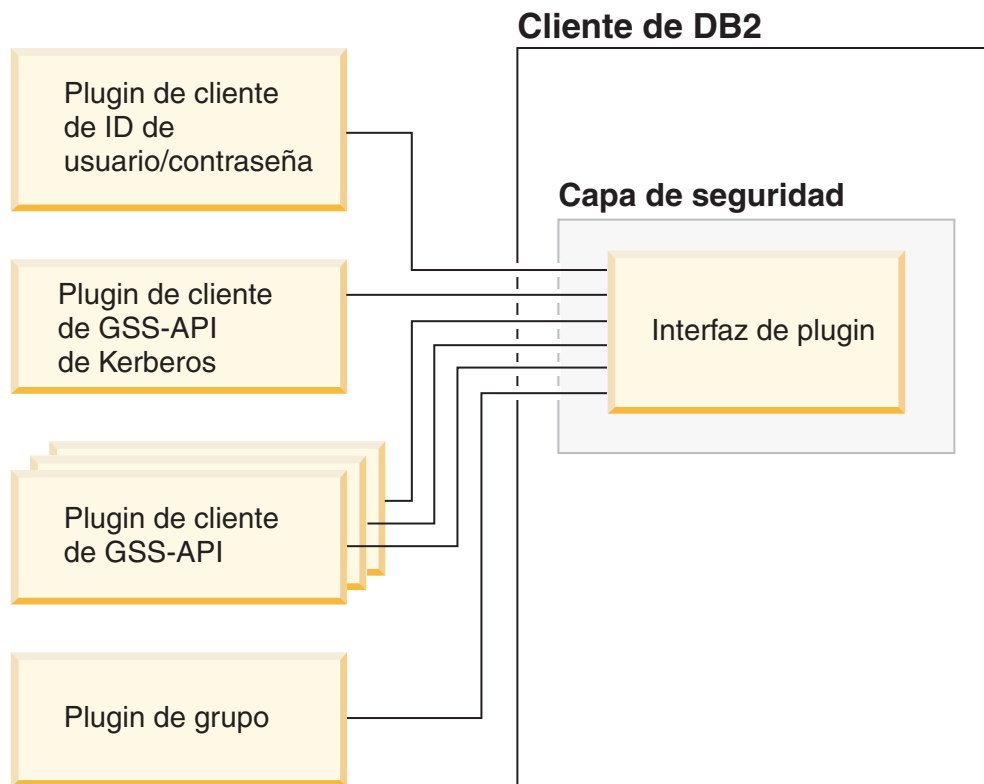


Figura 1. Despliegue de los plugins de seguridad en clientes DB2

Despliegue de plugins de seguridad en servidores DB2

Los servidores DB2 pueden soportar un plugin de recuperación de grupos, un plugin de autenticación por ID de usuario/contraseña y varios plugins de GSS-API. Puede especificar los plugins GSS-API disponibles en forma de lista de valores para el parámetro de configuración del gestor de bases de datos **srvcon_gssplugin_list**. Sin embargo, sólo un plugin GSS-API de esta lista puede ser un plugin Kerberos.

Además de desplegar los plugin de seguridad del lado del servidor en su servidor de bases de datos, es posible que deba desplegar plugins de autenticación de cliente en su servidor de bases de datos. Cuando el usuario ejecuta operaciones a nivel de instancia, como **db2start** y **db2trc**, el gestor de bases de datos DB2 comprueba la autorización para esas operaciones utilizando los plugins de autenticación del cliente. Por tanto, es posible que deba instalar el plugin de autenticación de cliente correspondiente al plugin de autenticación en el servidor. Este nombre de plugin se especifica mediante el parámetro de configuración **authentication** del gestor de bases de datos en el servidor.

Puede establecer los parámetros de configuración **authentication** y **srvcon_auth** en valores diferentes. Esta situación provoca que se utilice un mecanismo para autenticar las conexiones de base de datos y otro mecanismo para la autorización local.

El método más habitual para este enfoque consiste en:

- Establecer el parámetro de configuración **srvcon_auth** en GSSPLUGIN.
- Establecer el parámetro de configuración **authentication** en SERVER.

El parámetro de configuración **srvcon_auth** es un medio de alterar temporalmente el tipo de autenticación que utilizan las conexiones entrantes. Estas conexiones utilizan el método de autenticación que especifica el parámetro de configuración **srvcon_auth**, pero si este valor se deja en blanco, se utilizará en su lugar el valor del parámetro **authentication**.

Si no utiliza plugins de autenticación de cliente en el servidor de bases de datos, las operaciones a nivel de instancia, tales como **db2start**, fallarán.

La figura siguiente muestra la infraestructura del plugin de autenticación de seguridad en un servidor DB2:

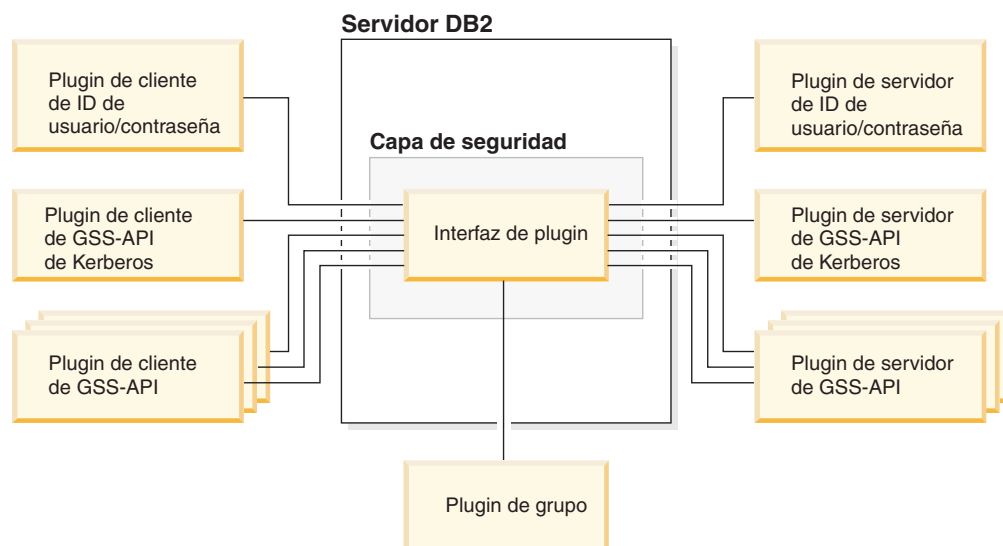


Figura 2. Despliegue de los plugins de seguridad en servidores DB2

Habilitación de los plugins de seguridad

Puede especificar los plugins que utilizará para cada mecanismo de autenticación estableciendo los parámetros de configuración del gestor de bases de datos. En la tabla siguiente se detallan estos parámetros:

Tabla 11. Parámetros de configuración del gestor de bases de datos para plugins de autenticación de seguridad

Descripción	Nombre de parámetro
Plugin de ID de usuario/contraseña de cliente	CLNT_PW_PLUGIN
Plugin Kerberos de cliente	CLNT_KRB_PLUGIN
Plugin de grupo	GROUP_PLUGIN
Plugin GSS para autorización local	LOCAL_GSSPLUGIN
Modo de plugin de servidor	SRV_PLUGIN_MODE
Lista de plugins GSS del servidor	SRVCON_GSSPLUGIN_LIST
Plugin de ID de usuario/contraseña de servidor	SRVCON_PW_PLUGIN
Autenticación de conexiones del servidor	SRVCON_AUTH
Autenticación del gestor de bases de datos	AUTHENTICATION

Plugins de seguridad

Si no establece los valores para estos parámetros, se utilizarán los plugins por omisión que proporciona el producto DB2 para la recuperación de grupos, la gestión de ID de usuario/contraseña y la autenticación Kerberos (si el parámetro **authentication** se establece en KERBEROS en el servidor). Sin embargo, no se proporciona un plugin GSS-API por omisión. Si especifica un tipo de autenticación GSSPLUGIN para el parámetro **authentication**, también debe especificar un plugin de autenticación GSS-API para el parámetro de configuración **srvcon_gssplugin_list**.

Carga de 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.

Durante las operaciones de conexión o unión, el cliente de DB2 carga un plugin apropiado para el mecanismo de seguridad que el cliente ha negociado con el servidor. Una aplicación cliente puede 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. En este caso, el programa del cliente establece una conexión inicial con el servidor A que utilizará un plugin GSS-API (G1). El servidor A envía una lista de plugins admitidos al cliente, y el plugin G1 coincidente se carga en el cliente. El programa cliente tiene otra hebra que se conecta con el servidor B, que utilizará un plugin GSS-API (G2). Se informa al cliente acerca de G2, que entonces se carga, y tanto el plugin G1 como el plugin G2 se utilizan simultáneamente en el cliente.

Las acciones que no sean operaciones de conexión o unión (como actualizar la configuración del gestor de bases de datos, iniciar y detener el gestor de bases de datos o activar o desactivar un rastreo de DB2) también requieren un mecanismo de autorización. Para tales acciones, el programa cliente de DB2 carga un plugin que especifica otro parámetro de configuración del gestor de bases de datos:

- Si establece el parámetro de configuración **authentication** en GSSPLUGIN, el gestor de bases de datos de DB2 utiliza el plugin especificado por el parámetro de configuración **local_gssplugin**.
- Si establece el parámetro de configuración **authentication** en KERBEROS, el gestor de bases de datos de DB2 utiliza el plugin especificado por el parámetro de configuración **clnt_krb_plugin**.
- En caso contrario, el gestor de bases de datos de DB2 utiliza el plugin especificado por el parámetro de configuración **clnt_pw_plugin**.

Los plugins de seguridad reciben soporte para conexiones establecidas en el servidor de bases de datos a través de protocolos de dirección IPv4 e IPv6.

Desarrollo de plugins de seguridad

Si desarrolla un plugin de autenticación de seguridad, debe implementar las funciones de autenticación estándar que utiliza el gestor de bases de datos de DB2. La funcionalidad que debe implementar para los tres tipos de plugin:

Plugin de recuperación de grupos

- Buscar y devolver la lista de grupos a los que pertenece un usuario

Plugin de autenticación por ID de usuario/contraseña

- Identificar el contexto de seguridad por omisión (sólo para un plugin de cliente)
- Validar y, opcionalmente, cambiar una contraseña
- Determinar si una determinada serie representa un usuario válido (sólo para un plugin de servidor)
- Modificar el ID de usuario o la contraseña que se proporciona en el cliente antes de enviarlos al servidor (sólo para un plugin de cliente)
- Devolver el ID de autorización de DB2 asociado a un usuario determinado

Plugin de autenticación GSS-API

- Identificar el contexto de seguridad por omisión (sólo para un plugin de cliente)
- Implementar las funciones de GSS-API necesarias
- Generar credenciales iniciales basadas en un ID de usuario y contraseña y, opcionalmente, cambiar una contraseña (sólo para un plugin de cliente)
- Crear y aceptar tíquets de seguridad
- Devolver el ID de autorización de DB2 asociado a un determinado contexto de seguridad GSS-API

Puede proporcionar un ID de usuario de hasta 255 para una sentencia de conexión que emita mediante el indicador de línea de mandatos o a través de una sentencia de SQL dinámico.

Importante: La integridad de la instalación del sistema de base de datos DB2 puede estar en peligro si los plugins de seguridad no se codifican, revisan y prueban debidamente. El producto de base de datos DB2 toma precauciones frente a muchos tipos de errores habituales, pero no puede garantizar una integridad completa si se despliegan plugins de seguridad escritos por el usuario.

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 autenticación 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 autenticación del lado del servidor en el directorio siguiente:

- UNIX de 32 bits: `$DB2PATH/security32/plugin/server`
- UNIX de 64 bits: `$DB2PATH/security64/plugin/server`

Ubicaciones de las bibliotecas de plugins de seguridad

- 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`

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`

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.

Convenios de denominación para los plugins de seguridad

Para evitar conflictos en los nombres de los plugins de seguridad, es recomendable que designe el plugin utilizando el método de autenticación 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 autenticación F00somemethod, el nombre del plugin podría tener este aspecto: F00somemethod.dll.

La longitud máxima de un nombre de plugin (sin incluir la extensión de archivo ni el sufijo ".dll") 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.

Soporte de plugin de seguridad para los ID de usuario de dos componentes

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 autenticación:

- 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 autenticación.

Tabla 12. *Plugins de seguridad de DB2*

Tipo de autenticación	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 autenticación 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 autenticación de servidor que correlaciona un ID de usuario de dos componentes con un ID de autorización de dos componentes, debe asignar los valores siguientes:

Soporte de plugin de seguridad para los ID de usuario de dos componentes

- `srvcon_pw_plugin` a `IBMOSauthserverTwoPart`
- `clnt_pw_plugin` a `IBMOSauthclientTwoPart`

Para la autenticación 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 autenticación 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 autenticación 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 especificar los 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.

Consideraciones sobre los sistemas de 32 y 64 bits para los plugins de seguridad

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 actualizar 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.

Los archivos de notificaciones de administración son un recurso útil para depurar y administrar plugins de seguridad. Para ver el archivo de registro de notificaciones de administración en UNIX, examine `sqllib/db2dump/nombre de instancia.N.nfy`. Para ver el registro 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 DB2 "nombre de api de gss" y ha devuelto el mensaje de error "mensaje de error"
```


Determinación de problemas para plugins de seguridad

- 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-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 autenticación 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 autenticación 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.

Autenticación por ID de usuario/contraseña

Autenticación que identifica el contexto de seguridad por omisión (cliente solamente), valida y opcionalmente cambia una contraseña, determina si

Las API del plugin de seguridad

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.

Autenticación GSS-API

Autenticación 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.

En la lista siguiente se muestran las definiciones para la terminología utilizada en las descripciones de las API de plugins.

Plugin

Biblioteca de carga dinámica que DB2 cargará para acceder a funciones de autenticación o de búsqueda de miembros de grupos escritas por el usuario.

Autenticación implícita

Conexión a una base de datos sin especificar un ID de usuario ni una contraseña.

Autenticación 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 especificará el valor para el parámetro de la API del plugin de seguridad.

Salida Indica que la API del plugin de seguridad especificará 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** 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)
```

API para plugins de recuperación de grupos

```
(
void *ptr,
char **errmsg,
db2int32 *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeErrorMsg)
(
char *msgtobefree
);

SQL_API_RC (SQL_API_FN * db2secPluginTerm)
(
char **errmsg,
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 a un grupo.

Si **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 Database para Linux, UNIX y Windows 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 **errmsg,
      db2int32  *errmsglen);
```

Parámetros de la API db2secDoesGroupExist

nombregrupo

Entrada. ID de autorización, escrito en mayúsculas y sin blancos de cola.

groupnamelen

Entrada. Longitud, en bytes, del valor del parámetro **groupname**.

errmsg

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.

errmsglen

Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro **errmsg**.

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

errorMsg

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 **errmsg,
  db2int32 *errormsglen);
```

Parámetros de la API db2secFreeGroupListMemory

ptr Entrada. Puntero a la memoria que se debe liberar.

errmsg

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 **errmsg**.

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 usernamespace,
  db2int32 usernamespace,
  const char *dbname,
  db2int32 dbname,
  void *token,
  db2int32 tokentype,
  db2int32 location,
  const char *authpluginname,
  db2int32 authpluginname,
  void **grouplist,
  db2int32 *numgroups,
  char **errmsg,
  db2int32 *errmsglen);
```

Parámetros de la API db2secGetGroupsForUser

idaut Entrada. El valor de este parámetro es un authid SQL, es decir, que DB2 Database para Linux, UNIX y Windows lo convierte en una serie en mayúsculas sin blancos de cola. DB2 Database para Linux, UNIX y Windows siempre proporciona 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 Database para Linux, UNIX y Windows 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**.

idusuario

Entrada. Es el ID de usuario correspondiente a **authid**. Cuando se llama a esta API en el servidor en una situación de falta de conexión, DB2 Database para Linux, UNIX y Windows no rellenará este parámetro.

API db2secGetGroupsForUser - Obtener la lista de grupos del usuario

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**.

usernamespacectype

Entrada. Es el tipo del espacio de nombres. Los valores válidos para el parámetro **usernamespacectype** (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 **usernamespacectype** 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.

símbolo

Entrada. Puntero a los datos proporcionados por el plugin de autenticación. DB2 Database para Linux, UNIX y Windows 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 autenticación 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 autenticación. Si el plugin de autenticación 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 autenticación está basado en el ID de usuario/contraseña, el tipo de datos será genérico. Los valores válidos para el 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 Database para Linux, UNIX y Windows llama a esta API en el extremo del cliente o del servidor. Los valores válidos para el parámetro **location** (definido en `db2secPlugin.h`) son:

API db2secGetGroupsForUser - Obtener la lista de grupos del usuario

- 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 autenticación 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 Database para Linux, UNIX y Windows 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 siguen al primer byte). 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 Database para Linux, UNIX y Windows llamará para liberar la memoria.

numgroups

Salida. Número de grupos contenidos en el parámetro **grouplist**.

errmsg

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 **errmsg**.

Notas de uso

En la lista siguiente se describen las situaciones en las que pueden producirse problemas si esta API devuelve una lista de grupos incompleta a DB2 Database para Linux, UNIX y Windows:

- 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.
- 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: DBADM o CREATEIN (implícito o explícito sobre el esquema jar).

API db2secGetGroupsForUser - Obtener la lista de grupos del usuario

La operación falla si los derechos indicados anteriormente se otorgan a un grupo que contiene el authid de sesión, pero no explícitamente al authid de sesión.

- Eliminar archivo jar. El authid de sesión debe tener uno de los siguientes derechos: DBADM o DROPIN (implícito o explícito sobre el esquema jar) o bien es el definidor del archivo jar. La operación falla si los derechos indicados anteriormente 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.
- Sustituir archivo jar. Equivale a eliminar el archivo jar y luego instalar el archivo jar. Son aplicables las dos situaciones descritas anteriormente.
- Cuando se emite la sentencia SET SESSION_USER. Las operaciones de DB2 posteriores 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 **errmsg,
  db2int32 *errmsglen);
```

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 aDB2 Database para Linux, UNIX y Windows la versión de las API que ha implementado el plugin. 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 bases de datos DB2. La API db2secGroupPluginInit puede llamar a la API db2secLogMessage para anotar mensajes en **db2diag** 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 los archivos de anotaciones cronológicas **db2diag**, y los dos últimos parámetros son la serie del mensaje y su longitud. Los valores válidos del primer parámetro de la API db2secLogMessage (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 aparece en los archivos de anotaciones de db2diag sólo si el valor del parámetro **level** de la API db2secLogMessage es inferior o igual al parámetro de configuración **diaglevel** del gestor de bases de

API db2secGroupPluginInit - Inicializar plugin de grupo

datos. Así, por ejemplo, si utiliza el valor DB2SEC_LOG_INFO, el texto del mensaje aparece en los archivos de anotaciones de **db2diag** sólo si el parámetro de configuración **diaglevel** del gestor de bases de datos está establecido en 4.

errmsg

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.

errmsglen

Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro **errmsg**.

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 libere de forma adecuada 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 ningún sistema operativo 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 autenticación 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** con fines de depuración o información. Esta API la proporciona el sistema de base de datos DB2, por lo que 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_USERSPACE_LENGTH],
        db2int32 *userspace1en,
        db2int32 *userspace2en,
        const char *dbname,
        db2int32 dbname1en,
        void **token,
        char **errorMsg,
        db2int32 *errormsg1en
    );
    /* Opcional */
    SQL_API_RC (SQL_API_FN * db2secRemapUserid)
    (
        char userid[DB2SEC_MAX_USERID_LENGTH],
        db2int32 *useridlen,
        char usernamespace[DB2SEC_MAX_USERSPACE_LENGTH],
        db2int32 *userspace1en,
```


Las API de los plugins de autenticación por ID de usuario/contraseña

```
db2int32 *usernamespacetype,
char password[DB2SEC_MAX_PASSWORD_LENGTH],
db2int32 *passwordlen,
char newpassword[DB2SEC_MAX_PASSWORD_LENGTH],
db2int32 *newpasswordlen,
const char *dbname,
db2int32 dbnamelen,
char **errmsg,
db2int32 *errormsglen
);

SQL_API_RC ( SQL_API_FN *db2secValidatePassword)
(
const char *userid,
db2int32 useridlen,
const char *usernamespace,
db2int32 usernamespace,
db2int32 usernamespacetype,
const char *password,
db2int32 passwordlen,
const char *newpassword,
db2int32 newpasswordlen,
const char *dbname,
db2int32 dbnamelen,
db2uint32 connection_details,
void **token,
char **errmsg,
db2int32 *errormsglen
);

SQL_API_RC ( SQL_API_FN *db2secFreeToken)
(
void **token,
char **errmsg,
db2int32 *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeErrorMsg)(char *errmsg);
SQL_API_RC (SQL_API_FN * db2secClientAuthPluginTerm)
(
char **errmsg,
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_USERSPACE_LENGTH],
db2int32 *usernamespace,
db2int32 *usernamespacetype,
const char *dbname,
db2int32 dbnamelen,
void **token,
char **errmsg,
db2int32 *errormsglen
);
```

Las API de los plugins de autenticación por ID de usuario/contraseña

```
);

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).

Las API de los plugins de autenticación por ID de usuario/contraseña

Para la biblioteca de plugins de autenticación 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** 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;
```

Las API de los plugins de autenticación por ID de usuario/contraseña

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` 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;
```

Las API de los plugins de autenticación por ID de usuario/contraseña

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 autenticación del cliente

API de inicialización, para el plugin de autenticación 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 **errmsg,
    db2int32 *errmsglen );
```

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 DB2 Database para Linux, UNIX y Windows admite actualmente.

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 autenticación GSS-API o una estructura db2secUseridPasswordClientAuthFunctions_<número_versión> (también conocida como userid_password_client_auth_functions_<número_versión>), si se utiliza la autenticación por ID de usuario/contraseña. Las estructuras db2secGssapiClientAuthFunctions_<número_versión> y db2secUseridPasswordClientAuthFunctions_<número_versión> contienen punteros que señalan hacia las API implementadas para el plugin de autenticación por GSS-API y para el plugin de autenticación por ID de usuario/contraseña. En las futuras versiones de DB2 Database para Linux, UNIX y Windows, 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 de gssapi_client_auth_functions_<número_versión> o la estructura userid_password_client_auth_functions_<número_versión> indica al gestor de bases de datos de DB2 la versión de las API que ha implementado el plugin.

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 **plugin_type** 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** con fines de depuración o

API db2secClientAuthPluginInit - Inicializar el plugin de autenticación del cliente

informativos. El primer parámetro (**level**) de la API db2secLogMessage especifica el tipo de errores de diagnóstico que se anotarán en los archivos de anotaciones cronológicas **db2diag**, y los dos últimos parámetros son la serie del mensaje y su longitud. Los valores válidos del primer parámetro de la API db2secLogMessage (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 aparecerá en los archivos de anotaciones de **db2diag** sólo si el valor del parámetro 'level' de la API db2secLogMessage es inferior o igual al parámetro de configuración **diaglevel** del gestor de bases de datos. Por ejemplo, si utiliza el valor DB2SEC_LOG_INFO, el texto del mensaje aparecerá en los archivos de anotaciones de **db2diag** sólo si el parámetro de configuración **diaglevel** 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 autenticación de cliente

Libera los recursos utilizados por el plugin de autenticación de cliente.

El gestor de bases de datos DB2 llama a esta API justo antes de descargar el plugin de autenticación de cliente. Se debe implementar de tal manera que libere de forma adecuada 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 ningún sistema operativo 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 autenticación

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 **errmsg,
  db2int32 *errormsglen);
```

Parámetros de la API db2secDoesAuthIDExist

idaut

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**.

errmsg

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 **errmsg**.

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 **errmsg,
  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.

errmsg

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 **errmsg**.

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 **errmsg,
  db2int32 *errormsglen);
```

Parámetros de la API db2secFreeToken

símbolo

Entrada. Puntero a la memoria que se debe liberar.

errmsg

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 **errmsg**.

API db2secGenerateInitialCred - Generar credenciales iniciales

La API db2secGenerateInitialCred 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. Sólo 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

idusuario

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.

contraseña

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

API db2secGenerateInitialCred - Generar credenciales iniciales

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.

dbname1en

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 Database para Linux, UNIX y Windows. 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 autenticación, momento en que se liberan estos recursos. Si la API db2secGenerateInitialCred no necesita la lista en cuestión, devolverá el valor NULL.

errmsg

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.

errmsg1en

Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro **errmsg**.

API db2secGetAuthIDs - Obtener los ID de autenticación

Devuelve un parámetro SQL authid de un usuario autenticado. Esta API se invoca durante las conexiones a base de datos para los métodos de autenticación 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 usernamespace,
    db2int32 usernamespace,
    const char *dbname,
    db2int32 dbname,
    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 *username,
    db2int32 *initsessionidtype,
    char **errmsg,
    db2int32 *errmsglen);
```

Parámetros de la API db2secGetAuthIDs

idusuario

Entrada. El usuario autenticado. No suele utilizarse para la autenticación GSS-API a menos que se haya permitido un contexto de confianza para permitir las operaciones de conmutación de usuario sin autenticación. 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.

usernamespace

Entrada. Longitud, en bytes, del valor del parámetro **usernamespace**.

usernamespace

Entrada. Valor de tipo de espacio de nombres. Actualmente, el único valor de tipo de espacio de nombres soportado es DB2SEC_NAMESPACE_SAM_COMPATIBLE (que se corresponde con un estilo de nombre del usuario como, por ejemplo, dominio\mi_nombre).

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.

dbname

Entrada. Longitud, en bytes, del valor del parámetro **dbname**. Este parámetro se establece en 0 si el parámetro **dbname** es NULL.

símbolo

Entrada o salida. Datos que el plugin puede pasar a la API db2secGetGroupsForUser. Para GSS-API, esto es un descriptor de contexto

API db2secGetAuthIDs - Obtener los ID de autenticación

(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 autenticación 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 autenticación, 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** mencionados anteriormente.

SystemAuthID

Salida. ID de autorización del sistema que se corresponde con el ID del usuario autenticado. 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**.

nombreusuario

Salida. Nombre correspondiente al usuario autenticado y al valor de authid. Este nombre sólo 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 especifica 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 autenticación 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_USERSPACE_LENGTH],
  db2int32 *userspacelen,
  db2int32 *userspacetype,
  const char *dbname,
  db2int32 dbnameLen,
  void **token,
  char **errmsg,
  db2int32 *errmsglen);
```

Parámetros de la API db2secGetDefaultLoginContext

idaut

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**.

idusuario

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 para el 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 y el 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.

API db2secGetDefaultLoginContext - Obtener contexto de conexión por omisión

usernamespace

Salida. Espacio de nombres del ID de usuario.

usernamespace1en

Salida. Longitud, en bytes, del valor del parámetro **usernamespace**. De acuerdo con la limitación de que el parámetro **usernamespace** 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 tipo de espacio de nombres. Actualmente, el único tipo de espacio de nombres soportado es DB2SEC_NAMESPACE_SAM_COMPATIBLE (que se corresponde con un estilo de nombre del usuario como, por ejemplo, dominio\mi_nombre).

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.

dbname1en

Entrada. Longitud, en bytes, del valor del parámetro **dbname**.

símbolo

Salida. Puntero a los datos asignados por el plugin para pasarlos a llamadas de autenticación 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.

errmsg

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.

errmsg1en

Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro **errmsg**.

API db2secProcessServerPrincipalName - Procesar nombre de principal de servicio devuelto desde servidor

La API db2secProcessServerPrincipalName 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 autenticación 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 **errmsg,
  db2int32 *errormsglen);
```

Parámetros de la API db2secProcessServerPrincipalName

nombre

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.

errmsg

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 **errmsg**.

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_USERSPACE_LENGTH],
  db2int32 *userspacelen,
  db2int32 *userspacetype,
  char password[DB2SEC_MAX_PASSWORD_LENGTH],
  db2int32 *passwordlen,
  char newpasswd[DB2SEC_MAX_PASSWORD_LENGTH],
  db2int32 *newpasswdlen,
  const char *dbname,
  db2int32 dbnameLen,
  char **errmsg,
  db2int32 *errmsglen);
```

Parámetros de la API db2secRemapUserid

idusuario

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**.

userspace

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 **userspace** para la autenticación de tipo CLIENT y no se tiene en cuenta para los otros tipos de autenticación.

userspacelen

Entrada o salida. Longitud, en bytes, del valor del parámetro **userspace**. De acuerdo con la limitación de que el parámetro **userspacetype** 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.

userspacetype

Entrada o salida. Valor nuevo y anterior del tipo de espacio de nombres. Actualmente, el único valor de tipo de espacio de nombres soportado es DB2SEC_NAMESPACE_SAM_COMPATIBLE (que se corresponde con un estilo de nombre del usuario como, por ejemplo, dominio\mi_nombre).

API db2secRemapUserid - Volver a correlacionar el ID de usuario y la contraseña

contraseña

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 **authentication** 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.

dbname1en

Entrada. Longitud, en bytes, del valor del parámetro **dbname**.

errmsg

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.

errmsg1en

Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro **errmsg**.

db2secServerAuthPluginInit - Inicializar el plugin de autenticación del servidor

La API `db2secServerAuthPluginInit` es la API de inicialización para el plugin de autenticación del servidor al que llama el gestor de bases de datos de DB2 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 autenticación GSS-API o para una estructura `db2secUseridPasswordServerAuthFunctions_<número_versión>` (también conocida como `userid_password_server_auth_functions_<número_versión>`), si se utiliza la autenticación por ID de usuario/contraseña. La estructura `db2secGssapiServerAuthFunctions_<número_versión>` y la estructura `db2secUseridPasswordServerAuthFunctions_<número_versión>` contienen punteros que señalan hacia las API implementadas para el plugin de autenticación por GSS-API y para el plugin de autenticación 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 de `gssapi_server_auth_functions_<número_versión>` o la estructura `userid_password_server_auth_functions_<número_versión>` indica al gestor de bases de datos de DB2 la versión de las API que ha implementado el plugin.

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.

db2secServerAuthPluginInit - Inicializar el plugin de autenticación del servidor

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 autenticación 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 autenticación. 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 `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

db2secServerAuthPluginInit - Inicializar el plugin de autenticación del servidor

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** 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 los archivos de anotaciones cronológicas **db2diag**, y los dos últimos parámetros son la serie del mensaje y su longitud. Los valores válidos del primer parámetro de la API `db2secLogMessage` (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 aparecerá en los archivos de anotaciones de **db2diag** sólo si el valor del parámetro **level** de la API `db2secLogMessage` es inferior o igual al parámetro de configuración **diaglevel** del gestor de bases de datos.

Así, por ejemplo, si utiliza el valor `DB2SEC_LOG_INFO`, el texto del mensaje aparecerá en los archivos de anotaciones de **db2diag** sólo si el parámetro de configuración **diaglevel** 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 `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 autenticación de servidor

La API db2secServerAuthPluginTerm libera los recursos utilizados por el plugin de autenticación de servidor.

El gestor de bases de datos DB2 llama a esta API justo antes de descargar el plugin de autenticación de servidor. Se debe implementar de tal manera que libere de forma adecuada 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 ningún sistema operativo Windows.

Sintaxis de la API y de las estructuras de datos

```
SQL_API_RC ( SQL_API_FN *db2secServerAuthPluginTerm)
( char      **errmsg,
  db2int32  *errormsglen);
```

Parámetros de la API db2secServerAuthPluginTerm

errmsg

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 **errmsg**.

API db2secValidatePassword - Validar contraseña

Proporciona un método para realizar la autenticación 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 **authentication** se establece en 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 casos anteriores si la autenticación 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

idusuario

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

API db2secValidatePassword - Validar contraseña

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 **usernamespace** se establece en el valor DB2SEC_USER_NAMESPACE_UNDEFINED (definido en db2secPlugin.h).

contraseña

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.

dbname1en

Entrada. Longitud, en bytes, del valor del parámetro **dbname**. Este parámetro se establece en 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.

API db2secValidatePassword - Validar contraseña

- 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 autenticación implícita es permitir la conexión sin realizar ninguna validación de contraseña. Sin embargo, los desarrolladores de plugins pueden no permitir la autenticación implícita devolviendo un error DB2SEC_PLUGIN_BADPASSWORD.

símbolo

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 las API db2secGetAuthIDs y db2secGetGroupsForUser.

errmsg

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.

errmsglen

Salida. Puntero a un valor entero que indica la longitud en bytes de la serie del mensaje de error contenida en el parámetro **errmsg**.

Las API y definiciones necesarias para los plugins de autenticación de GSS-API

En la tabla siguiente 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 13. Las API y definiciones necesarias para los plugins de autenticación de GSS-API

Tipo de API	Nombre de API	Descripción
Las API del extremo cliente	<code>gss_init_sec_context</code>	Iniciar un contexto de seguridad con una aplicación homóloga.
Las API del extremo servidor	<code>gss_accept_sec_context</code>	Aceptar un contexto de seguridad iniciado por una aplicación homóloga.
Las API del extremo servidor	<code>gss_display_name</code>	Convertir un nombre de formato interno a texto.
Las API comunes	<code>gss_delete_sec_context</code>	Suprimir un contexto de seguridad establecido.
Las API comunes	<code>gss_display_status</code>	Obtener el mensaje de error de texto asociado a un código de estado de GSS-API.
Las API comunes	<code>gss_release_buffer</code>	Suprimir un almacenamiento intermedio.
Las API comunes	<code>gss_release_cred</code>	Liberar las estructuras de datos locales asociadas a una credencial de GSS-API.
Las API comunes	<code>gss_release_name</code>	Suprimir un nombre con formato interno.
Definiciones necesarias	<code>GSS_C_DELEG_FLAG</code>	Solicita delegación.
Definiciones necesarias	<code>GSS_C_EMPTY_BUFFER</code>	Significa que <code>gss_buffer_desc</code> no contiene ningún dato.
Definiciones necesarias	<code>GSS_C_GSS_CODE</code>	Indica un código de estado principal GSS.
Definiciones necesarias	<code>GSS_C_INDEFINITE</code>	Indica que el mecanismo no es compatible con la caducidad de contexto.
Definiciones necesarias	<code>GSS_C_MECH_CODE</code>	Indica un código de estado secundario de GSS.
Definiciones necesarias	<code>GSS_C_MUTUAL_FLAG</code>	Se ha solicitado autenticación mutua.
Definiciones necesarias	<code>GSS_C_NO_BUFFER</code>	Denota que la variable <code>gss_buffer_t</code> no apunta a una estructura <code>gss_buffer_desc</code> válida.
Definiciones necesarias	<code>GSS_C_NO_CHANNEL_BINDINGS</code>	No existen vinculaciones de canales de comunicación.
Definiciones necesarias	<code>GSS_C_NO_CONTEXT</code>	Denota que la variable <code>gss_ctx_id_t</code> no apunta a un contexto válido.
Definiciones necesarias	<code>GSS_C_NO_CREDENTIAL</code>	Denota que la variable <code>gss_cred_id_t</code> no apunta a un descriptor de credencial válido.
Definiciones necesarias	<code>GSS_C_NO_NAME</code>	Denota que la variable <code>gss_name_t</code> no apunta a un nombre interno válido.
Definiciones necesarias	<code>GSS_C_NO_OID</code>	Utilizar el mecanismo de autenticación por omisión.

Las API y definiciones necesarias para los plugins de autenticación de GSS-API

Tabla 13. Las API y definiciones necesarias para los plugins de autenticación de GSS-API (continuación)

Tipo de API	Nombre de API	Descripción
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 autenticación de GSS-API

En la lista siguiente se describen las restricciones para los plugins de autenticación 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 autenticación 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 `sqlib/samples/security/plugins` y los ejemplos de plugins de GSS-API de JCC (.java) se encuentran en `sqlib/samples/java/jdbc`.

Directorio de Windows: los ejemplos de 'C' se encuentran en `sqlib\samples\security\plugins` y los ejemplos de plugins de GSS-API de JCC (.java) se encuentran en `sqlib\samples\java\jdbc`.

Tabla 14. Archivos de programas de ejemplo de plugins de seguridad

Nombre de programa de ejemplo	Descripción del programa
<code>combined.c</code>	Ejemplo de autenticación combinada de IDusuario/contraseña y búsqueda de grupo
<code>duaconfigschemaldif</code>	Se utiliza para el servidor LDAP de ITDS cuando un sistema HP-UX quiere conectar con el servidor LDAP.
<code>group_file.c</code>	Ejemplo de plugin de gestión de grupo simple basada en el archivo
<code>gssapi_simple.c</code>	Ejemplo de plugin de autenticación GSS-API básica (tanto de cliente como de servidor).
<code>IBMLDAPauthclient.c</code>	Implementa un plugin de seguridad de DB2 de cliente que interactúa con un registro de usuarios LDAP.
<code>IBMLDAPauthserver.c</code>	Implementa un plugin de seguridad de DB2 de servidor que interactúa con un registro de usuarios LDAP.
<code>IBMLDAPconfig.c</code>	Contiene funciones relacionadas con la búsqueda y el análisis del archivo de configuración para un plugin de seguridad LDAP de DB2.
<code>IBMLDAPgroups.c</code>	Implementa un plugin de seguridad de DB2 para la búsqueda de grupos basada en LDAP.
<code>IBMLDAPutils.c</code>	Contiene funciones de programa de utilidad utilizadas en el plugin de seguridad LDAP de DB2.
<code>IBMLDAPutils.h</code>	Archivo de cabecera de plugin de seguridad LDAP.
<code>JCKKerberosPlugin.java</code>	Implementa un plugin GSS-API que efectúa la autenticación de Kerberos mediante IBM DB2 Universal Driver.
<code>JCKKerberosPluginTest.java</code>	Utiliza <code>JCKKerberosPlugin</code> para obtener una conexión de DB2 mediante IBM DB2 Universal Driver.
<code>JCCSimpleGSSPlugin.java</code>	Implementa un plugin GSS-API que efectúa la comprobación de ID de usuario y de contraseña mediante IBM DB2 Universal Driver.
<code>JCCSimpleGSSContext.java</code>	Implementa un <code>GSSContext</code> que debe utilizar <code>JCCSimpleGSSPlugin</code> .
<code>JCCSimpleGSSCredential.java</code>	Implementa un <code>GSSCredential</code> que debe utilizar <code>JCCSimpleGSSPlugin</code> .
<code>JCCSimpleGSSException.java</code>	Implementa un <code>GSSException</code> que debe utilizar <code>JCCSimpleGSSPlugin</code> .
<code>JCCSimpleGSSName.java</code>	Implementa un <code>GSSName</code> que debe utilizar <code>JCCSimpleGSSPlugin</code> .
<code>JCCSimpleGSSPluginTest.java</code>	Utiliza <code>JCCSimpleGSSPlugin</code> para obtener una conexión de DB2 mediante 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 `sql1lib/samples/BARVendor` en plataformas UNIX y en el directorio `sql1lib\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 almacenamiento de proveedor de copia de seguridad y restauración.

Salida Indica que la API del plugin de almacenamiento de proveedor de copia de seguridad y restauración 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.

db2VendorGetNextObj - Obtener el objeto siguiente en el dispositivo

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.

aliasbd

Especifica el alias de la base de datos a la que pertenece el objeto.

indicación de fecha y hora

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 de 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 15. Códigos de retorno de la API db2VendorGetNextObj

Número	Código de retorno	Explicación
2	SQLUV_COMM_ERROR	Error de comunicación con dispositivo - Error.
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.

db2VendorGetNextObj - Obtener el objeto siguiente en el dispositivo

Tabla 15. Códigos de retorno de la API db2VendorGetNextObj (continuación)

Número	Código de retorno	Explicación
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 emisor 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 16. 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 *hdlc,
               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)

hdlc 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 grabación (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 estos 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

sqluvend - Desenlazar un dispositivo de proveedor y liberar sus recursos

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 **action** = SQLUV_ABORT. Para una operación de copia de seguridad, las sesiones confirmadas reciben una secuencia de llamadas sqluvint, sqluvdel y sqluvend.

Abort El sistema de bases de datos DB2 ha encontrado un problema y no se leerán ni grabarán más datos en la sesión.

Para una sesión de grabación (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 **action** = 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 **action** = 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 **action** = SQLUV_ABORT, se debería haber producido un error inicial muy grave.

Códigos de retorno

Tabla 17. 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.
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 * hdlc,
              struct Data *data,
              struct Return_code *return_code);
```

Parámetros de la API sqluvget

hdlc Entrada. Puntero al espacio asignado para la estructura Data (incluido el almacenamiento intermedio de datos) y el código de retorno.

datos 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 18. 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 (vea la nota siguiente)	La sesión terminará.
SQLUV_INV_ACTION	Se ha solicitado una acción no válida.	sqluvend, action = SQLU_ABORT (vea la nota siguiente)	La sesión terminará.
SQLUV_INV_DEV_HANDLE	Handle de dispositivo no válido.	sqluvend, action = SQLU_ABORT (vea la nota siguiente)	La sesión terminará.
SQLUV_INV_BUFF_SIZE	Se ha especificado un tamaño de almacenamiento intermedio no válido.	sqluvend, action = SQLU_ABORT (vea la nota siguiente)	La sesión terminará.
SQLUV_DEV_ERROR	Error de dispositivo.	sqluvend, action = SQLU_ABORT (vea la nota siguiente)	La sesión terminará.

sqluvget - Leer datos de un dispositivo de proveedor

Tabla 18. Códigos de retorno válidos para sqluvget y acción resultante de DB2 (continuación)

Literal en archivo de cabecera	Descripción	Próxima llamada probable	Otros comentarios
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, mediante este código de retorno pueden indicarse condiciones de dispositivo no preparado.	sqluvget o sqluvend, action = SQLU_ABORT	
SQLUV_LINK_NOT_EXIST	No existe ningún enlace actualmente	sqluvend, action = SQLU_ABORT (vea la nota siguiente)	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 (vea la nota siguiente)	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

sqluvint - Inicializar y enlazar con un dispositivo de proveedor

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 grabando o la restauración que se debe leer a través de campos de la estructura DB2_info. En el caso de **action** = 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 19. 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 API (a datos BACKUP). Si action = SQLUV_READ, verifique la existencia del objeto especificado 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.

sqluvint - Inicializar y enlazar con un dispositivo de proveedor

Tabla 19. 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_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.
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 el parámetro action 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 el parámetro action de la llamada de sqluvint es "R" (lectura) y más de un objeto coincide con los criterios especificados de 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.

sqluvint - Inicializar y enlazar con un dispositivo de proveedor

Tabla 19. 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_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.
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 especificado antes de devolver SQLUV_MAX_LINK_GRANT; la siguiente llamada será 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 - Grabar 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 *hdlc,
              struct Data *data,
              struct Return_code *return_code);
```

Parámetros de la API sqluvput

hdlc Entrada. Puntero al espacio asignado para la estructura DATA (incluido el almacenamiento intermedio de datos) y el código de retorno.

datos Salida. Almacenamiento de datos llenado con datos que se deben grabar.

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 20. 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 (vea la nota siguiente).	La sesión terminará.
SQLUV_INV_ACTION	Se ha solicitado una acción no válida.	sqluvend, action = SQLU_ABORT (vea la nota siguiente).	La sesión terminará.
SQLUV_INV_DEV_HANDLE	Handle de dispositivo no válido.	sqluvend, action = SQLU_ABORT (vea la nota siguiente).	La sesión terminará.
SQLUV_INV_BUFF_SIZE	Se ha especificado un tamaño de almacenamiento intermedio no válido.	sqluvend, action = SQLU_ABORT (vea la nota siguiente).	La sesión terminará.

sqluvpt - Grabar datos en un dispositivo de proveedor

Tabla 20. Códigos de retorno válidos para sqluvput y acción resultante de DB2 (continuación)

Literal en archivo de cabecera	Descripción	Próxima llamada probable	Otros comentarios
SQLUV_ENDOFMEDIA	Se ha llegado al final del soporte de almacenamiento; por ejemplo, fin de cinta.	sqluvend	
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 (vea la nota siguiente).	La sesión terminará.
SQLUV_WARNING	Aviso. No se debe utilizar para indicar el fin de soporte de almacenamiento 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 (vea la nota siguiente).	La sesión terminará.
SQLUV_IO_ERROR	Error de E/S.	sqluvend, action = SQLU_ABORT (vea la nota siguiente).	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 21. 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.
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.

Tabla 21. Campos de la estructura DB2_info (continuación)

Nombre de campo	Tipo de datos	Descripción
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 posibles son los siguientes: si action es SQLUV_WRITE: 0 - copia de seguridad de base de datos completa 3 - copia de seguridad a nivel de espacio de tablas. Si action 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
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.
indicación de fecha y hora	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.
secuencia	char	Especifica la extensión de archivo para la imagen de copia de seguridad. Para operaciones de grabación, 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 las 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.

Tabla 21. Campos de la estructura DB2_info (continuación)

Nombre de campo	Tipo de datos	Descripción
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.
contraseña	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.
dbPartition	db2NoteType	Número de partición de base de datos. La interfaz de proveedor da soporte a los números del rango {0 - 999}.

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;  
db2NodeType dbPartition;  
} 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 22. 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.
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.

DB2 utiliza esta estructura de datos para enviar información al plugin de copia de seguridad y restauración del almacenamiento de proveedor mediante las API `sqluvint` y `sqluvdel`.

Tabla 23. Campos de la estructura `Init_input`.

Nombre de campo	Tipo de datos	Descripción
<code>DB2_session</code>	<code>struct DB2_info</code>	Descripción de la sesión desde la perspectiva de DB2.
<code>size_options</code>	<code>unsigned short</code>	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 .
<code>size_HI_order</code>	<code>sqluint32</code>	32 bits más a la izquierda del cálculo de tamaño de la base de datos en bytes; el tamaño total es 64 bits.
<code>size_LOW_order</code>	<code>sqluint32</code>	32 bits más a la derecha del cálculo de tamaño de la base de datos en bytes; el tamaño total es 64 bits.
<code>options</code>	<code>void</code>	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; esto significa que no recibe soporte 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 .
<code>reserve</code>	<code>void</code>	Reservado para una utilización futura.

Tabla 23. 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

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.

Las API sqluvint y sqluvdel utilizan esta estructura de datos.

Tabla 24. 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 `sqlvput` al grabar datos en el dispositivo de proveedor y por la API `sqlvget` al leer datos del dispositivo de proveedor.

Tabla 25. Campos de la estructura `Data`

Nombre de campo	Tipo de datos	Descripción
<code>obj_num</code>	<code>sqlint32</code>	Número de secuencia asignado por DB2 durante una operación de copia de seguridad.
<code>buff_size</code>	<code>sqlint32</code>	Tamaño del almacenamiento intermedio.
<code>actual_buff_size</code>	<code>sqlint32</code>	Número real de bytes enviados o recibidos. No debe sobrepasar el valor de <code>buff_size</code> .
<code>dataptr</code>	<code>void</code>	Puntero al almacenamiento intermedio de datos. DB2 asigna espacio para el almacenamiento intermedio.
<code>reserve</code>	<code>void</code>	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 26. Campos de la estructura Return_code

Nombre de campo	Tipo de datos	Descripción
return_code (ver nota siguiente)	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 interactuar 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 los productos de la base de datos 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

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 parámetro a las API `InitCompression` e `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.

aliasbd

Alias de 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.

indicación de fecha y hora

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 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 include).

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 validar 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 comprime los datos solamente en incrementos de bytes, la biblioteca establece este campo en 0. Consulte la descripción del parámetro `srcGran` de la API `Compress API` 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

COMPR_PIINFO

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 de 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

Compresión de un bloque de datos.

El parámetro **src** apunta a un bloque de datos cuyo volumen es de **srcLen** bytes. El parámetro **tgt** apunta a un almacenamiento intermedio cuyo volumen es de **tgtSize** bytes. La biblioteca de conectores comprime los datos en la dirección **src** y graba los datos comprimidos en el almacenamiento intermedio en la dirección **tgt**. La cantidad real de datos descomprimidos que estaban comprimidos se almacena en **srcAct**. El volumen real de los datos comprimidos se devuelve como **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 ha devuelto un valor de 0 para `piInfo->srcGran`, la granularidad es de 1 byte.

tgt

Entrada y salida. Almacenamiento intermedio de destino para datos

Compress - Comprimir un bloque de datos

comprimidos. DB2 suministrará este almacenamiento intermedio de destino y el plugin comprimirá los datos situados en **src** y grabará 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 de CRC del bloque descomprimido 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** apunta a un bloque de datos cuyo volumen es de **srcLen** bytes. El parámetro **tgt** apunta a un almacenamiento intermedio cuyo volumen es de **tgtSize** bytes. La biblioteca de conectores descomprime los datos en la dirección **src** y graba los datos descomprimidos en el almacenamiento intermedio en la dirección **tgt**. El volumen real de los datos descomprimidos se devuelve como **tgtLen**. Si la biblioteca ha devuelto un valor para 1 para **piInfo->useCRC**, el valor CRC del bloque descomprimido se devuelve como **tgtCRC**. Si la biblioteca ha devuelto un valor de 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 grabará 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

Decompress - Descomprimir un bloque de datos

de CRC del bloque descomprimido 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 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 como **blockSize**. La biblioteca de plugins graba 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**.

datos

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 proporcionará los parámetros adecuados de piInfo, y asignará pCB y devolverá 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 asignará pCB y devolverá 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 liberará 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 liberará 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 compresión).

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 10. Estructuras de datos que las API utilizan

db2DistMapStruct

La API db2GetDistMap utiliza esta estructura.

Tabla 27. Campos de la estructura db2DistMapStruct

Nombre de campo	Tipo de datos	Descripción
nombret	unsigned char	Nombre de la tabla totalmente calificado.
partinfo	db2PartitioningInfo	Esta estructura se utiliza para almacenar información sobre particionamiento, tal como la correlación de distribución y la clave de distribución de una tabla.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE db2DistMapStruct
{
    unsigned char          *tname;    /* Nombre de tabla      */
                                /* completo             */
    struct db2PartitioningInfo *partinfo; /* Información de
                                /* particionamiento     */
};
```

db2HistoryData

Esta estructura se utiliza para devolver información después de una llamada a la API `db2HistoryGetEntry`.

Nota: La API `db2HistoryGetEntry` solamente recibe soporte en los lenguajes de programación C, C++ o Java. En los lenguajes de programación COBOL, FORTRAN y REXX ya no recibe soporte. Puede emitir una consulta para acceder a los registros históricos de la base de datos utilizando la vista administrativa `DB_HISTORY`.

Tabla 28. 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 <code>SQLUHINF</code> . No existe ninguna definición simbólica para esta serie.
oObjectPart	db2Char	Los 14 primeros caracteres son una indicación de fecha y hora con el formato <code>aaaammddhhmmss</code> , 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 <code>aaaammddhhmmss</code> , que indica cuándo se ha completado la operación.
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 : <ul style="list-style-type: none"> • 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	db2LSN	Último número de secuencia de anotaciones cronológicas.

db2HistoryData

Tabla 28. Campos de la estructura db2HistoryData (continuación)

Nombre de campo	Tipo de datos	Descripción
oEID	Estructura	Identificador de entrada exclusivo.
poEventSQLCA	Estructura	sqlca resultante del suceso 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.
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.
ioLogRange	db2HistoryLogRange	Rangos de anotaciones cronológicas para cada corriente de anotaciones cronológicas.
oOperation	char	Consulte el apartado Tabla 29.
oObject	char	Granularidad de la operación: D para base de datos completa, P para espacio de tablas y T para tabla.
oOtype	char	Consulte el apartado Tabla 30 en la página 637.
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 29. 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	renombrar 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

Tabla 29. Valores de **o0peration** válidos en la estructura *db2HistoryData* (continuación)

Valor	Descripción	Definición C	Definición COBOL/FORTRAN
T	alterar espacio de tablas	DB2HISTORY_OP_ALT_ TABLESPACE	DB2HIST_OP_ALT_TBS
U	descargar	DB2HISTORY_OP_UNLOAD	DB2HIST_OP_UNLOAD

Tabla 30. Valores de **o0ptype** válidos en la estructura *db2HistData*

o0peration	o0ptype	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	fuera de línea, en línea, fuera de línea incremental, en línea incremental	DB2HISTORY_OPTYPE_OFFLINE, DB2HISTORY_OPTYPE_ONLINE, DB2HISTORY_OPTYPE_INCR_OFFLINE, DB2HISTORY_OPTYPE_INCR_ONLINE
T	C R	añadir contenedores, reequilibrar	DB2HISTORY_OPTYPE_ADD_CONT, DB2HISTORY_OPTYPE_REB

Tabla 31. 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 de los registros de histórico de base de datos local.

Esta estructura se utiliza en la estructura *db2HistoryData*

Tabla 32. Campos en la estructura *db2HistoryLogRange*

Nombre de campo	Tipo de datos	Descripción
iNumLogStreams	DB2UINT32	Número asignado de corrientes de anotaciones cronológicas en oStream .

Tabla 32. Campos en la estructura db2HistoryLogRange (continuación)

Nombre de campo	Tipo de datos	Descripción
oNumLogStreams	DB2UINT32	Número de corrientes de anotaciones cronológicas devueltas en oStream .
oStream	db2HistoryLogStreamRange	Lista de rangos de anotaciones cronológicas (por corrientes de anotaciones cronológicas).

Esta estructura se utiliza en la estructura db2HistoryLogRange

Tabla 33. Campos de la estructura db2HistoryLogStreamRange

Nombre de campo	Tipo de datos	Descripción
oStreamID	db2LogStreamIDType	ID de corriente de anotaciones cronológicas del rango de anotaciones cronológicas
oFirstLog	DB2UINT32	La primera extensión del archivo de anotaciones cronológicas. <ul style="list-style-type: none"> • 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 33. Campos de la estructura db2HistoryLogStreamRange (continuación)

Nombre de campo	Tipo de datos	Descripción
oLastLog	DB2UINT32	<p>La última extensión del archivo de anotaciones cronológicas.</p> <ul style="list-style-type: none"> • 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 en el nivel del espacio de tablas o la base de datos completa que era actual cuando se terminó la carga.

Sintaxis de la API y de las estructuras de datos

```
typedef SQL_STRUCTURE db2HistoryData
{
    char ioHistDataID[8];
    db2Char oObjectPart;
    db2Char oEndTime;
    db2Char oID;
    db2Char oTableQualifier;
    db2Char oTableName;
    db2Char oLocation;
    db2Char oComment;
    db2Char oCommandText;
    db2LSN oLastLSN;
    db2HistoryEID oEID;
    struct sqlca *poEventsSQLCA;
    struct db2Char *poTablespace;
    db2UInt32 iNumTablespaces;
    db2UInt32 oNumTablespaces;
    db2HistoryLogRange ioLogRange;
    char oOperation;
    char oObject;
    char oOtype;
    char oStatus;
    char oDeviceType;
    db2HistoryData;
}
typedef SQL_STRUCTURE db2HistoryLogRange
{
    db2UInt32 iNumLogStreams;
    db2UInt32 oNumLogStreams;
    struct db2HistoryLogStreamRange *oStream;
    db2HistoryLogRange;
}
typedef SQL_STRUCTURE db2HistoryLogStreamRange
{
    db2LogStreamIDType oStreamID;
    db2UInt32 oFirstLog;
```

db2HistoryData

```
    db2UInt32 oLastLog;
    db2HistoryLogStreamRange;
}

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 de los registros de histórico de la base de datos local.

Parámetros de la estructura de datos db2HistoryLogRange

iNumLogStreams

Número asignado de db2HistoryLogStreamRange en **oStream**

oNumLogStreams

Número de corrientes de anotaciones cronológicas devueltas en **oStream**.

oStream

Lista de rangos de anotaciones cronológicas (por corrientes de anotaciones cronológicas).

Estructura de datos db2LSN

Esta unión, que las API db2ReadLog y db2ReadLogNoConn utilizan, contiene la definición del número de secuencia de anotaciones cronológicas.

Un número de secuencia de anotaciones cronológicas (LSN) representa una dirección de byte relativa dentro de la anotación cronológica de base de datos. Todos los registros de anotaciones cronológicas se identifican mediante este número. Un LSN representa el desplazamiento en bytes del registro de anotaciones cronológicas con respecto al principio del registro de anotaciones cronológicas de la base de datos.

Tabla 34. Campos de la estructura db2LSN

Nombre de campo	Tipo de datos	Descripción
lsnU64	db2UInt64	Especifica el número de secuencia de anotaciones cronológicas.

Sintaxis de la API y de las estructuras de datos

```
typedef SQL_STRUCTURE db2LSN
{
    db2UInt64    lsnU64;    /* Número de secuencia
                           de anotaciones cronológicas */
} db2LSN;
```

sql_dir_entry

Esta estructura es utilizada por las API del directorio de DCS.

Tabla 35. Campos de la estructura SQL-DIR-ENTRY

Nombre de campo	Tipo de datos	Descripción
STRUCT_ID RELEASE CODEPAGE COMMENT LDB TDB AR PARAM	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 rellenos 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 36. 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 .
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. Hace referencia al número de página de la primera extensión libre que sigue a 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 recientemente añadido, pero estas nuevas páginas no se reflejarán en el número de páginas libres hasta que se complete el reequilibrado. Cuando no tiene lugar un reequilibrado de espacio de tablas, el número de páginas utilizadas más el número de páginas libres será 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 37. 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.
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** (definidos 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 38. 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	INTEGER (64-BIT)	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 38. 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 (definidos 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 normal.

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.

SQLB_TBSPQRY_DATA

Los valores posibles para TBSSTATE (definidos 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

La definición de almacenamiento se han cambiado antes de un 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 (definidos 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;
    sqluint64 lifeLSN;
    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.cb1
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.
```

SQLB_TBSPQRY_DATA

```
05 SQL-TBS-STATE          PIC 9(9) COMP-5.
05 SQL-LIFE-LSN           PIC 9(18) COMP-5.
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 39. 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. La siguiente salida es un ejemplo en el que se 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 del 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 de una SQLDA depende de su utilización. En 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 del lenguaje principal.

Sintaxis del lenguaje

Estructura en C

```

/* Archivo: sqlda.h */
/* Estructura: SQLDA */
/* ... */
SQL_STRUCTURE sqlda
{
    _SQLOLDCHAR    sqldaid[8];
    Tong          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
{

```

SQLDA

```
    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 40. 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 DCOLNUM sqldcoln.

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 41. 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 42. Campos de la estructura **SQLLOCTAB**

Nombre de campo	Tipo de datos	Descripción
LOCPAIR	Matriz	Matriz de estructuras sqllocpair .

Tabla 43. 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 44. 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 el apartado siguiente para ver 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 <code>MANAGED BY AUTOMATIC STORAGE</code> de la sentencia <code>CREATE TABLESPACE</code> o sin especificar la cláusula <code>MANAGED BY CLAUSE</code>). Para estos espacios de tablas, 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 los contenedores automáticamente en función de las vías de acceso de almacenamiento asociadas con 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 <code>db2nodes.cfg</code> , y solo se utiliza si el campo tblspace_type se ha establecido en <code>SQLE_TABLESPACES_LIKE_NODE</code> .

Los valores válidos para **TBLSPACE_TYPE** (definidos en `sqlenv`) son:

SQLE_TABLESPACES_NONE

No crear ningún espacio de tablas temporal 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. Sin embargo, 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 45. 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 suficientemente grande para contener la longitud máxima para el tipo. Una longitud 0 indica un valor nulo.

Tabla 45. Campos de la estructura SQLE-CLIENT-INFO (continuación)

Nombre de campo	Tipo de datos	Descripción
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.

En la tabla siguiente se muestran las entradas válidas para el elemento SQLE-CLIENT-INFO **TYPE** y las descripciones correspondientes a cada entrada:

Tabla 46. 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;
};
```

sqle_client_info

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 sqlegetc.

Tabla 47. 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 SQLE-CONN-SETTING **TYPE** y las descripciones correspondientes a cada entrada aparecen listadas en la tabla siguiente (se definen en sqlenv y sql):

Tabla 48. 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 de SQL CONNECT. Se deberá utilizar la sentencia SET CONNECTION de SQL para conmutar la conexión actual por una conexión latente.
SQL_DISCONNECT	SQL_DISCONNECT_EXPL SQL_DISCONNECT_COND SQL_DISCONNECT_AUTO	Elimina las conexiones que la sentencia de 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 de SQL RELEASE ha marcado explícitamente para liberar. Interrumpe todas las conexiones durante la confirmación.

sqle_conn_setting

Tabla 48. Valores de conexión (continuación)

Tipo	Valor	Descripción
SQL_SYNCPOINT	SQL_SYNC_TWOPHASE SQL_SYNC_ONEPHASE SQL_SYNC_NONE	Requiere un gestor de transacciones (TM) para coordinar las confirmaciones de dos fases entre las bases de datos que soportan este protocolo. Utiliza una confirmación de una fase para confirmar el trabajo efectuado por cada base de datos en múltiples transacciones de bases de datos. Impone un comportamiento de actualizador único y lectura múltiple. Utiliza confirmaciones de una fase para confirmar el trabajo realizado, pero no impone el comportamiento de actualizador único y lectura múltiple.
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 48. 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 49. 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 rellenos 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 50. 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 rellenos 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.

Tabla 51. 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.
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 rellenos con blancos hasta completar la longitud del campo.

Los valores válidos para **PROTOCOL** (definido en sqleenv) 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: sqleenv.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, establezca el tipo de **PROTOCOL** de la estructura del directorio de nodos en SQL_PROTOCOL_TCPIP, SQL_PROTOCOL_TCPIP4 o SQL_PROTOCOL_TCPIP6 en la estructura SQLE-NODE-STRUCT antes de llamar a la API sqlectnd. Para catalogar un nodo SOCKS de TCP/IP o TCP/IPv4, establezca el tipo de **PROTOCOL** de la estructura del directorio de nodos en SQL_PROTOCOL_SOCKS o SQL_PROTOCOL_SOCKS4 en la estructura SQLE-NODE-STRUCT antes de llamar a la API sqlectnd. SOCKS no está soportado en IPv6. Por ejemplo, no se admite SQL_PROTOCOL_SOCKS con una dirección IPv6.

Tabla 52. 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 rellenos 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 53. 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 la fuente del orden de clasificación de la base de datos. 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 <i>n</i> de este campo contiene el peso de clasificación del elemento de código cuya representación decimal subyacente es el carácter <i>n</i> 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 (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.

Tabla 53. Campos de la estructura SQLEDBDESC (continuación)

Nombre de campo	Tipo de datos	Descripción
SQLDBCMT	CHAR(30)	El comentario de la base de datos.
SQLDBSGP	INTEGER	Campo reservado. Ya no se utiliza.
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 .

Tabla 53. Campos de la estructura SQLEDBDESC (continuación)

Nombre de campo	Tipo de datos	Descripción
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 54. 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 SQLC_DBTDESC_1 (definido en sqlenv). El contenido de este campo se valida para el control de versiones.
SQLXTNT	INTEGER	Tamaño de extensión del espacio de tablas, en páginas de 4 KB. Si se especifica un valor de -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 de -1, este campo tomará por omisión el valor actual del parámetro de configuración dft_prefetch_sz .

Tabla 54. Campos de la estructura SQLETSDESC (continuación)

Nombre de campo	Tipo de datos	Descripción
SQLFSCACHING	UNSIGNED CHAR	Almacenamiento en antememoria del sistema de archivos. Si se especifica un valor 1, el almacenamiento en antememoria del sistema de archivos estará desactivado (OFF) para el espacio de tablas actual. Si se especifica un valor 0, el almacenamiento en antememoria del sistema de archivos estará activado (ON) para el espacio de tablas actual. Especifique 2 para indicar el valor por omisión. En este caso, el almacenamiento en antememoria del sistema de archivos estará desactivado (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. El almacenamiento en antememoria del sistema de archivos estará activado (ON) para todas las demás plataformas.
SQLPOVHD	DOUBLE	Uso 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 versiones futuras.
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 versiones futuras.
SQLTSTYP	CHAR(1)	Indica si el espacio de tablas es gestionado por el sistema o gestionado por la base de datos. .

Tabla 54. Campos de la estructura SQLETSDESC (continuación)

Nombre de campo	Tipo de datos	Descripción
SQLCCNT	SMALLINT	Número de contenedores asignados al espacio de tablas. Indica cuántos valores SQLCTYPE/SQLCSIZE/SQLCLEN/SQLCONTR siguen.
CONTAINR	Matriz	Matriz de estructuras sqlccnt SQLETSDESC.

Tabla 55. Campos de la estructura SQLETSCDESC

Nombre de campo	Tipo de datos	Descripción
SQLCTYPE	CHAR(1)	Identifica el tipo de este contenedor. .
SQLCSIZE	INTEGER	Tamaño del contenedor identificado en SQLCONTR , especificado en páginas de 4 KB. Solo 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 especificar **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. Es el valor por omisión para las 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

Secuencia de ordenación CESU-8 (Esquema de codificación de compatibilidad para UTF-16: 8 Bits) tal como se especifica en el documento Unicode Technical Report #26, 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 información detallada sobre UCA en el documento Unicode Technical Standard #10, disponible en el sitio web de Unicode Consortium (www.unicode.org). Este valor sólo se puede especificar al crear una base de datos Unicode.

Importante: Las clasificaciones basadas en el Algoritmo de clasificación Unicode del estándar Unicode versión 4.0.0 han quedado en desuso en la Versión 10.1 y podrían eliminarse en un release futuro. Para obtener más información, consulte el tema “Las clasificaciones basadas en el Algoritmo de clasificación Unicode del estándar Unicode versión 4.0.0 han quedado en desuso” en *Novedades en DB2 Versión 10.1*

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 información detallada sobre UCA en el documento Unicode Technical Standard #10, que está disponible en el sitio web de Unicode Consortium (www.unicode.org). Este valor sólo se puede especificar al crear una base de datos Unicode.

Importante: Las clasificaciones basadas en el Algoritmo de clasificación Unicode del estándar Unicode versión 4.0.0 han quedado en desuso en la Versión 10.1 y podrían eliminarse en un release futuro. Para obtener más información, consulte el tema “Las clasificaciones basadas en el Algoritmo de clasificación Unicode del estándar Unicode versión 4.0.0 han quedado en desuso” en *Novedades en DB2 Versión 10.1*

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 información detallada sobre UCA en el documento Unicode Technical Standard #10, disponible en el sitio web de Unicode Consortium (www.unicode.org). Este valor sólo se puede especificar al crear una base de datos Unicode.

Importante: Las clasificaciones basadas en el Algoritmo de clasificación Unicode del estándar Unicode versión 4.0.0 han quedado en desuso en la Versión 10.1 y podrían eliminarse en un release futuro. Para obtener más información, consulte el tema “Las clasificaciones basadas en el Algoritmo de clasificación Unicode del estándar Unicode versión 4.0.0 han quedado en desuso” en *Novedades en DB2 Versión 10.1*

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. Solo es válido cuando **SQLTSTYP** = **SQL_TBS_TYP_DMS**.

SQL_TBSC_TYP_FILE

Archivo. Solo es válido cuando **SQLTSTYP** = **SQL_TBS_TYP_DMS**.

SQL_TBSC_TYP_PATH

Vía de acceso (directorio). Solo es válido cuando **SQLTSTYP** = **SQL_TBS_TYP_SMS**.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqlbdbdesc
{
    _SQLOLDCHAR sqldbdid[8];
    sqlint32 sqldbccp;
    sqlint32 sqldbcsc;
    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 SQLETSDESC containr[1];
};

SQL_STRUCTURE SQLETSDESC
{
    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 SQLETSDESC

- 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 SOLTSEXT          PIC S9(9) COMP-5.
   05 SQLCATTS          USAGE IS POINTER.
   05 SQLUSRTS          USAGE IS POINTER.
   05 SQLTMPPTS          USAGE IS POINTER.
*

```

```

* Archivo: sqletsd.cbl
01 SQLETSDESC.
   05 SOLTSDID          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 SOLTSTYP          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 SQLCFSIZE        PIC S9(9) COMP-5.

```

sqledbdesc

```
      10 SQLCLEN          PIC S9(4) COMP-5.
      10 SQLCONTR        PIC X(256).
      10 SQL-PAD2        PIC X(2).
*
* Archivo: sqlenv.cbl
01 SQLETSDESC.
   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).
*
```

sqlledbdescxt

La estructura de bloque de descripción de base de datos ampliado (sqlledbdescxt) se emplea durante una llamada a la API sqllecrea para especificar valores permanentes para los atributos de la base de datos.

El bloque de descripción de base de datos ampliado crea el grupo de almacenamiento por omisión, 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 indicado. Esta estructura se utiliza además de la estructura de bloque de descripción de base de datos (sqlledbdesc), no en lugar de ella.

Si esta estructura no se pasa a la API sqllecrea, se utiliza el siguiente comportamiento:

- Se crea el grupo de almacenamiento por omisión, IBMSTOGROUP.
- 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

Nota: Aunque puede crear una base de datos especificando la cláusula AUTOMATIC STORAGE NO, la cláusula AUTOMATIC STORAGE está en desuso y es posible que se elimine en un release futuro.

Sintaxis de la API y de las estructuras de datos

```
SQL_STRUCTURE sqlledbdescxt
{
    sqluint32 sqlPageSize;
    struct sqlAutoStorageCfg *sqlAutoStorage;
    struct SLETSDESCEXT *sqlcattsext;
    struct SLETSDESCEXT *sqlusrsext;
    struct SLETSDESCEXT *sqltmptsext;
    void *reserved;
};

SQL_STRUCTURE sqlAutoStorageCfg
{
    char sqlEnableAutoStorage;
    char pad[3];
    sqluint32 sqlNumStoragePaths;
    char **sqlStoragePaths;
};

SQL_STRUCTURE SLETSDESCEXT
{
    sqlint64 sqlInitSize;
    sqlint64 sqlIncreaseSize;
    sqlint64 sqlMaximumSize;
    char sqlAutoResize;
    char sqlInitSizeUnit;
    char sqlIncreaseSizeUnit;
    char sqlMaximumSizeUnit;
};

SQL_STRUCTURE sqlledboptions
{
    void *piAutoConfigInterface;
    sqlint32 restrictive;
    void *reserved;
};
```

Parámetros de la estructura de datos sqledbdescext

Tabla 56. 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 crea el grupo de almacenamiento por omisión y se presupone una vía de acceso de almacenamiento simple con un valor determinado por el parámetro dbpath pasado, o bien el parámetro de configuración del gestor de base 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 usuario, 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).
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 para **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 sqlebdbdescxt, y especifica las vías de acceso de almacenamiento para el grupo de almacenamiento por omisión.

Tabla 57. 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).
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 mediante SQLNUMSTORAGEPATHS . El valor de SQLSTORAGEPATHS debe ser NULL si no se proporcionan vías de acceso de almacenamiento (en cuyo caso el valor de SQLNUMSTORAGEPATHS deber ser 0). La longitud máxima de una vía de acceso es de 175 caracteres.

Los valores válidos para **SQLENABLEAUTOSTORAGE** (definido en sqleenv) son:

SQL_AUTOMATIC_STORAGE_NO

No se crea el grupo de almacenamiento por omisión y los espacios de tablas gestionados por el almacenamiento automático no se pueden crear. Cuando se utiliza este valor, **SQLNUMSTORAGEPATHS** debe establecerse en 0 y **SQLSTORAGEPATHS** debe establecerse en NULL.

SQL_AUTOMATIC_STORAGE_YES

Se crea el grupo de almacenamiento por omisión, IBMSTOGROUP. 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 grupo de almacenamiento por omisión se crea o no. Actualmente, la elección se realiza en función del valor del puntero **SQLSTORAGEPATHS**. Si el valor de este puntero NULL, no se crea el grupo de almacenamiento por omisión; de lo contrario, se crea. 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 58. 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 58. 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 espacio 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 para **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 actualizaciones futuras de esta base de datos. Es decir, cuando una base de datos se actualiza a un release ulterior de DB2, **UPGRADE DATABASE** comprueba el valor del parámetro de configuración de base de datos **restrict_access** 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 incorporado en el nuevo release de DB2.

Los valores válidos (definidos en el archivo de cabecera `sqlenv`, ubicado en el directorio `include`) 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 IMPLICIT_SCHEMA
- 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 59. 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 sqldbcharset[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 sqlengne.

Tabla 60. Campos de la estructura SQLENINFO

Nombre de campo	Tipo de datos	Descripción
NODENAME	CHAR(8)	Nombre del nodo donde reside la base de datos (solamente es válido en el directorio del sistema)
LOCAL_LU	CHAR(8)	Unidad lógica local.
PARTNER_LU	CHAR(8)	Unidad lógica asociada.
MODE	CHAR(8)	Modalidad servicio 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	Adaptador de red local.
NETWORKID	CHAR(8)	ID de red.
PROTOCOL	CHAR(1)	Protocolo de comunicaciones.
SYM_DEST_NAME	CHAR(8)	Nombre de destino simbólico.
SECURITY_TYPE	SMALLINT	Tipo de seguridad.
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.

Tabla 60. Campos de la estructura SQLENINFO (continuación)

Nombre de campo	Tipo de datos	Descripción
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 61. 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.
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 símbolo 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 símbolo SQLFUPD se indican en la lista siguiente:

Tabla 62. Parámetros actualizables de configuración de base de datos

Nombre de parámetro	Símbolo	Valor de símbolo	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

Tabla 62. Parámetros actualizables de configuración de base de datos (continuación)

Nombre de parámetro	Símbolo	Valor de símbolo	Tipo de datos
<ul style="list-style-type: none"> • auto_maint • auto_db_backup • auto_tbl_maint • auto_runstats • auto_stats_prof ¹ • auto_prof_upd ¹ • auto_reorg 	<ul style="list-style-type: none"> • SQLF_DBTN_AUTO_MAINT • SQLF_DBTN_AUTO_DB_BACKUP • SQLF_DBTN_AUTO_TBL_MAINT • SQLF_DBTN_AUTO_RUNSTATS • SQLF_DBTN_AUTO_STATS_PROF ¹ • SQLF_DBTN_AUTO_PROF_UPD ¹ • SQLF_DBTN_AUTO_REORG 	<ul style="list-style-type: none"> • 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
chnpggs_thresh	SQLF_DBTN_CHNGPGS_THRESH	38	Uint16
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
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)

Tabla 62. Parámetros actualizables de configuración de base de datos (continuación)

Nombre de parámetro	Símbolo	Valor de símbolo	Tipo de datos
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
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
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)
util_heap_sz	SQLF_DBTN_UTIL_HEAP_SZ	55	UInt32
vendoropt	SQLF_DBTN_VENDOROPT	829	char(242)

1.

Importante: La creación automática de perfiles de estadísticas queda en desuso en la Versión 10.1 y podría eliminarse en un release futuro. Para obtener más información, consulte el tema “Descripción automática de estadísticas está en desuso” en *Novedades en DB2 Versión 10.1*.

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)

Tabla 63. Parámetros no actualizables de configuración de base de datos

Nombre de parámetro	Símbolo	Valor de símbolo	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 a continuación)
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

sqlfupd

Tabla 63. Parámetros no actualizables de configuración de base de datos (continuación)

Nombre de parámetro	Símbolo	Valor de símbolo	Tipo de datos
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 a continuación)
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 símbolo SQLFUPD se indican en la lista siguiente:

Tabla 64. Parámetros actualizables de configuración del gestor de bases de datos

Nombre de parámetro	Símbolo	Valor de símbolo	Tipo de datos
agent_stack_sz	SQLF_KTN_AGENT_STACK_SZ	61	UInt16
agentpri	SQLF_KTN_AGENTPRI	26	Sint16
alternate_auth_enc	SQLF_KTN_ALTERNATE_AUTH_ENC	938	UInt16
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

Tabla 64. Parámetros actualizables de configuración del gestor de bases de datos (continuación)

Nombre de parámetro	Símbolo	Valor de símbolo	Tipo de datos
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
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_KEEFENCED	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)

Tabla 64. Parámetros actualizables de configuración del gestor de bases de datos (continuación)

Nombre de parámetro	Símbolo	Valor de símbolo	Tipo de datos
srv_plugin_mode	SQLF_KTN_SRV_PLUGIN_MODE	809	UInt16
srvcon_pw_plugin	SQLF_KTN_SRVCON_PW_PLUGIN	813	char(33)
ssl_cipherspecs	SQLF_KTN_SSL_CIPHERSPECS	934	char(255)
ssl_clnt_keydb	SQLF_KTN_SSL_CLNT_KEYDB	936	char(1023)
ssl_clnt_stash	SQLF_KTN_SSL_CLNT_STASH	937	char(1023)
ssl_svcename	SQLF_KTN_SSL_SVCENAME	933	char(14)
ssl_svr_keydb	SQLF_KTN_SSL_SVR_KEYDB	930	char(1023)
ssl_svr_label	SQLF_KTN_SSL_SVR_LABEL	932	char(1023)
ssl_svr_stash	SQLF_KTN_SSL_SVR_STASH	931	char(1023)
ssl_versions	SQLF_KTN_SSL_VERSIONS	935	char(255)
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_SYSMANT_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
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 **alternate_auth_enc** (definidos en sqlenv.h) son:

- SQL_ALTERNATE_AUTH_ENC_AES (0)
- SQL_ALTERNATE_AUTH_ENC_AES_CMP (1)
- SQL_ALTERNATE_AUTH_ENC_NOTSPEC (255)

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** (definidos 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** (definidos en `sqlutil.h`) son:

- SQLF_TRUST_ALLCLNTS_NO (0)
- SQLF_TRUST_ALLCLNTS_YES (1)
- SQLF_TRUST_ALLCLNTS_DRDAONLY (2)

Tabla 65. Parámetros no actualizables de configuración del gestor de bases de datos

Nombre de parámetro	Símbolo	Valor de símbolo	Tipo de datos
nodetype	SQLF_KTN_NODETYPE	100	Uint16
release	SQLF_KTN_RELEASE	101	Uint16

Los valores válidos para **nodetype** (definidos 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)

sqlfupd

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 66. Campos de la estructura sqllob

Nombre de campo	Tipo de datos	Descripción
longitud	sqluint32	Longitud, en bytes, del parámetro data.
datos	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 de área de supervisor de SQL (SQLMA) se utiliza para enviar peticiones de instantánea de supervisión de base de datos al gestor de bases de datos. También se utiliza para realizar estimaciones del tamaño (en bytes) de la salida de la instantánea.

Tabla 67. 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 68. 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. Solo 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. Nota: El valor SQLM_HMON_OPT_COLL_FULL ha quedado en desuso y se puede eliminar en una versión futura ya que el Supervisor de salud ha quedado en desuso en la Versión 9.7.
OBJ_TYPE	INTEGER	El tipo de objeto que deben supervisarse.
OBJECT	CHAR(128)	Nombre del objeto que debe supervisarse. Solo 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 include) son:

SQLMA_DB2

Información referente a la instancia.

SQLMA_DBASE

Información relacionada con la base de datos para 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 correspondientes a las conexiones establecidas con 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 (Directorio de servicios de conexión de bases 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 (Directorio de servicios de conexión de bases de datos) 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 (Directorio de servicios de conexión de bases de datos) activas.

SQLMA_DCS_APPL_ALL

Información sobre aplicaciones DCS (Directorio de servicios de conexión de bases de datos) para todas las conexiones.

SQLMA_DCS_APPL

Información sobre aplicaciones DCS (Directorio de servicios de conexión de bases de datos) para una aplicación que corresponde al ID 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 (Directorio de servicios de conexión de bases 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 (Directorio de servicios de conexión de bases 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 `object` 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;
    _SQLLDCHAR 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 sqlarwnd.

Tabla 69. Campos de la estructura SQLOPT

Nombre de campo	Tipo de datos	Descripción
HEADER	Estructura	Estructura sqloptheadr.
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 70. 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 71. 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 sqloptheadr header;
    SQL_STRUCTURE sqloptions option[1];
};

SQL_STRUCTURE sqloptheadr
{
    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 cronológicas.

Un número de secuencia de anotaciones cronológicas (LSN) representa una dirección de byte relativa dentro de la anotación cronológica de base de datos. Todos los registros de anotaciones cronológicas se identifican mediante este número. Un LSN representa el desplazamiento en bytes del registro de anotaciones cronológicas con respecto al principio del registro de anotaciones cronológicas de la base de datos.

Tabla 72. 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.

Nota: La estructura SQLU_LSN se ha sustituido por db2LSN.

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 73. 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 adecuada.
FILLER	CHAR(3)	Rellenador utilizado para la alineación correcta de la estructura de datos en la memoria.

Tabla 74. Campos de la estructura SQLU-MEDIA-LIST-TARGETS

Nombre de campo	Tipo de datos	Descripción
MEDIA	Puntero	Puntero a la estructura <code>sqlu_media_entry</code> .
VENDOR	Puntero	Puntero a una estructura <code>sqlu_vendor</code> .
LOCATION	Puntero	Puntero a la estructura <code>sqlu_location_entry</code> .
PSTATEMENT	Puntero	Puntero a la estructura <code>sqlu_statement_entry</code> .
PREMOTEFETCH	Puntero	Puntero a la estructura <code>sqlu_remotefetch_entry</code> .

Tabla 75. 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 76. 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 77. 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 78. 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 79. Campos de la estructura SQLU-REMOETFETCH-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.

Tabla 79. Campos de la estructura SQLU-REMOTEFETCH-ENTRY (continuación)

Nombre de campo	Tipo de datos	Descripción
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** (definidos 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). Solo puede especificarse para el parámetro **piSourceList** o **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.cb1
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.cb1
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.cb1
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.cb1
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 80. 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 cronológicas que se graba 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 cronológicas grabados en el almacenamiento intermedio.
logBytesWritten	sqluint32	Especifica el número de bytes grabados 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;
```

sqlupi

Esta estructura se utiliza para almacenar información sobre particionamiento, tal como la correlación de distribución y la clave de distribución de una tabla.

Tabla 81. Campos de la estructura SQLUPI

Nombre de campo	Tipo de datos	Descripción
PMAPLEN	INTEGER	Longitud de la correlación de distribución en bytes. Para tablas de un solo nodo, el valor es <code>sizeof(SQL_PDB_NODE_TYPE)</code> . Para tablas de varios nodos, el valor es <code>SQL_PDB_MAP_SIZE * sizeof(SQL_PDB_NODE_TYPE)</code> .
PMAP	SQL_PDB_NODE_TYPE	La correlación 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 <code>SQL_MAX_NUM_PART_KEYS</code> .

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 82. 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
Decimal	484	485	Byte 1 = precisión Byte 2 = escala	Sí

Tabla 82. Tipos de datos SQL y longitudes para la estructura SQLUPI (continuación)

Tipo de datos	SQLTYPE (Nulos no permitidos)	SQLTYPE (Nulos permitidos)	SQLLEN	AIX
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 gráfica 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 83. Campos de la estructura SQLXA-XID

Nombre de campo	Tipo de datos	Descripción
FORMATID	INTEGER	ID de formato XA.
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;
```

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 las 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 cronológicas de DB2

Este apartado describe la estructura de los registros de anotaciones cronológicas de DB2 devueltos por la API db2ReadLog cuando se especifica el valor de entrada de **iFilterOption** DB2READLOG_FILTER_ON.

Cuando se utiliza este valor sólo se devuelven los registros de anotaciones cronológicas marcados como propagables (el distintivo de propagable está definido en la cabecera del registro de anotaciones cronológicas). Sólo se documentan los registros de anotaciones cronológicas propagables. Todos los demás registros de anotaciones cronológicas están destinados solamente al uso interno de IBM y por lo tanto no están documentados.

Cuando una transacción invoca un programa de utilidad de grabación de anotaciones o realiza tareas de grabación para una tabla con el atributo DATA CAPTURE CHANGES establecido en ON, la transacción se marca como propagable. Solamente las transacciones propagables tienen marcados como propagables los registros de anotaciones del gestor de transacciones.

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, rastros ni evaluación del rendimiento. Para obtener más información consulte "Cabecera de registro de anotaciones cronológicas" en la página 725.

Los registros de anotaciones se identifican de forma unívoca mediante un identificador de registro de anotaciones cronológicas (LRI). Debido a que los identificadores de números de secuencia de registro (LSN) de 64 bits que determinan el orden de las operaciones que generaron los registros de anotaciones cronológicas se generan independientemente en cada miembro y hay varias corrientes de anotaciones cronológicas, es posible tener valores LSN duplicados en distintas corrientes de anotaciones cronológicas. Se utiliza el LRI para identificar los registros de anotaciones cronológicas en las corrientes de registros de anotaciones cronológicas; a cada registro de anotaciones cronológicas de cualquier corriente de la base de datos se le asigna un LRI exclusivo. Use el mandato **db2pd** para determinar qué LRI se está procesando por parte de una operación de recuperación.

Los registros de anotaciones cronológicas grabados por una transacción individual se identifican unívocamente mediante un campo contenido en la cabecera del registro de anotaciones cronológicas. 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 cronológicas grabados por una transacción individual contienen el mismo identificador.

Tabla 84. Registros de anotaciones cronológicas de DB2

Tipo	Nombre de registro	Descripción
Gestor de datos	"Registro de anotaciones de inicialización de tabla" en la página 746	Creación de nueva tabla permanente.
Gestor de datos	"Registro de anotaciones de sustitución de importación (truncamiento)" en la página 748	Actividad de importar sustitución.

Registros de anotaciones cronológicas de DB2

Tabla 84. Registros de anotaciones cronológicas de DB2 (continuación)

Tipo	Nombre de registro	Descripción
Gestor de datos	"Registro de anotaciones de activación no registrada inicialmente" en la página 749	La actividad de modificar tabla que incluye la cláusula ACTIVATE NOT LOGGED INITIALLY.
Gestor de datos	"Registro de anotaciones de retrotracción de inserción" en la página 749	Retrotraer inserción de fila.
Gestor de datos	"Registro de anotaciones de reorganización de tabla" en la página 750	REORG confirmado.
Gestor de datos	"Registros de anotaciones de crear índice, descartar índice" en la página 751	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 751	Actividad de tabla.
Gestor de datos	"Registro de anotaciones cronológicas de atributo de alteración de tabla" en la página 752	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 753	Adición de columnas a tablas existentes.
Gestor de datos	"Registro de anotaciones de atributo de alterar columna" en la página 754	Actividad de columnas.
Gestor de datos	"Registro de anotaciones del atributo de deshacer alteración de columna" en la página 754	Actividad de columna.
Gestor de datos	"Registro de anotaciones de insertar registro, eliminar registro, retrotraer supresión de registro, retrotraer actualización de registro" en la página 754	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 759	Actividad de tabla de clúster multidimensional(MDC).
Gestor de datos	"Registro de anotaciones de actualizar registro" en la página 761	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 761	Actividad de nombres de esquema o tabla.

Registros de anotaciones cronológicas de DB2

Tabla 84. Registros de anotaciones cronológicas de DB2 (continuación)

Tipo	Nombre de registro	Descripción
Gestor de datos	"Registro de anotaciones cronológicas de deshacer renombramiento de una tabla o un esquema" en la página 762	Actividad de nombres de esquema o tabla.
Gestor de campos largos	"Registro de anotaciones de adición/supresión/no actualización de campo largo" en la página 739	Actividad de registro en campos largos.
Gestor de transacciones	"Registro de anotaciones de confirmación normal" en la página 728	Se confirma la transacción.
Gestor de transacciones	"Registro de anotaciones de confirmación heurística" en la página 729	Se confirma la transacción pendiente.
Gestor de transacciones	"Registro de anotaciones de confirmación de coordinador de MPP" en la página 729	Se confirma la transacción. Esto se graba 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 730	Se confirma la transacción. Esto se graba en un nodo de subordinador.
Gestor de transacciones	"Registro de anotaciones de cancelación normal" en la página 731	Se cancela la transacción.
Gestor de transacciones	"Registro de anotaciones de cancelación heurística" en la página 732	Se cancela la transacción pendiente.
Gestor de transacciones	"Registro de anotaciones de lista pendiente local" en la página 732	Se confirma la transacción con una lista pendiente existente.
Gestor de transacciones	"Registro de anotaciones de lista pendiente global" en la página 733	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 733	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 734	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 735	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.

Registros de anotaciones cronológicas de DB2

Tabla 84. Registros de anotaciones cronológicas de DB2 (continuación)

Tipo	Nombre de registro	Descripción
Gestor de transacciones	“Registro de anotaciones libre de retrotracción” en la página 736	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 la transacción.
Gestor de transacciones	“Registro de anotaciones de información de la aplicación” en la página 737	Información sobre la aplicación que ha iniciado la transacción.
Gestor de transacciones	“Registro de anotaciones de preparación federada” en la página 737	Información sobre el gestor de recursos federado implicado en la transacción.
Gestor de programas de utilidad	“Registro de anotaciones cronológicas del inicio de la migración del catálogo del sistema” en la página 740	Empieza la migración del catálogo del sistema.
Gestor de programas de utilidad	“Registro de anotaciones cronológicas del fin de la migración del catálogo del sistema” en la página 741	Se completa la migración del catálogo del sistema.
Gestor de programas de utilidad	“Registro de anotaciones de inicio de carga” en la página 741	Se inicia la carga de la tabla.
Gestor de programas de utilidad	“Registro de anotaciones de final de copia de seguridad” en la página 742	Finaliza la actividad de copia de seguridad.
Gestor de programas de utilidad	“Registro de anotaciones de avance del espacio de tablas” en la página 742	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 743	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 cronológicas de final del avance del espacio de tablas hasta punto en el tiempo” en la página 743	Indica el final de un avance de un espacio de tablas hasta un punto en el tiempo.
Gestor de relaciones	“registro de anotaciones cronológicas de las sentencias DDL” en la página 766	Actividad DDL.
Gestor de relaciones	“Deshacer registro de anotaciones cronológicas de las sentencias DDL” en la página 770	Deshacer registro de anotaciones cronológicas para actividad DDL.
Gestor de relaciones	“Registro de anotaciones cronológicas de información de la partición” en la página 770	Actividad de partición

Cabecera de registro de anotaciones cronológicas

Todos los registros de anotaciones cronológicas de DB2 comienzan con una cabecera del gestor de anotaciones cronológicas.

Cabecera de registro de anotaciones cronológicas

Esta cabecera contiene información sobre el registro de anotaciones e información de transacciones del transcriptor de registros de anotaciones.

Tabla 85. Cabecera del registro de anotaciones del gestor de anotaciones cronológicas (LogManagerLogRecordHeader).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción de campo	Tipo de datos	Desplazamiento (bytes)
Longitud del registro de anotaciones cronológicas	db2UInt32	0(4)
Tipo del registro de anotaciones cronológicas (Vea la Tabla 86 en la página 727)	db2UInt16	4(2)
Distintivo del registro de anotaciones cronológicas ¹	db2UInt16	6(2)
Número de secuencia de anotaciones cronológicas (LSN) del registro de anotaciones cronológicas	db2LSN ²	8(8)
Desplazamiento de secuencia de anotaciones cronológicas (LSO) del registro de anotaciones cronológicas anterior en esta transacción	db2UInt64	24(8)
Identificador de transacción exclusivo	SQLU_TID ³	32(6)
ID de corriente de anotaciones cronológicas	db2LogStreamIDType	38(2)
ID de corriente de anotaciones cronológicas adicional del LSO anterior para registros de anotaciones cronológicas de compensación	db2LogStreamIDType	40(2)
LSO adicional para registros de anotaciones cronológicas de compensación	db2UInt64	48(8)
LSO adicional para registros de anotaciones cronológicas de compensación propagables	db2UInt64	56(8)
<p>Nota: Una cabecera básica del registro de anotaciones cronológicas tiene una longitud total de 40 bytes y el ID de corriente de anotaciones cronológicas no está establecido. Los registros de anotaciones cronológicas de compensación solicitan un valor de LSN extra en el extremo, para alcanzar una longitud total de 48 bytes. Los registros de anotaciones cronológicas de compensación propagables solicitan un valor de LSN adicional en el extremo, para alcanzar una longitud total de 56 bytes. En ambos casos, el campo de ID de corriente de anotaciones cronológicas se ha establecido adecuadamente.</p>		

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 restriccc. comp.	0x0100
Retrotracción de tiempo de ejecución	0x0200
Pseudocompensación	0x0800

2. Número de secuencia de anotaciones cronológicas (LSN)

Cabecera de registro de anotaciones cronológicas

Identificador de 64 bits que determina el orden de las operaciones que generó los registros de anotaciones cronológicas. LSN es un valor en continuo crecimiento. Cada miembro graba en su propio conjunto de archivos de anotaciones cronológicas (corriente de anotaciones cronológicas), y el LSN en una única corriente de anotaciones cronológicas es un número exclusivo.

```
db2LSN: { db2Uint64 lsnU64;
          }
```

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 86. Valores y definiciones para el tipo de registro de cabecera del gestor de anotaciones cronológicas

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 del catálogo del sistema
0x0057	Fin de migración del catálogo del sistema
0x0069	Sólo informativo. Si el tipo de registro de anotaciones es 0x0069, es un registro informativo solamente. El gestor de bases de datos DB2 no lo tiene en cuenta durante las operaciones de avance, retrotracción y recuperación de anomalía general.
0x006F	Inicio de copia de seguridad
0x0071	Avance del espacio de tablas hasta finalización 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

Cabecera de registro de anotaciones cronológicas

Tabla 86. Valores y definiciones para el tipo de registro de cabecera del gestor de anotaciones cronológicas (continuación)

Valor	Definición
0x0086	confirmación de coordinador de MPP
0x0087	Confirmación heurística
0x0089	Avance del espacio de tablas hasta inicio de 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 cronológicas se graba 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 cronológicas se graba 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 87. Estructura del registro de anotaciones para la confirmación normal.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Hora en que se confirmó la transacción	sqluint64	22 (8)
Longitud del identificador de autorización ¹ (si el registro de anotaciones está marcado como propagable)	unsigned short	30(2)
Identificador de autorización de la aplicación ¹ (si el registro de anotaciones está marcado como propagable)	char []	32 (variable ²)
<i>Longitud total: 32 bytes más variable propagable (30 bytes no propagables)</i>		

Nota:

Registros de anotaciones del gestor de transacciones

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 cronológicas se graba cuando se confirma una transacción dudosa.

Tabla 88. Estructura del registro de anotaciones para la confirmación heurística.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Hora en que se confirmó la transacción	sqluint64	22 (8)
Longitud del identificador de autorización ¹ (si el registro de anotaciones está marcado como propagable)	unsigned short	30(2)
Identificador de autorización de la aplicación ¹ (si el registro de anotaciones está marcado como propagable)	char []	32 (variable ²)
<i>Longitud total: 32 bytes más variable propagable (30 bytes no propagables)</i>		

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 cronológicas se graba en un nodo de coordinador para una aplicación que realiza actualizaciones en un nodo de subordinador como mínimo.

Tabla 89. Estructura del registro de anotaciones para la confirmación de coordinador MPP.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Hora en que se confirmó la transacción	sqluint64	22 (8)
Identificador MPP de la transacción	SQLP_GXID	30 (20)
Número de nodo máximo	unsigned short	50(2)

Registros de anotaciones del gestor de transacciones

Tabla 89. Estructura del registro de anotaciones para la confirmación de coordinador MPP (continuación).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
TNL	unsigned char []	52 (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)
<i>Longitud total: variable</i>		

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

Este registro de anotaciones cronológicas se graba en un nodo de subordinador en MPP.

Tabla 90. Estructura del registro de anotaciones para la confirmación de subordinador MPP.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Hora en que se confirmó la transacción	sqluint64	22 (8)
Identificador MPP de la transacción	SQLP_GXID	30 (20)
Número de partición coordinadora ¹	unsigned short	50(2)
Longitud del identificador de autorización (si el registro de anotaciones está marcado como propagable)	unsigned short	52 (2)

Registros de anotaciones del gestor de transacciones

Tabla 90. Estructura del registro de anotaciones para la confirmación de subordinador MPP (continuación).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Identificador de autorización de la aplicación ² (si el registro de anotaciones está marcado como propagable)	char []	54 (variable ³)
<i>Longitud total: 54 bytes más variable</i>		

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 cronológicas se graba 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 de anomalía general
- Se produce una retrotracción implícita durante una recuperación **ROLLFORWARD** (en avance).

Tabla 91. Estructura del registro de anotaciones para la cancelación normal.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Longitud del identificador de autorización ¹ (si el registro de anotaciones está marcado como propagable)	unsigned short	22(2)
Identificador de autorización de la aplicación ¹ (si el registro de anotaciones está marcado como propagable)	char []	24 (variable ²)
<i>Longitud total: 24 bytes más variable propagable (22 bytes no propagables)</i>		

Nota:

1. Si el registro de anotaciones está marcado como propagable

Registros de anotaciones del gestor de transacciones

- Variable basada en la longitud del identificador de autorización

Registro de anotaciones de cancelación heurística

El registro de anotaciones cronológicas se graba cuando se cancela una transacción dudosa.

Tabla 92. Estructura del registro de anotaciones para la cancelación heurística.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Longitud del identificador de autorización ¹ (si el registro de anotaciones está marcado como propagable)	unsigned short	22(2)
Identificador de autorización de la aplicación ¹ (si el registro de anotaciones está marcado como propagable)	char []	24 (variable ²)
<i>Longitud total: 24 bytes más variable propagable (22 bytes no propagables)</i>		

Nota:

- Si el registro de anotaciones está marcado como propagable
- Variable basada en la longitud del identificador de autorización

Registro de anotaciones de lista pendiente local

Este registro de anotaciones cronológicas se graba 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 93. Estructura del registro de anotaciones para la lista pendiente local.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Hora en que se confirmó la transacción	sqluint64	22 (8)
Longitud del identificador de autorización ¹	unsigned short	30(2)
Identificador de autorización de la aplicación ¹	char []	32 (variable ²)
Entradas de la lista pendiente	variable	variable (variable)

Registros de anotaciones del gestor de transacciones

Tabla 93. Estructura del registro de anotaciones para la lista pendiente local (continuación).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
<i>Longitud total: 32 bytes más variables propagables (30 bytes más entradas de lista pendiente no propagables)</i>		

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 cronológicas se graba 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 94. Estructura del registro de anotaciones para la lista pendiente global.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Longitud del identificador de autorización ¹	unsigned short	22(2)
Identificador de autorización de la aplicación ¹	char []	24 (variable) ²
Entradas de la lista pendiente global	variable	variable (variable)
<i>Longitud total: 24 bytes más variables propagables (22 bytes más entradas de lista pendiente no propagables)</i>		

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 cronológicas se graba 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 cronológicas se graba para indicar la preparación de la transacción como parte de una

Registros de anotaciones del gestor de transacciones

confirmación en dos fases. El registro de anotaciones cronológicas de preparación de XA describe la aplicación que ha iniciado la transacción, y se utiliza para volver a crear una transacción dudosa.

Tabla 95. Estructura del registro de anotaciones de preparación de XA.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Hora en que se preparó la transacción	sqluint64	22 (8)
Espacio de registro de anotaciones utilizado por la transacción	sqluint64	30 (8)
Tamaño de la lista de nodos de la transacción	sqluint32	38 (4)
Lista de nodos de la transacción	unsigned char []	42 (variable)
Reservado	sqluint32	variable(2)
Identificador XA de la transacción	SQLXA_XID ¹	variable(140)
Información de Synclog	variable	variable (variable)
<i>Longitud total: 184 bytes más variables</i>		

Nota: 1. Para obtener detalles sobre el tipo de anotaciones cronológica SQLXA_XID, consulte "SQLXA_XID" en la página 717.

Registro de anotaciones cronológicas de preparación de subordinador de MPP

Este registro de anotaciones cronológicas se graba para transacciones MPP en nodos de subordinador. El registro de anotaciones cronológicas se graba para indicar la preparación de la transacción como parte de una confirmación en dos fases. El registro de anotaciones cronológicas de preparación de MPP describe la aplicación que ha iniciado la transacción, y se utiliza para volver a crear una transacción dudosa.

Tabla 96. Estructura del registro de anotaciones para la preparación del subordinador MPP.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Hora en que se preparó la transacción	sqluint64	22 (8)

Registros de anotaciones del gestor de transacciones

Tabla 96. Estructura del registro de anotaciones para la preparación del subordinador MPP (continuación).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Espacio de registro de anotaciones utilizado por la transacción	sqluint64	30 (8)
LSN de coordinador	db2LSN	38 (8)
Relleno	char []	46 (2)
Identificador MPP de la transacción	SQLP_GXID ¹	48(20)
<i>Longitud total: 68 bytes</i>		

Nota: 1.El registro de anotaciones SQLP-GXID se utiliza para identificar transacciones en un entorno MPP.

Tabla 97. 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 graba para transacciones coordinadas de DB2 en un entorno de bases de datos de una sola partición o en la partición coordinadora en MPP, donde la base de datos actúa como base de datos de TM. El registro de anotaciones cronológicas se graba para indicar la preparación de la transacción como parte de una confirmación en dos fases.

Tabla 98. Estructura del registro de anotaciones de preparación de TM.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Hora en que se preparó la transacción	sqluint64	22 (8)
Espacio de registro de anotaciones utilizado por la transacción	sqluint64	30 (8)

Registros de anotaciones del gestor de transacciones

Tabla 98. Estructura del registro de anotaciones de preparación de TM (continuación).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Tamaño de la lista de nodos de la transacción	sqluint32	38 (4)
Lista de nodos de la transacción	unsigned char []	42 (variable)
Reservado	sqluint32	variable(2)
Identificador XA de la transacción	SQLXA_XID	variable(140)
Información de Synclog	variable	variable (variable)
<i>Longitud total: 184 bytes más variables</i>		

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 la transacción. Este registro de anotaciones cronológicas contiene un número de secuencia de anotaciones cronológicas de 8 bytes (*compsln*, almacenado en la cabecera de registro de anotaciones cronológicas que comienza en la posición cuyo desplazamiento es 22). En determinadas situaciones, el registro de anotaciones cronológicas libre de retrotracción también contiene datos de anotaciones cronológicas, que comienzan en la posición con desplazamiento 30, que son los mismos datos que se registran en los registros de anotaciones cronológicas del gestor de datos correspondientes. Cuando este registro de anotaciones se lee durante una retrotracción (después de una transacción cancelada), *compsln* indica el siguiente registro de anotaciones que se debe compensar.

Tabla 99. Estructura del registro de anotaciones libre de retrotracción.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Compsln	db2LSN	22 (8)
Datos de registro de anotaciones ¹	variable	variable
<i>Longitud total: 30 bytes más variables</i>		

Nota: 1. Solamente se aplica en determinadas situaciones, y cuando se utiliza, la longitud del registro de anotaciones cronológicas completo contenido en la cabecera de anotaciones cronológicas es mayor que 28 bytes.

Registros de anotaciones del gestor de transacciones

Registro de anotaciones de información de la aplicación

Este registro de anotaciones contiene información sobre la aplicación que ha iniciado esta transacción.

Tabla 100. Estructura del registro de anotaciones de información de la aplicación.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Hora de inicio de la transacción	sqluint32	22 (4)
Reservado	char[]	26 (16)
Página de códigos	sqluint32	42 (4)
Longitud del nombre de la aplicación	sqluint32	46 (4)
Nombre de la aplicación	char []	50 (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)
<i>Longitud total: 66 bytes más variables</i>		

Registro de anotaciones de preparación federada

Este registro de anotaciones contiene información sobre los gestores de recursos federados implicados en la transacción.

Tabla 101. Estructura del registro de anotaciones de preparación federada.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Número de gestores de recursos	sqluint32	22 (4)

Registros de anotaciones del gestor de transacciones

Tabla 101. Estructura del registro de anotaciones de preparación federada (continuación).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Longitud del identificador de autorización	sqluint16	26(2)
Longitud de la contraseña cifrada	sqluint16	28(2)
Identificador de autorización	char [128]	30 (128)
Contraseña cifrada	char [263]	158 (263)
Entradas del gestor de recursos	variable	421 (variable)
<i>Longitud total: 421 bytes más variables</i>		

Registros de anotaciones cronológicas del gestor de campos largos

Solo se graban los registros de anotaciones cronológicas del gestor de campos largos si una base de datos es recuperable. Es decir, si la base de datos se configura con el parámetro **logarchmeth1** o **logarchmeth2** con un valor diferente de OFF. Los registros de anotaciones cronológicas se graban cada vez que se añaden, 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 cronológicas del gestor de campos largos están determinados con respecto al final de la cabecera de registro de anotaciones cronológicas del gestor de anotaciones cronológicas.

Cuando se modificó una tabla para capturar columnas LONG VARCHAR OR LONG VARGRAPHIC mediante la especificación de INCLUDE LONGVAR COLUMNS en la sentencia ALTER TABLE:

- El gestor de campos largos guarda el correspondiente registro de anotaciones de campos largos.
- 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 o añadir campo largo está asociado o no a una operación de actualización de la tabla, el valor original de la operación se registraría en el registro de anotaciones del gestor de campos largos.
- Cuando se actualizan tablas que tienen columnas de campos largos, pero sin que se actualicen estas, se guarda un registro de no actualización de campos largos.

Registros de anotaciones cronológicas del gestor de campos largos

- El registro de supresión de campo largo y el registro de no actualización de campo largo son registros de anotaciones cronológicas con fines informativos solamente.

Tabla 102. Cabecera del registro de anotaciones del gestor de campos largos (LongFieldLogRecordHeader).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Código de origen (identificador de componente = 3)	unsigned char	0(1)
Tipo de operación (Vea la Tabla 103).	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)
<i>Longitud total: 10 bytes</i>		

Nota:

1. ID de espacio de tablas del objeto de datos
2. ID de objeto del objeto de datos

Tabla 103. Definiciones y valores del tipo de operaciones de cabecera del registro de anotaciones 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 cronológicas se graban 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 104. Estructura del registro de anotaciones de adición/supresión/no actualización de campo largo

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LongFieldLogRecordHeader	0(10)
Interno	Interno	10(1)

Registros de anotaciones cronológicas del gestor de campos largos

Tabla 104. Estructura del registro de anotaciones de adición/supresión/no actualización de campo largo (continuación)

Descripción	Tipo	Desplazamiento (Bytes)
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 graba registros de anotaciones cronológicas 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 cronológicas 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: actualización de base de datos; carga; copia de seguridad; avance de espacios 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 cronológicas del inicio de la migración del catálogo del sistema

Durante la actualización de la base de datos, los objetos del catálogo del sistema se convierten al formato del nuevo release. Este registro de anotaciones cronológicas indica el inicio de la migración del catálogo del sistema.

Registros de anotaciones cronológicas del gestor de programas de utilidad

Tabla 105. Estructura del registro de anotaciones cronológicas de inicio de la migración del catálogo del sistema.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Hora de inicio	char[]	22 (10)
Release anterior	unsigned short	32(2)
Nuevo release	unsigned short	34 (2)
<i>Longitud total: 36 bytes</i>		

Registro de anotaciones cronológicas del fin de la migración del catálogo del sistema

Durante la actualización de la base de datos, los objetos del catálogo del sistema se convierten al formato del nuevo release. Este registro de anotaciones cronológicas indica la conclusión satisfactoria de la migración del catálogo del sistema.

Tabla 106. Estructura del registro de anotaciones cronológicas de fin de la migración del catálogo del sistema.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Hora de finalización	char[]	22 (10)
Nuevo release	unsigned short	32(2)
<i>Longitud total: 34 bytes</i>		

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.

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.

Registros de anotaciones cronológicas del gestor de programas de utilidad

Tabla 107. Estructura del registro de anotaciones de comienzo de carga.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Identificador de registro de anotaciones	sqluint32	22 (4)
Identificador de agrupación	unsigned short	26(2)
Identificador de objeto	unsigned short	28(2)
Distintivo	sqluint32	30 (4)
Lista de agrupaciones de objetos	variable	34 (variable)
<i>Longitud total: 34 bytes más variable</i>		

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 108. Estructura del registro de anotaciones de final de copia de seguridad.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Hora de finalización de la copia de seguridad	sqluint64	22 (8)
<i>Longitud total: 30 bytes</i>		

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. Se graba para cada espacio de tablas para el que se ha realizado satisfactoriamente un avance.

Tabla 109. Estructura del registro de anotaciones de avance del espacio de tablas.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Identificador de espacio de tablas	sqluint32	22 (4)
<i>Longitud total: 26 bytes</i>		

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 110. Estructura de registro de anotaciones cronológicas del inicio del avance del espacio de tablas hasta punto en el tiempo.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Indicación de fecha y hora para este registro de anotaciones cronológicas	sqluint64	22 (8)
Indicación de fecha y hora para la que se realiza el avance de los espacios de tablas	sqluint32	30 (4)
Número de agrupaciones sometidas al avance	sqluint32	34 (4)
<i>Longitud total: 38 bytes</i>		

Registro de anotaciones cronológicas 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 111. Estructura del registro de anotaciones cronológicas del final del avance del espacio de tablas hasta punto en el tiempo.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	LogManagerLogRecordHeader	0 (22)
Indicación de fecha y hora para este registro de anotaciones cronológicas	sqluint64	22 (8)
Indicación de fecha y hora a la que se realizó el avance de los espacios de tablas	sqluint32	30 (4)
Indicador cuyo valor es TRUE si el avance fue correcto o FALSE si el avance se canceló.	sqluint32	34 (4)

Registros de anotaciones cronológicas del gestor de programas de utilidad

Tabla 111. Estructura del registro de anotaciones cronológicas del final del avance del espacio de tablas hasta punto en el tiempo (continuación).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
<i>Longitud total: 38 bytes</i>		

Se necesitan dos campos de indicación de fecha y hora para proporcionar una precisión adecuada, de manera que puedan diferenciarse los tiempos de los sucesos de las anotaciones cronológicas de sucesos. La primera indicación de fecha y hora utiliza 8 bytes para indicar la hora a la que se grabó la anotación cronológica con una precisión de segundos. Los 4 primeros bytes de esta indicación de fecha y hora indican la parte correspondiente a los 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 como contador para indicar el orden de los registros de anotaciones cronológicas que se producen en el mismo segundo. Si las indicaciones de fecha y hora de dos registros de anotaciones cronológicas son idénticas, el campo del contador de indicación de fecha y hora de 4 bytes adicional puede utilizarse para determinar el orden de los sucesos de anotaciones cronológicas asociados.

Registros de anotaciones del gestor de datos

Los registros de anotaciones cronológicas 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 112. Estructura de la cabecera del registro de anotaciones de DMS (*DMSLogRecordHeader*).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Identificador de componente (=1)	unsigned char	0(1)
Identificador de función (Vea la Tabla 113 en la página 745.)	unsigned char	1(1)
Identificadores de tabla		
Identif. espacio tablas	unsigned short	2(2)
Identificador de tabla	unsigned short	4(2)
<i>Longitud total: 6 bytes</i>		

Registros de anotaciones del gestor de datos

Tabla 113. 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
108	Deshacer actualización de estado de partición
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
137	Actualizar estado de partición
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 114. Estructura de la cabecera del registro de anotaciones de DOM (DOMLogRecordHeader).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Identificador de componente (=4)	unsigned char	0(1)
Identificador de función (Vea la Tabla 115 en la página 746.)	unsigned char	1(1)
Identificadores de objeto		
Identif. espacio tablas	unsigned short	2(2)
Identificador de objeto	unsigned short	4(2)

Registros de anotaciones del gestor de datos

Tabla 114. Estructura de la cabecera del registro de anotaciones de DOM (DOMLogRecordHeader) (continuación).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Identificadores de tabla		
Identif. espacio tablas	unsigned short	6(2)
Identificador de tabla	unsigned short	8(2)
Tipo de objeto	unsigned char	10(1)
Distintivos	unsigned char	11(1)
<i>Longitud total: 12 bytes</i>		

Tabla 115. 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 cronológicas del gestor de datos están determinados con respecto al final de la cabecera de registro del gestor de anotaciones cronológicas.

Todos los registros de anotaciones cronológicas cuyo nombre corto de identificador de función comienza con UNDO son registros de anotaciones cronológicas grabados 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 de anomalía general
- 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 cronológicas de inicializar tabla se graba 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

Registros de anotaciones del gestor de datos

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 116. Estructura del registro de anotaciones de inicialización de tabla.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	DMSLogRecordHeader	0(6)
LSN de creación de archivo	db2LSN	6(8)
Interno	Interno	14(74)
Longitud de la descripción de la tabla	sqluint32	88(4)
Registro de descripción de la tabla	variable	92(variable)
<i>Longitud total: 92 bytes más longitud del registro de descripción de la tabla</i>		

Tabla 117. Registro de descripción de la tabla.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página 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	variable
<i>Longitud total: 4 bytes más la longitud de la matriz de descriptores de columna</i>		

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

Registros de anotaciones del gestor de datos

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)
- longitud en línea (INLINE_LENGTH, sqluint16, 2 bytes)
- distintivo de anotaciones cronológicas (IS COLUMN LOGGED, sqluint16, 2 bytes)
- reservado (internal, 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 cronológicas de importar sustitución (truncar) se graba 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 cronológicas identifican la tabla que se está truncando (IMPORT REPLACE). Esto es un registro de anotaciones normal. El ID de función es 11.

Registros de anotaciones del gestor de datos

Tabla 118. Estructura del registro de anotaciones de sustitución de importación (truncamiento).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	DOMLogRecordHeader	0(12)
Interno	Interno	12(variable)
<i>Longitud total: 12 bytes más longitud variable</i>		

Registro de anotaciones de activación no registrada inicialmente

El registro de anotaciones cronológicas de activación no registrada inicialmente se graba 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 119. Estructura del registro de anotaciones de activación no registrada inicialmente.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	DOMLogRecordHeader	0(12)
Interno	Interno	12(4)
ID espacio de tablas largo*	unsigned short	16(2)
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)
<i>Longitud total: 28 bytes</i>		

* 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 cronológicas de retrotraer inserción se graba 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.

Registros de anotaciones del gestor de datos

Tabla 120. Estructura del registro de anotaciones de retroacción de inserción.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	DMSLogRecordHeader	0(6)
Interno	Interno	6(2)
Longitud de registro	unsigned short	8(2)
Espacio libre	unsigned short	10(2)
RID	char[]	12(6)
<i>Longitud total: 16 bytes</i>		

IrlUDflags

Tabla 121. Definición y valores de IrlUDflags

Valor	Definición
0x0200	La operación es el resultado de la redistribución o la carga
0x0400	Es el resultado de la operación temporal interna
0x0800	Es el resultado de la operación de mantenimiento MQT
0x1000	Es el resultado de la operación de integridad referencial
0x2000	La operación es el resultado de un activador
0x4000	La operación es el resultado de una sentencia SET INTEGRITY
0x8000	Es el resultado de una actualización de descomposición

Registro de anotaciones de reorganización de tabla

El registro de anotaciones cronológicas de reorganizar tabla se graba 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 122. Estructura del registro de anotaciones de reorganización de tabla.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	DOMLogRecordHeader	0(12)
Interno	variable	12(476)
Símbolo de índice ¹	unsigned short	488(2)
ID de espacio de tablas temporal ²	unsigned short	490(2)

Registros de anotaciones del gestor de datos

Tabla 122. Estructura del registro de anotaciones de reorganización de tabla (continuación).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
ID de espacio de tablas temporal largo	unsigned short	492(2)
<i>Longitud total: 494 bytes</i>		

Nota:

1. Si el valor del símbolo de índice no es 0, este valor es el índice utilizado para agrupar la reorganización (índice de clústeres).
2. Si el ID del espacio de tablas temporal no es 0, este valor es el espacio de tablas temporal del sistema que se utilizó para crear la tabla reorganizada.

Registros de anotaciones de crear índice, descartar índice

Estos registros de anotaciones cronológicas se graban 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 equivale a la columna IID de SYSCAT.INDEXES. 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 123. Estructura de los registros de anotaciones de crear índice, descartar índice.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	DOMLogRecordHeader	0(12)
Interno	Interno	12(2)
Símbolo de índice	unsigned short	14(2)
Página raíz del índice	sqluint32	16(4)
<i>Longitud total: 20 bytes</i>		

Registros de anotaciones para crear tabla, descartar tabla, retrotraer creación de tabla, retrotraer el descarte de la tabla

Estos registros de anotaciones cronológicas se graban cuando se crea o se descarta el objeto DATA para una tabla permanente. Para la creación de una tabla MDC o ITC, existe también un registro de anotaciones de crear tabla para la creación del objeto Mapa de bloques. El objeto DATA (y el objeto de mapa de bloques si se aplica) 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

Registros de anotaciones del gestor de datos

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 124. Estructura de los registros de anotaciones para crear tabla, descartar tabla, retrotraer creación de tabla, retrotraer el descarte de la tabla.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	DOMLogRecordHeader	0(12)
Interno	variable	12(72)
<i>Longitud total: 84 bytes</i>		

Registro de anotaciones cronológicas de atributo de alteración de tabla

El registro de anotaciones cronológicas del atributo de alterar tabla se graba 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 125. Atributo de alterar tabla, atributo de deshacer alteración de tabla.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	DMSLogRecordHeader	0(6)
Máscara (atributo) de alterar bit	sqluint64	6(8)
Valores (atributo) de alterar bit	sqluint64	14(8)
<i>Longitud total: 22 bytes</i>		

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 cronológicas de añadir columnas por alteración de tabla se graba cuando el usuario añade a columnas a una tabla existente mediante una sentencia ALTER TABLE. Se anota cronológicamente información completa acerca de 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 con anterioridad a la sentencia ALTER TABLE, mientras que la nueva matriz paralela define la tabla resultante desde 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 126. Estructura de los registros de anotaciones de añadir columnas por alteración de tabla, retrotraer adición de columnas.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	DMSLogRecordheader	0(6)
Interno	Interno	6(2)
Número de columnas antiguo	sqluint32	8(4)
Número de columnas nuevo	sqluint32	12(4)
Matrices paralelas antiguas ¹	variable	16(variable)
Matrices paralelas nuevas	variable	variable(variable)
<i>Longitud total: 16 bytes más 2 conjuntos de matrices paralelas.</i>		

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, consulte la descripción que hay después de la Tabla 117 en la página 747.

Registros de anotaciones del gestor de datos

Registro de anotaciones de atributo de alterar columna

El ID de función es 113.

Tabla 127. Estructura del registro de anotaciones de atributo de alterar columna.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	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)
<i>Longitud total: 24 bytes más longitud de registro.</i>		

¹ Para obtener una descripción de la matriz de descriptores de columna, consulte la descripción que hay después de la Tabla 117 en la página 747.

Registro de anotaciones del atributo de deshacer alteración de columna

El ID de función es 115.

Tabla 128. Estructura del registro de anotaciones del atributo de deshacer alteración de columna.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	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)
<i>Longitud total: 24 bytes más longitud de registro.</i>		

¹ Para obtener una descripción de la matriz de descriptores de columna, consulte la descripción que hay después de la Tabla 117 en la página 747.

Registro de anotaciones de insertar registro, eliminar registro, retrotraer supresión de registro, retrotraer actualización de registro

Estos registros de anotaciones cronológicas se graban 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

Registros de anotaciones del gestor de datos

registro son registros de anotaciones normales. Los registros Retrotraer supresión y Retrotraer actualización son registros de anotaciones de compresión. Los ID de función son 162 (insertar), 161 (eliminar), 111 (retrotraer supresión) o 112 (retrotraer actualización).

Tabla 129. Estructura de registro de anotaciones de insertar registro, eliminar registro, retrotraer supresión de registro, retrotraer actualización de registro.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	DMSLogRecordHeader	0(6)
lrIUDflags	unsigned short	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	variable	20(variable)
<i>Longitud total: 20 bytes más longitud de registro</i>		

A continuación se ofrecen detalles sobre la cabecera y los datos del registro:

Cabecera de anotaciones cronológicas

- 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 758.

Nota: la matriz de desplazamiento contendrá 1 + el número de columnas.

Registros de anotaciones del gestor de datos

- Registro formateado (matriz de desplazamiento, seguida de las columnas de datos).

Tabla 130. Estructura del registro de anotaciones de retrotraer supresión de registro y retrotraer actualización de registro.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	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	variable	20(variable)
<i>Longitud total: 20 bytes más longitud de registro</i>		

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
 - El tipo de registro interior tiene el bit 0x01 establecido o
 - El tipo de registro interior tiene el bit 0x02 establecido.

Extracción de los datos LOB en línea desde un registro de datos de usuario formateado

Los datos LOB en línea pueden extraerse desde el registro de datos de usuario. Una vez ubicado el inicio de los datos de columna LOB (basándose en que la tabla tenga habilitado VALUE COMPRESSION o no), el examen del primer byte puede mostrar la presencia de los datos LOB en línea.

Si el primer byte es 0x69, marca el principio de una cabecera de datos LOB en línea de 4 bytes. Los datos LOB en línea comienzan tras esta cabecera de 4 bytes.

Si el primer byte es 0x80, los datos LOB se encuentran en una serie vacía.

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: *nnnnnn...s*. La longitud del campo es: $(\text{precisión} + 2)/2$. La parte del signo (s) es xC para positivo (+) y es 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.

Estructura del registro descriptor de tabla

tipo de registro

número de columnas

estructura de columnas

- tipo de campo
- longitud
- indicador de nulos
- desplazamiento de campo

Información de LOB

Nota: Para obtener más información, consulte la descripción que hay después de la Tabla 116 en la página 747.

Registros de anotaciones del gestor de datos

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.

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 129 en la página 755.

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.

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 129 en la página 755.

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 cronológicas se graban cuando la tabla es una tabla de clúster multidimensional (tabla MDC). El registro de anotaciones cronológicas Insertar registro en página vacía se graba 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 cronológicas Suprimir registro en página vacía se graba 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 131. Retrotraer inserción de registro en página vacía.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	DMSLogRecordHeader	0(6)
lrIUDflags	unsigned short	6(2)
Longitud de registro	unsigned short	8(2)
Espacio libre	unsigned short	10(2)
RID	char[]	12(6)

Registros de anotaciones del gestor de datos

Tabla 131. Retrotraer inserción de registro en página vacía (continuación).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Interno	Interno	18(2)
Primera página del bloque	sqluint32	20(4)
<i>Longitud total: 24 bytes</i>		

Tabla 132. Insertar registro en página vacía.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	DMSLogRecordHeader	0(6)
lrIUDflags	unsigned short	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	variable	26(variable)
<i>Longitud total: 26 bytes más longitud de registro</i>		

Nota: Para obtener información detallada sobre la cabecera y los datos del registro, consulte la descripción que hay después de la Tabla 129 en la página 755.

Tabla 133. Retrotraer supresión de registro a página vacía y suprimir registro en página vacía.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	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	variable	26(variable)
<i>Longitud total: 26 bytes más longitud de registro</i>		

Registro de anotaciones de actualizar registro

El registro de anotaciones cronológicas de actualizar registro se graba 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, eliminar registro, retrotraer supresión de registro, retrotraer actualización de registro” en la página 754). 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 134. Estructura del registro de anotaciones de actualizar registro.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	DMSLogRecordHeader	0(6)
lrIUDflags	unsigned short	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	variable	20(variable)
Cabecera de anotaciones cronológicas	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	variable	variable(variable)
<i>Longitud total: 40 bytes más 2 longitudes de registro</i>		

Registro de anotaciones de renombramiento de una tabla o un esquema

El registro de anotaciones cronológicas de renombramiento de una tabla o un esquema se graba cuando se modifica un nombre de tabla o de esquema. Es el ID de función 122.

Registros de anotaciones del gestor de datos

Tabla 135. Estructura de las anotaciones cronológicas de renombramiento de una tabla o un esquema.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	DMSLogRecordHeader	0(6)
<i>Longitud total: 6 bytes</i>		

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 cronológicas de deshacer renombramiento de una tabla o un esquema

El registro de anotaciones cronológicas de deshacer renombramiento de una tabla o un esquema se graba cuando se retrotrae una modificación de nombre de una tabla o un esquema. Es el ID de función 123.

Tabla 136. Estructura del registro de anotaciones cronológicas de deshacer renombramiento de una tabla o un esquema.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	DMSLogRecordHeader	0(6)
<i>Longitud total: 6 bytes</i>		

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.

Registros de anotaciones del gestor de datos

Tabla 137. Inserción de varias estructuras de registro.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	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	variable	20(variable)
Consulte Tabla 138 y Tabla 139.		
<i>Longitud total: 20 bytes más longitud de registro</i>		

Tabla 138. Descripciones de registro (uno para cada registro).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
RID	unsigned char[6]	0(6)
Desplazamiento de registro	unsigned short	6(2)
Cabecera y datos de registro	variable	8(variable)
<i>Longitud total: 8 bytes más longitud de registro</i>		

Tabla 139. Descripciones de retrotracción (una para cada registro).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
RID	unsigned char[6]	0(6)
Desplazamiento de registro	unsigned short	6(2)
<i>Longitud total: 8 bytes</i>		

Para ver los detalles de la cabecera y los datos, consulte la descripción después de la Tabla 129 en la página 755.

Estado de partición de actualización, registros de anotaciones cronológicas del estado de partición de retrotracción

El registro de anotaciones cronológicas del estado de partición de actualización se graba cuando un usuario emite una sentencia ALTER TABLE con las cláusulas ADD PARTITION, ATTACH PARTITION o DETACH PARTITION. También se graba cuando la sentencia SET INTEGRITY se ejecuta en una tabla particionada para hacer que una partición anteriormente conectada esté en línea y sea visible. El ID de función del registro de anotaciones cronológicas es 137, mientras que el ID de función del registro de anotaciones de UNDO o ROLLBACK es 108.

Tabla 140. Estado de partición de actualización, estructura del registro de anotaciones cronológicas del estado de partición de retrotracción.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	DMSLogRecordHeader	0(6)
Identificador de espacio de tablas maestro	unsigned short	6(2)
Identificador de tabla maestra	unsigned short	8(2)
Interno	Interno	10(2)
Identificador de partición de datos	unsigned short	12(2)
Interno	Interno	14(6)
Interno	Interno	20(2)
Acción de partición	unsigned short	22(2)
<i>Longitud total: 24 bytes</i>		

Cabecera de anotaciones cronológicas

Consulte la Tabla 112 en la página 744. El espacio de tablas y los identificadores de tabla en DMSLogRecordHeader coinciden con los valores de las columnas TBSPACEID y PARTITIONOBJECTID en la vista de catálogo SYSCAT.DATAPARTITIONS para la partición de datos.

Identificador de espacio de tablas maestro

El identificador de espacio de tablas maestro coincide con el valor de la columna TBSPACEID en la vista de catálogo SYSCAT.TABLES para la tabla particionada.

Identificador de tabla maestra

El identificador de tabla maestra coincide con el valor de la columna TABLEID en la vista de catálogo SYSCAT.TABLES para la tabla particionada.

Identificador de partición de datos

El identificador de partición de datos coincide con el valor de la columna DATAPARTITIONID en la vista de catálogo SYSCAT.DATAPARTITIONS para la partición de tablas.

Acción de partición

Los valores de acción de partición tienen las definiciones siguientes:

Registros de anotaciones del gestor de datos

Tabla 141. Valores y definiciones de acciones de partición de datos:

Valor de la acción	Definición
1	ADD PARTITION
2	ATTACH PARTITION
4	SET INTEGRITY después de ATTACH PARTITION
5	DETACH PARTITION (diferido: las tablas de consulta materializada (MQT) deben mantenerse). ¹
6	DETACH PARTITION (diferido: los índices no particionados requieren limpieza). ¹
7	DETACH PARTITION (inmediato: la partición enlazada no había ejecutado nunca SET INTEGRITY). ¹
8	DETACH PARTITION (inmediato: no hay MQT ni índices no particionados). ¹
13	DETACH PARTITION (diferido: desenlazada lógicamente) ²
15	DETACH PARTITION (diferido: las tablas de consulta materializada (MQT) deben mantenerse). ²
16	DETACH PARTITION (diferido: la partición enlazada no había ejecutado nunca SET INTEGRITY). ²
18	DETACH PARTITION (diferido: se mantienen los elementos dependientes enlazados) ²
20	DETACH PARTITION (diferido: los índices no particionados requieren limpieza). ²
21	DETACH PARTITION (inmediato: DETACH realizado). ²

Nota:

1. A partir de DB2 Versión 9.7 Fixpack 1, el valor de la acción no se generará. El valor se muestra sólo por motivos de compatibilidad con las versiones anteriores.
2. El valor de la acción se define a partir de DB2 Versión 9.7 Fixpack 1.

Registros de anotaciones del gestor de relaciones

Los registros de anotaciones del gestor de relaciones facilitan la duplicación de las operaciones DDL en una base de datos de destino.

Tabla 142. Estructura de la cabecera del registro de anotaciones cronológicas del gestor de relaciones (RDSLogRecordHeader).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Identificador de componente (=11)	unsigned char	0(1)
Identificador de función (Vea la Tabla 143.)	unsigned char	1(1)
Identificadores de tabla		
Identif. espacio tablas	unsigned short	2(2)
Identificador de tabla	unsigned short	4(2)
rdsLogRecFlags	unsigned short	6(2)
<i>Longitud total: 8 bytes</i>		

Tabla 143. Valores y definiciones del identificador de función en la estructura de la cabecera del registro de anotaciones cronológicas del gestor de relaciones

Valor	Definición
4	Sentencia DDL
5	Deshacer sentencia DDL
6	Información de la partición

registro de anotaciones cronológicas de las sentencias DDL

Se guardan los registros de anotaciones de sentencias DDL de todas las operaciones de Lenguaje de definición de datos (DDL). Incluyen información de alto nivel de la DDL, como el tipo de operación (por ejemplo, CREATE, ALTER o DROP), así como el tipo de objeto (por ejemplo, TABLE, VIEW o INDEX). También existe información relacionada con el entorno de compilación donde se ejecuta la sentencia DDL, incluido el nombre del esquema por omisión, la vía de acceso de función y el id de autorización de la sentencia.

Los registros de anotaciones cronológicas de sentencias DDL contienen la sentencia de SQL real para la operación de DDL. Si la sentencia es demasiado larga para que encaje en un registro de anotaciones cronológicas, se dividirá en varios registros de anotaciones cronológicas, cada uno de los cuales incluye la longitud total de la sentencia y la longitud del texto restante.

El ID de función es 4.

Tabla 144. Estructura del registro de anotaciones cronológicas de las sentencias DDL

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	RDSLogRecordHeader	0(8)

Registros de anotaciones del gestor de relaciones

Tabla 144. Estructura del registro de anotaciones cronológicas de las sentencias DDL (continuación)

Descripción	Tipo	Desplazamiento (Bytes)
Identificador de operación (Vea la Tabla 145).	unsigned short	8(2)
Identificador de objeto (Vea la Tabla 146 en la página 768).	unsigned short	10(2)
Opciones	unsigned short	12(2)
NumEntries	unsigned short	14(2)
Texto restante de las sentencias DDL	sqluint32	variable(4)
Longitud del texto de las sentencias DDL	sqluint32	variable(4)
Entradas Cada entrada (Vea la Tabla 147 en la página 770)	DDLStmtRecordEntry	16(NumEntries * 4)
Texto de entrada Longitud de texto de cada entrada determinada por la longitud correspondiente en DDLStmtRecordEntry	char[]	variable(variable)
Texto de sentencia DDL	char[]	variable(variable)

Tabla 145. Valores y definiciones del identificador de operaciones de DDL

Valor	Definición
1	CREATE
2	DROP
3	ALTER
4	SET CONSTRAINTS
5	RENAME
6	COLLECT STATISTICS
7	ATTACH
8	DETACH
9	FLUSH
10	COMMENT
11	GRANT
12	REVOKE
13	LOCK
14	TRUNCATE
15	TRANSFER
16	RECREATE
17	SET
18	REFRESH

Registros de anotaciones del gestor de relaciones

Tabla 146. Valores y definiciones del identificador de objetos DDL

Valor	Definición
1	ALIAS
2	CHECK CONSTRAINT
3	COLUMN
4	DATABASE
5	FUNCTION
6	INDEX
7	MONITOR
8	PACKAGE
9	REFERENTIAL INTEGRITY CONSTRAINT
10	TABLE
11	TABLE SPACE
12	TRIGGER
13	TYPE
14	VIEW
15	CONSTRAINT
16	SCHEMA
17	PARTITION GROUP
18	BUFFERPOOL
19	PROCEDURE
20	DISTINCT TYPE
21	ABSTRACT DATA TYPE
22	INDEX EXTENSION
23	INDEX METHOD
24	NICKNAME
25	FUNCTION MAPPING
26	TYPE MAPPING
27	STORED PROCEDURE NICKNAME
28	SERVER MAPPING
29	USER MAPPING
30	SERVER OPTION
31	REVERSE TYPE MAPPING
32	WRAPPER
33	PASS THROUGH
34	COMPOUND
35	TABLE HIERARCHY
36	VIEW HIERARCHY
37	METHOD DEFINITION
38	COLLECT STATS
39	METHOD BODY
40	SEQUENCE

Registros de anotaciones del gestor de relaciones

Tabla 146. Valores y definiciones del identificador de objetos DDL (continuación)

Valor	Definición
41	XML CONTAINER
42	XML COLLECTION
43	XML COLLECTION VIEW
44	XML ITEM VIEW
45	XML INDEX
46	XSR OBJECT
47	SECURITY LABEL COMPONENT
48	SECURITY LABEL
49	SECURITY LABEL POLICY
50	ROLE
51	VARIABLE
52	WORKLOAD
53	SERVICE CLASS
54	WORK CLASS SET
55	WORK ACTION SET
56	THRESHOLD
57	HISTOGRAM TEMPLATE
58	TRUSTED CONTEXT
59	AUDIT POLICY
60	MODULE
61	MODULE BODY
62	PERMISSION
63	MASK
64	PROFILE CACHE
65	AUTHORITIES AND PRIVILEGES
66	EXEMPTION
67	SUMMARY TABLE
68	GLOBAL TEMP TABLE
69	CREATED TEMPORARY TABLE
70	INSERT ONLY TABLE
71	SUBTABLE
72	TABLE OF STRUCTURED TYPE
73	STRUCTURED TYPE
74	ROW TYPE
75	TABLE ALIAS
76	SEQUENCE ALIAS
77	MODULE ALIAS
78	PRIVATE ALIAS
79	PUBLIC ALIAS
80	EVENT MONITOR

Registros de anotaciones del gestor de relaciones

Tabla 146. Valores y definiciones del identificador de objetos DDL (continuación)

Valor	Definición
81	SERVER
82	TRANSFORM
83	STORAGE GROUP
84	USAGE LIST
85	FEDERATED CACHE

Tabla 147. Estructura DDLStmtRecordEntry

Descripción	Tipo	Desplazamiento (Bytes)
Tipo de entrada (Vea la Tabla 148)	unsigned short	0(2)
Longitud de entrada	unsigned short	2(2)

Tabla 148. Valores y definiciones del tipo de entrada

Valor	Definición
1	Nombre del esquema por omisión
2	Vía de acceso de la función
3	ID de autorización

Deshacer registro de anotaciones cronológicas de las sentencias DDL

El ID de función es 5.

Tabla 149. Deshacer la estructura del registro de anotaciones cronológicas de sentencias DDL

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	RDSLogRecordHeader	0(8)

Registro de anotaciones cronológicas de información de la partición

Los registros de anotaciones cronológicas de información de la partición se añaden cuando crea una tabla particionada, al igual que cuando añade, conecta o descarta una partición. Hay un registro de anotaciones cronológicas por cada partición. El ID de función es 6.

Tabla 150. Estructura del registro de anotaciones cronológicas de información de la partición.

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Cabecera de anotaciones cronológicas	RDSLogRecordHeader	0(8)

Registros de anotaciones del gestor de relaciones

Tabla 150. Estructura del registro de anotaciones cronológicas de información de la partición (continuación).

La fila inferior de esta tabla es una celda distribuida que contiene notas a pie de página de la tabla.

Descripción	Tipo	Desplazamiento (Bytes)
Acción de partición (Vea la Tabla 151)	unsigned short	8(2)
Distintivos de rango de partición (Vea la Tabla 152)	unsigned short	10(2)
ID de partición	unsigned short	12(2)
ID de espacio de tablas de datos	unsigned short	14(2)
ID de espacio de tablas largo	unsigned short	16(2)
ID de espacio de tablas de índice	unsigned short	18(2)
Longitud del nombre de partición	unsigned short	20(2)
Longitud del esquema de tabla	unsigned short	22(2)
Longitud del nombre de tabla	unsigned short	24(2)
Longitud de valor bajo	unsigned short	26(2)
Longitud de valor alto	unsigned short	28(2)
Nombre de partición	char[]	30(variable)
Esquema de tabla	char[]	variable(variable)
Nombre de tabla	char[]	variable(variable)
Valor bajo ¹	char[]	variable(variable)
Valor alto ¹	char[]	variable(variable)
¹ Los valores alto y bajo se registran como serie ASCII. Por ejemplo, aunque el tipo de columna sea entero y el valor sea 100, en el registro de anotaciones cronológicas, el valor será una serie de tres bytes "100"..		

Tabla 151. Valores y definiciones de la acción de partición

Valor	Definición
1	CREATE
2	ADD
3	ATTACH
4	DETACH

Tabla 152. Valores y definiciones de los distintivos de rango de partición

Valor	Definición
0x0001	Incluye el valor bajo
0x0002	Incluye el valor alto

Registros de anotaciones del gestor de relaciones

Apéndice C. Visión general de la información técnica de DB2

La información técnica de DB2 está disponible en diversos formatos a los que se puede acceder de varias maneras.

La información técnica de DB2 está disponible a través de las herramientas y los métodos siguientes:

- DB2Centro de información
 - Temas (Tareas, concepto y temas de consulta)
 - 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 la 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 conforme pasen a estar 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.

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/e-business/linkweb/publications/servlet/pbi.wss. Los manuales de DB2 Versión 10.1 en inglés y las versiones traducidas en formato PDF se pueden descargar del sitio web 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 153. Información técnica de DB2

Nombre	Número de documento	Copia impresa disponible	Última actualización
<i>Consulta de las API administrativas</i>	SC11-8067-00	Sí	Abril de 2012
<i>Rutinas y vistas administrativas</i>	SC11-8068-00	No	Abril de 2012
<i>Call Level Interface Guide and Reference Volume 1</i>	SC27-3866-00	Sí	Abril de 2012
<i>Call Level Interface Guide and Reference Volume 2</i>	SC27-3867-00	Sí	Abril de 2012
<i>Consulta de mandatos</i>	SC11-8069-00	Sí	Abril de 2012
<i>Database Administration Concepts and Configuration Reference</i>	SC27-3871-00	Sí	Abril de 2012
<i>Data Movement Utilities Guide and Reference</i>	SC27-3869-00	Sí	Abril de 2012
<i>Database Monitoring Guide and Reference</i>	SC27-3887-00	Sí	Abril de 2012
<i>Data Recovery and High Availability Guide and Reference</i>	SC27-3870-00	Sí	Abril de 2012
<i>Database Security Guide</i>	SC27-3872-00	Sí	Abril de 2012
<i>Guía y consulta de DB2 Workload Management</i>	SC11-8079-00	Sí	Abril de 2012
<i>Developing ADO.NET and OLE DB Applications</i>	SC27-3873-00	Sí	Abril de 2012
<i>Developing Embedded SQL Applications</i>	SC27-3874-00	Sí	Abril de 2012
<i>Desarrollo de aplicaciones Java</i>	SC11-8065-00	Sí	Abril de 2012

Tabla 153. Información técnica de DB2 (continuación)

Nombre	Número de documento	Copia impresa disponible	Última actualización
<i>Desarrollo de aplicaciones Perl, PHP, Python y Ruby on Rails</i>	SC11-8066-00	No	Abril de 2012
<i>Developing User-defined Routines (SQL and External)</i>	SC27-3877-00	Sí	Abril de 2012
<i>Getting Started with Database Application Development</i>	GI13-2046-00	Sí	Abril de 2012
<i>Iniciación a la instalación y administración de DB2 en Linux y Windows</i>	GI13-1946-00	Sí	Abril de 2012
<i>Globalization Guide</i>	SC27-3878-00	Sí	Abril de 2012
<i>Instalación de servidores DB2</i>	GC11-8073-00	Sí	Abril de 2012
<i>Instalación de clientes de IBM Data Server</i>	GC11-8074-00	No	Abril de 2012
<i>Consulta de mensajes Volumen 1</i>	SC11-8079-00	No	Abril de 2012
<i>Consulta de mensajes Volumen 2</i>	SC11-8080-00	No	Abril de 2012
<i>Net Search Extender Guía de administración y del usuario</i>	SC11-8082-00	No	Abril de 2012
<i>Partitioning and Clustering Guide</i>	SC27-3882-00	Sí	Abril de 2012
<i>pureXML Guide</i>	SC27-3892-00	Sí	Abril de 2012
<i>Spatial Extender Guía del usuario y manual de consulta</i>	SC11-8081-00	No	Abril de 2012
<i>SQL Procedural Languages: Application Enablement and Support</i>	SC27-3896-00	Sí	Abril de 2012
<i>Consulta de SQL - Volumen 1</i>	SC11-8070-00	Sí	Abril de 2012
<i>Consulta de SQL - Volumen 2</i>	SC11-8071-00	Sí	Abril de 2012
<i>Guía de Text Search</i>	SC11-8083-00	Sí	Abril de 2012
<i>Troubleshooting and Tuning Database Performance</i>	SC27-3889-00	Sí	Abril de 2012
<i>Actualización a DB2 Versión 10.1</i>	SC11-8072-00	Sí	Abril de 2012
<i>Novedades en DB2 Versión 10.1</i>	SC11-8078-00	Sí	Abril de 2012
<i>XQuery Reference</i>	SC27-3893-00	No	Abril de 2012

Tabla 154. Información técnica específica de DB2 Connect

Nombre	Número de documento	Copia impresa disponible	Última actualización
DB2 Connect Instalación y configuración de DB2 Connect Personal Edition	SC11-8075-00	Sí	Abril de 2012
DB2 Connect Instalación y configuración de servidores DB2 Connect	SC11-8076-00	Sí	Abril de 2012
Guía del usuario de DB2 Connect	SC11-8077-00	Sí	Abril de 2012

Visualización de la ayuda para estados de SQL desde el procesador de línea de mandatos

Los productos DB2 devuelven 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.

Procedimiento

Para iniciar 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

La documentación correspondiente a otras versiones de los productos DB2 se encuentra en otros centros de información en ibm.com.

Acerca de esta tarea

Para los temas de DB2 Versión 10.1, el URL del *Centro de información de DB2* es <http://publib.boulder.ibm.com/infocenter/db2luw/v10r1>.

Para los temas de DB2 Versión 9.8, el URL del *Centro de información de DB2* es <http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/>.

Para los temas de DB2 Versión 9.7, el URL del *Centro de información de DB2* es <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/>.

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 9.1, 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 DB2* en el sitio: <http://publib.boulder.ibm.com/infocenter/db2luw/v8/>.

Actualización del Centro de información de DB2 instalado en el sistema o en el servidor de intranet

El Centro de información de DB2 instalado en local se debe actualizar periódicamente.

Antes de empezar

Ya debe haber un Centro de información de DB2 Versión 10.1 instalado. Para obtener información adicional, consulte el tema “Instalación del Centro de información de DB2 utilizando el Asistente de instalación de DB2” en la publicación *Instalación de servidores DB2*. Todos los requisitos previos y las restricciones aplicables a la instalación del Centro de información se aplican también a la actualización del Centro de información.

Acercas de esta tarea

Un Centro de información de DB2 existente se puede actualizar automática o manualmente:

- Las actualizaciones automáticas actualizan las funciones y los idiomas del Centro de información existentes. Una ventaja de las actualizaciones automáticas es que el Centro de información deja de estar disponible durante un período de tiempo más breve a cuando se realiza la actualización manual. Además, la ejecución de las actualizaciones automáticas se puede configurar como parte de otros trabajos de proceso por lotes que se ejecutan periódicamente.
- Las actualizaciones manuales se pueden utilizar para actualizar las funciones y los idiomas existentes del Centro de información. Las actualizaciones automáticas reducen el tiempo de inactividad durante el proceso de actualización. Sin embargo, debe utilizar el proceso manual cuando desee añadir funciones o idiomas. Por ejemplo, un Centro de información en local se instaló inicialmente tanto en inglés como en francés, y ahora se desea instalar el idioma alemán. Con la actualización manual, se instalará el alemán y se actualizarán además las funciones y los idiomas existentes del Centro de información. No obstante, la actualización manual requiere que el usuario detenga, actualice y reinicie manualmente el Centro de información. El Centro de información no está disponible durante todo el proceso de actualización. En el proceso de actualización automática, el Centro de información incurre en una interrupción de servicio para reiniciar el Centro de información solo después de la actualización.

Este tema detalla el proceso de las actualizaciones automáticas. Para conocer las instrucciones para la actualización manual, consulte el tema “Actualización manual del Centro de información de DB2 instalado en el sistema o en el servidor de intranet”.

Procedimiento

Para actualizar automáticamente el Centro de información de DB2 instalado en el sistema o en el servidor de Intranet:

1. En sistemas operativos Linux,

Actualización del Centro de información de DB2 instalado en el sistema o en el servidor de intranet

- a. Navegue hasta la vía de acceso en la que está instalado el Centro de información. Por omisión, el Centro de información de DB2 se instala en el directorio `/opt/ibm/db2ic/V10.1`.
- b. Navegue desde el directorio de instalación al directorio `doc/bin`.
- c. Ejecute el script `update-ic`:
`update-ic`
2. En sistemas operativos Windows,
 - a. Abra una ventana de mandatos.
 - b. Navegue hasta la vía de acceso en la que está instalado el Centro de información. Por omisión, el Centro de información de DB2 se instala en el directorio `<Archivos de programa>\IBM\DB2 Information Center\Versión 10.1`, siendo `<Archivos de programa>` la ubicación del directorio Archivos de programa.
 - c. Navegue desde el directorio de instalación al directorio `doc\bin`.
 - d. Ejecute el archivo `update-ic.bat`:
`update-ic.bat`

Resultados

El Centro de información de DB2 se reinicia automáticamente. Si hay actualizaciones disponibles, el Centro de información muestra los temas nuevos y actualizados. Si no había actualizaciones del Centro de información disponibles, se añade un mensaje al archivo de anotaciones cronológicas. El archivo de anotaciones cronológicas está ubicado en el directorio `doc\eclipse\configuration`. El nombre del archivo de anotaciones cronológicas es un número generado aleatoriamente. Por ejemplo, `1239053440785.log`.

Actualización manual 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 localmente, puede obtener e instalar actualizaciones de la documentación de IBM.

Acerca de esta tarea

Para actualizar manualmente 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. La versión de estación de trabajo del Centro de información de DB2 siempre se ejecuta en modalidad autónoma.
2. Utilice la función Actualizar para ver qué actualizaciones están disponibles. Si hay actualizaciones que debe instalar, puede utilizar la función Actualizar para obtenerlas y actualizarlas.

Nota: Si su entorno requiere la instalación de actualizaciones del *Centro de información de DB2* en una máquina no conectada a Internet, duplique 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 2008 y Windows Vista (y posterior), los mandatos listados más abajo deben ejecutarse como administrador. Para abrir 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**.

Procedimiento

Para actualizar el *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/db2icdv10 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. Por omisión, el *Centro de información de DB2* se instala en el directorio `Archivos_de_programa\IBM\DB2 Information Center\Versión 10.1`, siendo `Archivos_de_programa` 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. Por omisión, el *Centro de información de DB2* se instala en el directorio `/opt/ibm/db2ic/V10.1`.
 - b. Navegue desde el directorio de instalación al directorio `doc/bin`.
 - c. Ejecute el script `help_start`:
`help_start`

Se abre el navegador Web por omisión de los sistemas para visualizar el Centro de información autónomo.
3. Pulse en el botón **Actualizar** (🔄). (JavaScript debe estar habilitado en el navegador.) 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**.

Actualización manual del Centro de información de DB2 instalado en el sistema o en el servidor de intranet

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 detener 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 detener `help_start.bat`.
 - En Linux, navegue hasta el directorio `doc/bin` del directorio de instalación y ejecute el script `help_end`:
`help_end`

Nota: El script `help_end` contiene los mandatos necesarios para detener sin peligro los procesos que se iniciaron mediante el script `help_start`. No utilice ningún otro método para detener 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/db2icdv10 start`

Resultados

El *Centro de información de DB2* actualizado muestra los temas nuevos y actualizados.

Guías de aprendizaje de DB2

Las guías de aprendizaje de DB2 le ayudan a conocer diversos aspectos de productos de base de datos 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/db2luw/v10r1/>.

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.

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 *Troubleshooting and Tuning Database Performance* o en la sección sobre conceptos fundamentales sobre bases de datos del *Centro de información de DB2*, que contiene:

- Información sobre cómo aislar e identificar problemas con programas de utilidad y herramientas de diagnóstico de DB2.
- Soluciones a algunos de los problemas más comunes.
- Consejo para ayudarlo a resolver problemas que podría encontrar en los productos de base de datos DB2

Portal de Soporte de IBM

Consulte el portal de soporte de IBM 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 portal de Soporte de IBM en el sitio http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows

Términos y condiciones

Los permisos para utilizar estas publicaciones se otorgan sujetos a los siguientes términos y condiciones.

Aplicación: Además de las condiciones de uso del sitio web de IBM, se aplican estos 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 derivados de estas publicaciones, ni reproducirlas, distribuirlas o visualizarlas, ni de partes de las mismas fuera de su empresa, sin el consentimiento expreso de IBM.

Derechos: 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.

Términos y condiciones

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.

Marcas registradas de IBM: 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 web en www.ibm.com/legal/copytrade.shtml.

Apéndice D. Avisos

Esta información ha sido desarrollada para productos y servicios que se ofrecen en Estados Unidos de América. La información acerca de productos que no son IBM se basa en la información disponible cuando se publicó este documento por primera vez y está sujeta a cambio.

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 juegos de caracteres de doble byte (DBCS), puede ponerse en contacto con el Departamento de Propiedad Intelectual de IBM de su país o escribir a:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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

Avisos

efectuar, en cualquier momento y sin previo aviso, mejoras y cambios en los productos y programas descritos en esta publicación.

Las referencias hechas en esta publicación a sitios web que no son de IBM se proporcionan sólo para la comodidad del usuario y no constituyen un aval de esos sitios web. La información de esos sitios web no forma parte de la información del presente producto de IBM y la utilización de esos sitios web se realiza bajo la responsabilidad del usuario.

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
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
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 contiene 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. Los programas de ejemplo se proporcionan "TAL CUAL", sin ningún tipo de garantía. IBM no se hará responsable de los daños derivados de la utilización que haga el usuario de los programas de ejemplo.

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. La lista actual de marcas registradas de IBM está disponible en la web, en "Copyright and trademark information", en la dirección 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 todos los logotipos y marcas registradas basadas en Java son marcas registradas de Oracle, sus filiales o ambos.
- 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, Celeron, Intel SpeedStep, Itanium y Pentium son marcas registradas de Intel Corporation o de sus empresas subsidiarias en Estados Unidos y en otros países.
- Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en los Estados Unidos y/o en otros países.

Otros nombres de empresas, productos o servicios, pueden ser marcas registradas o marcas de servicio de otras empresas.

Índice

A

- actualizaciones
 - Centro de información de DB2 777, 778
- actualizar base de datos, API 85
- actualizar registro, registro de anotaciones 744
- anotaciones cronológicas
 - recuperación 404
- anyorder, modificador de tipo de archivo 205
- Añadir partición de base de datos, API 387
- API
 - Cambiar nivel de aislamiento (REXX) 487
 - Comprimir 623
 - db2AddContact 38
 - db2AddContactGroup 40
 - db2AddSnapshotRequest 42
 - db2AdminMsgWrite 44
 - db2ArchiveLog 46
 - db2AutoConfig 49
 - db2AutoConfigFreeMemory 53
 - db2Backup 54
 - db2CfgGet 65
 - db2CfgSet 68
 - db2ConvMonStream 73
 - db2DatabasePing 76
 - db2DatabaseQuiesce 78
 - db2DatabaseRestart 80
 - db2DatabaseUnquiesce 83
 - db2DropContact 94
 - db2DropContactGroup 95
 - db2Export 96
 - db2GetAlertCfg 103
 - db2GetAlertCfgFree 108
 - db2GetContactGroup 109
 - db2GetContactGroups 111
 - db2GetContacts 113
 - db2GetDistMap 115
 - db2GetHealthNotificationList 117
 - db2GetRecommendations 119
 - db2GetRecommendationsFree 122
 - db2GetSnapshot 123
 - db2GetSnapshotSize 127
 - db2GetSyncSession 130
 - db2HADRStart 131
 - db2HADRStop 133
 - db2HADRTakeover 135
 - db2HistoryCloseScan 138
 - db2HistoryGetEntry 139
 - db2HistoryOpenScan 141
 - db2HistoryUpdate 145
 - db2Import 149
 - db2Inspect 164
 - db2InstanceQuiesce 172
 - db2InstanceStart 175
 - db2InstanceStop 181
 - db2InstanceUnquiesce 186
 - db2LdapCatalogDatabase 188
 - db2LdapCatalogNode 191
 - db2LdapDeregister 193
 - db2LdapRegister 194
 - db2LdapUncatalogDatabase 198
 - db2LdapUncatalogNode 199
- API (continuación)
 - db2LdapUpdate 200
 - db2LdapUpdateAlternateServerForDB 203
 - db2Load 205
 - db2LoadQuery 228
 - db2MonitorSwitches 236
 - db2Prune 239
 - db2QuerySatelliteProgress 242
 - db2ReadLog 244
 - db2ReadLogNoConn 251
 - db2ReadLogNoConnInit 255
 - db2ReadLogNoConnTerm 258
 - db2Recover 259
 - db2Reorg 265
 - db2ResetAlertCfg 275
 - db2ResetMonitor 277
 - db2Restore 280
 - db2Rollforward 296
 - db2Runstats 307
 - db2SelectDB2Copy 320
 - db2SetSyncSession 322
 - db2SetWriteForDB 323
 - db2SpmListIndTrans 325
 - db2SyncSatellite 329
 - db2SyncSatelliteStop 330
 - db2SyncSatelliteTest 331
 - db2UpdateAlertCfg 332
 - db2UpdateAlternateServerForDB 338
 - db2UpdateContact 340
 - db2UpdateContactGroup 342
 - db2UpdateHealthNotificationList 344
 - db2UtilityControl 346
 - db2VendorGetNextObj 589
 - db2VendorQueryApiVersion 592
 - db2XaGetInfo 491
 - db2XaListIndTrans 492
 - Decompress 625
 - GetMaxCompressedSize 627
 - GetSavedBlock 628
 - heurísticas 489
 - InitCompression 629
 - InitDecompression 630
 - nivel anterior 23
 - personalización del precompilador 720
 - plugin de ID de usuario/contraseña 556
 - plugin de recuperación de grupos 544
 - plugin de seguridad
 - db2secClientAuthPluginInit 562
 - db2secClientAuthPluginTerm 564
 - db2secDoesAuthIDExist 565
 - db2secDoesGroupExist 546
 - db2secFreeErrorMsg 547
 - db2secFreeGroupListMemory 548
 - db2secFreeInitInfo 566
 - db2secFreeToken 567
 - db2secGenerateInitialCred 568
 - db2secGetAuthIDs 570
 - db2secGetDefaultLoginContext 572
 - db2secGetGroupsForUser 549
 - db2secGroupPluginInit 553
 - db2secPluginTerm 555

API (continuación)

plugin de seguridad (continuación)
 db2secProcessServerPrincipalName 574
 db2secRemapUserid 575
 db2secServerAuthPluginInit 577
 db2secServerAuthPluginTerm 580
 db2secValidatePassword 581
 visión general 541
 resumen 1
 sintaxis de REXX 485
 sqlabndx 348
 sqlaintp 351
 sqlaprep 353
 sqlarwnd 356
 sqlbctcq 359
 sqlbctsq 360
 sqlbftcq 361
 sqlbftpq 363
 sqlbgtss 365
 sqlbmtsq 367
 sqlbotcq 369
 sqlbotsq 371
 sqlbstpq 373
 sqlbstsc 375
 sqlbtcq 378
 sqlcspqy 380
 sqle_activate_db 381
 sqle_deactivate_db 384
 sqleaddn 387
 sqleatcp 390
 sqleatin 393
 sqleAttachToCtx 502
 sqleBeginCtx 503
 sqlecadb 396
 sqlecran 402
 sqlecrea 404
 sqlectnd 412
 sqledcgd 415
 sqledcls 87
 sqleDetachFromCtx 504
 sqledgne 88
 sqledosd 92
 sqledpan 418
 sqledrpd 420
 sqledrpn 422
 sqledtin 424
 sqleEndCtx 505
 sqlefmem 425
 sqlefrce 426
 sqlegdad 429
 sqlegdcl 431
 sqlegdel 432
 sqlegdge 434
 sqlegdgt 436
 sqlegdsc 438
 sqleGetCurrentCtx 507
 sqlegins 439
 sqleInterruptCtx 508
 sqleintr 440
 sqleisig 442
 sqlemgdb 443
 sqlencls 445
 sqlengne 446
 sqlenops 448
 sqleqryc 450
 sqleqryi 452
 sqlesact 454

API (continuación)

sqlesdeg 455
 sqlesetc 457
 sqleseti
 detalles 460
 sqleSetTypeCtx 509
 sqleuncd 463
 sqleuncn 465
 sqlgaddr 467
 sqlgdref 468
 sqlgmcpy 469
 sqlogstt 470
 sqludrtd 472
 sqluexpr 96
 sqlugrpn 476
 sqlugtpi 479
 sqluimpr 149
 sqluvdel 593
 sqluvend 594
 sqluvget 596
 sqluvint 598
 sqluvput 602
 sqluvqdp 481
 sqlxhfrg 497
 sqlxphcm 498
 sqlxphrl 499
 TermCompression 631
 TermDecompression 632
 API, cargar 205
 API abrir consulta de contenedor de espacio de tablas 369
 API abrir consulta de espacio de tablas 371
 API actualizar configuración de alerta 332
 API actualizar contacto 340
 API actualizar grupo de contactos 342
 API actualizar lista de notificaciones de salud 344
 API Actualizar servidor alternativo para base de datos 338
 API Añadir contacto 38
 API añadir grupo de contactos 40
 API archivar anotaciones cronológicas activas 46
 API captar consulta de contenedor de espacio de tablas 361
 API cargar consulta 228
 API catalogar base de datos 396
 API catalogar base de datos DCS 429
 API catalogar entrada LDAP de base de datos 188
 API catalogar entrada LDAP de nodo 191
 API catalogar nodo 412
 API Cerrar consulta de contenedor de espacio de tablas 359
 API cerrar consulta de espacios de tablas 360
 API cerrar exploración de directorio de bases de datos 87
 API cerrar exploración de directorio de DCS 431
 API cerrar exploración de directorio de nodos 445
 API comprimir un bloque de datos 623
 API confirmar transacción dudosa 498
 API consulta de contenedor de espacio de tablas 378
 API consulta de espacio de tablas 367
 API consultar cliente 450
 API consultar información de cliente 452
 API consultar sincronización de satélite 242
 API control de programa de utilidad 346
 API convertir corriente de supervisor 73
 API copiar memoria 469
 API crear base de datos en nodo 402
 API crear y conectar a contexto de aplicación 503
 API db2AddContact 38
 API db2AddContactGroup 40
 API db2AdminMsgWrite 44
 API db2ArchiveLog 46

API db2AutoConfig 49
 API db2AutoConfigFreeMemory 53
 API db2CfgGet 65
 API db2CfgSet 68
 API db2ConvMonStream 73
 API db2DatabasePing 76
 API db2DatabaseQuiesce 78
 API db2DatabaseRestart 80
 API db2DatabaseUnquiesce 83
 API db2DropContact 94
 API db2DropContactGroup 95
 API db2GetAlertCfg 103
 API db2GetAlertCfgFree 108
 API db2GetContactGroup 109
 API db2GetContactGroups 111
 API db2GetContacts 113
 API db2GetHealthNotificationList 117
 API db2GetRecommendations 119
 API db2GetRecommendationsFree 122
 API db2GetSyncSession 130
 API db2HADRStart 131
 API db2HADRTStop 133
 API db2HADRTakeover 135
 API db2HistoryCloseScan 138
 API db2HistoryGetEntry 139
 API db2InstanceQuiesce 172
 API db2InstanceStart 175
 API db2InstanceStop 181
 API db2InstanceUnquiesce 186
 API db2LdapCatalogDatabase 188
 API db2LdapUncatalogDatabase 198
 API db2LdapUncatalogNode 199
 API db2LdapUpdate 200
 API db2LdapUpdateAlternateServerForDB 203
 API db2LoadQuery 228
 API db2QuerySatelliteProgress 242
 API db2ReadLog 244
 API db2ReadLogNoConn 251
 API db2ReadLogNoConnInit 255
 API db2ReadLogNoConnTerm 258
 API db2Recover
 detalles 259
 API db2Reorg 265
 API db2ResetAlertCfg 275
 API db2Rollforward
 detalles 296
 API db2SelectDB2Copy
 detalles 320
 API db2SetSyncSession 322
 API db2SetWriteForDB 323
 API db2SyncSatellite 329
 API db2SyncSatelliteStop 330
 API db2SyncSatelliteTest 331
 API db2UpdateAlertCfg 332
 API db2UpdateAlternateServerForDB 338
 API db2UpdateContact 340
 API db2UpdateContactGroup 342
 API db2UpdateHealthNotificationList 344
 API db2UtilityControl 346
 API db2VendorGetNextObj 589
 API db2VendorQueryApiVersion 592
 API db2XaGetInfo 491
 API db2XaListFndTrans 492
 API de base de datos ping 76
 API de copia de seguridad de base de datos
 detalles 54
 API de detención de instancia 181
 API de exportación 96
 API de importación 149
 API de inicio de instancia 175
 API de lectura de anotación cronológica asíncrona 244
 API de movilización de instancia 186
 API de REXX cambiar nivel de aislamiento 487
 API de vinculación
 sqlabndx 348
 API descatalogar base de datos 463
 API descatalogar base de datos DCS 432
 API descatalogar entrada LDAP de base de datos 198
 API descatalogar entrada LDAP de nodo 199
 API descatalogar nodo 465
 API desenlazar de una memoria libre asociada con contexto de aplicación 505
 API desenlazar el dispositivo y liberar sus recursos 594
 API detener HADR 133
 API detener sincronización de satélite 330
 API enlazar 393
 API enlazar y cambiar contraseña 390
 API establecer cliente 457
 API establecer contenedores de espacios de tablas 375
 API establecer grado de ejecución 455
 API establecer información de cliente 460
 API establecer parámetros de configuración 68
 API establecer serie de contabilidad 454
 API establecer sesión de sincronización de satélites 322
 API establecer tipo de contexto de aplicación 509
 API grabar datos en un dispositivo de proveedor 602
 API inicializar la biblioteca de compresión 629
 API inicializar la biblioteca de descompresión 630
 API inicializar lectura de anotación cronológica sin una conexión de base de datos 255
 API inicializar y enlazar con archivo 598
 API iniciar exploración de directorio DCS 438
 API iniciar exploración de directorio de base de datos 92
 API iniciar exploración de directorio de nodos 448
 API iniciar HADR 131
 API inmovilizar espacios de tablas para tabla 481
 API inmovilizar instancia 172
 API inspeccionar base de datos 164
 API instalar manejador de señales 442
 API interrumpir 440
 API interrumpir contexto 508
 API leer anotación cronológica sin una conexión de base de datos 251
 API leer datos de dispositivo 596
 API liberar memoria asignada por las API sqlbctq y sqlbmtsq 425
 API liberar memoria de db2GetRecommendations 122
 API liberar obtención de la configuración de alerta 108
 API listar transacciones dudosas 492
 API listar transacciones dudosas de DRDA 380
 API listar transacciones dudosas de SPM
 detalles 325
 API obtener configuración de alerta 103
 API obtener contactos 113
 API obtener contexto actual 507
 API obtener dirección 467
 API obtener entrada de directorio de base de datos siguiente 88
 API obtener entrada de directorio de DCS para base de datos 434
 API obtener entrada de directorio de nodo siguiente 446
 API obtener entradas de directorio de DCS 436
 API obtener estadísticas de espacio de tablas 365
 API obtener grupo de contactos 109

API obtener grupos de contactos 111
 API obtener información para gestor de recursos 491
 API obtener instancia 439
 API obtener instantánea 123
 API obtener lista de notificaciones de salud 117
 API obtener mensaje de SQLSTATE 470
 API obtener número de distribución de filas 476
 API obtener o actualizar conmutaciones de supervisor 236
 API obtener parámetros de configuración 65
 API obtener recomendaciones para un indicador de salud en estado de alerta 119
 API obtener sesión de sincronización de satélites 130
 API olvidar estado de la transacción 497
 API para activar base de datos 381
 API para actualizar los registros del histórico de la base de datos 145
 API para desactivar base de datos 384
 API para descartar base de datos 420
 API para descartar contacto 94
 API para descartar grupo de contactos 95
 API para descomprimir un bloque de datos 625
 API para desconectar de contexto 504
 API para desconectar de instancia 424
 API para eliminar archivos históricos 239
 API para eliminar la referencia de una dirección 468
 API para inmovilizar la base de datos 78
 API para movilizar base de datos 83
 API para obtener los registros siguientes del histórico de la base de datos 139
 API para precompilar programa de aplicaciones 353
 API para suprimir sesión confirmada 593
 API probar sincronización de satélites 331
 API recuperar base de datos 259
 API redistribuir grupo de particiones de base de datos 472
 API reiniciar base de datos 80
 API reorganizar 265
 API restablecer configuración de alerta 275
 API restablecer supervisor 277
 API restaurar base de datos 280
 API runstats 307
 API sincronizar satélite 329
 API sqlabndx 348
 API sqlaintp 351
 API sqlaprep 353
 API sqlarbnd 356
 API sqlbctcq 359
 API sqlbctsq 360
 API sqlbftcq 361
 API sqlbftpq 363
 API sqlbgtss 365
 API sqlbmtsq 367
 API sqlbotcq 369
 API sqlbotsq 371
 API sqlbstpq 373
 API sqlbtcq 378
 API sqlcspqy 380
 API SQLDB2
 llamada a las API de DB2 485
 API sqleaddn 387
 API sqleatin 393
 API sqleAttachToCtx
 detalles 502
 API sqleBeginCtx
 detalles 503
 API sqlecadb 396
 API sqlecran 402
 API sqlecrea 404
 API sqlectnd 412
 API sqledcgd 415
 API sqledcls 87
 API sqleDetachFromCtx
 detalles 504
 API sqledosd 92
 API sqledpan 418
 API sqledrpd 420
 API sqledrpn 422
 API sqledtin 424
 API sqleEndCtx
 detalles 505
 API sqlefmem 425
 API sqlefrce 426
 API sqlegdad 429
 API sqlegdcl 431
 API sqlegdel 432
 API sqlegdge 434
 API sqlegdgt 436
 API sqlegdsc 438
 API sqleGetCurrentCtx
 detalles 507
 API sqlegins 439
 API sqleInterruptCtx
 detalles 508
 API sqleintr 440
 API sqleisig 442
 API sqlemgdb
 detalles 443
 API sqlencls 445
 API sqlenops 448
 API sqleqryi 452
 API sqlesact 454
 API sqlesdeg 455
 API sqleseti
 detalles 460
 API sqleSetTypeCtx
 detalles 509
 API sqleuncd 463
 API sqleuncn 465
 API sqlgaddr 467
 API sqlgdref 468
 API sqlgmcpy 469
 API sqlgstt 470
 API sqluexpr 96
 API sqlugrpn
 detalles 476
 API sqlugtpi
 detalles 479
 API sqluvdel 593
 API sqluvend 594
 API sqluvget 596
 API sqluvput 602
 API sqluvqdp 481
 API sqlxhfrg 497
 API sqlxphcm 498
 API sqlxphrl 499
 API terminar la lectura de las anotaciones cronológicas sin una conexión de base de datos 258
 API tomar el control como base de datos primaria 135
 API volver a vincular 356
 aplicaciones
 acceso mediante gestor de bases de datos 348
 archivo de configuración de base de datos
 entradas válidas 692
 archivos de inclusión
 visión general 35

- atributo de alterar columna, registro de anotaciones 744
- authentication
 - de dos componentes 537
 - GSS-API 530
 - ID y contraseña 530
 - Kerberos 530
 - plugins
 - API de plugins de autenticación por ID de usuario/contraseña 556
 - API para comprobar si existe ID de autenticación 565
 - API para inicializar plugin de autenticación de cliente 562
 - API para inicializar plugin de autenticación de servidor 577
 - API para liberar los recursos de plugin de autenticación de cliente 564
 - API para liberar los recursos de plugin de autenticación de servidor 580
 - API para liberar los recursos retenidos por la API db2secGenerateInitialCred 566
 - API para obtener los ID de autenticación 570
 - API para validar contraseñas 581
 - desplegar 512, 515
 - seguridad 530
 - ubicaciones de bibliotecas 535
 - plugins de seguridad 530
- autoconfigure, API 49
- avisos 783
- ayuda
 - sentencias SQL 776

B

- bases de datos
 - aislar datos 487
 - creación
 - API sqlecrea 404
 - descartar
 - API sqledrpd 420
 - exportación de tabla a archivo
 - API db2Export 96
 - gestión 511
 - importación de archivo a tabla
 - API db2Import 149
 - proceso de peticiones concurrentes 487
 - suprimir
 - API sqledrpd 420
 - vinculación de programas de aplicación 348
- bases de datos particionadas
 - información de distribución de tablas 115, 479
- bibliotecas
 - plugins de seguridad
 - carga en DB2 516
 - restricciones 518
- binarynumerics, modificador de tipo de archivo
 - db2load, API 205
- bloque ampliado de descripción de base de datos 681
- bloques
 - cambiar 487

C

- cancelación heurística, registro de anotaciones 728
- cancelación normal, registro de anotaciones 728
- captar consulta de espacios de tablas, API 363
- cargar lista pendiente, registro de anotaciones 740

- Centro de información de DB2
 - actualización 777, 778
 - versiones 776
- cerrar exploración de registros de histórico de base de datos, API 138
- chardel, modificador de tipo de archivo
 - API db2Export 96
 - API db2Import 149
 - db2Load, API 205
- COBOL, lenguaje
 - aplicaciones
 - archivos de inclusión 35
 - manejo de punteros 467, 468, 469
- códigos de retorno
 - API 31
- coldel, modificador de tipo de archivo
 - API db2Export 96
 - API db2Import 149
 - db2Load, API 205
- columnas
 - importación 149
- comentarios
 - base de datos 415
- cómo esta estructurado este manual ix
- compound, modificador de tipo de archivo
 - API db2Import 149
- compresión
 - API detener la biblioteca de compresión 631
 - API detener la biblioteca de descompresión 632
 - interfaz de plug-in 511, 615
- confirmación heurística, registro de anotaciones 728
- confirmación normal, registro de anotaciones 728
- consulta de espacio de tablas individual, API 373
- contraseñas
 - cambiar
 - sqleatcp, API 390
- convenios de resaltado x
- copias de seguridad
 - final de registro de anotaciones 740
- crear API de base de datos 404
- crear índice, registro de anotaciones 744
- crear tabla, registro de anotaciones 744

D

- dateformat, modificador de tipo de archivo
 - API db2Import 149
 - db2Load, API 205
- DB2 Connect
 - conexiones soportadas 429
- DB2-INFO, estructura 604
- db2Backup, API
 - detalles 54
- db2DatabaseUpgrade, API 85
- db2DistMapStruct, estructura 634
- db2GetDistMap, API 115
- db2GetSnapshot, API
 - cálculo del tamaño del almacenamiento intermedio de salida 127
 - detalles 123
- db2GetSnapshotSize, API 127
- db2HistData, estructura 635
- db2HistoryOpenScan, API 141
- db2HistoryUpdate, API 145
- db2Inspect, API 164
- db2LdapCatalogNode, API 191
- db2LdapDeregister, API 193

- db2LdapRegister, API 194
- db2Load, API 205
- db2LSN, estructura 641
- db2LSN, tipo de datos 27
- db2MonitorSwitches, API 236
- db2Prune, API 239
- db2ResetMonitor, API 277
- db2Restore, API
 - detalles 280
- db2Runstats, API 307
- depurar
 - plugins de seguridad 540
- descartar base de datos en la API de servidor de particiones
 - de base de datos 418
- descartar índice, registro de anotaciones 744
- descatalogación
 - directorio de bases de datos del sistema 463
- deshacer acción de renombrar registro de anotaciones de esquema 744
- deshacer acción de renombrar registro de anotaciones de tabla 744
- deshacer registro de anotaciones cronológicas de las sentencias DDL 766
- determinación de problemas
 - guías de aprendizaje 781
 - información disponible 781
 - plugins de seguridad 540
- directorio DCS (servicios de conexión de bases de datos)
 - adición de entradas 429
 - catalogar entradas 429
 - copia de entradas 436
 - eliminación de entradas 432
 - inicio de exploración 438
 - recuperación de entradas 434
- directorio de base de datos local
 - inicio de exploración 92
 - recuperación de entradas 88
- directorio de bases de datos del sistema
 - adición de entradas 396
 - catalogar bases de datos 396
 - descatalogación 463
 - inicio de exploración 92
 - recuperación de entradas 88
 - supresión de entradas 463
- directorios
 - base de datos local
 - inicio de exploración 92
 - recuperación de entradas 88
- Database Connection Services (DCS)
 - adición de entradas 429
 - copia de entradas 436
 - inicio de exploración 438
 - recuperación de entradas 434
 - supresión de entradas 432
- de bases de datos del sistema
 - adición de entradas 396
 - catalogar bases de datos 396
 - descatalogación de bases de datos (mandato) 463
 - inicio de exploración 92
 - recuperación de entradas 88
 - supresión de entradas 463
- nodo
 - adición de entradas 412
 - detalles 465
 - recuperación de entradas 446
 - supresión de entradas 465

- directorios de bases de datos
 - recuperar entrada siguiente 88
- directorios de nodos
 - adición de entradas 412
 - recuperación de entradas 446
 - supresión de entradas 465
- diseño de aplicación
 - instalación del manejador de señales 442
 - manejo de punteros
 - copia de datos 469
 - desreferencia de dirección 468
 - obtención de la dirección de la variable 467
 - valor de secuencia de clasificación 404
- documentación
 - archivos PDF 774
 - copia impresa 774
 - términos y condiciones de uso 781
 - visión general 773
- DROP, sentencia
 - tablas
 - registro de anotaciones 744

E

- enlazar con contexto, API 502
- errores
 - avance 296
- espacios de tablas
 - final de recuperación ascendente a PIT, registro de anotaciones 740
 - inicio de recuperación ascendente a PIT, registro de anotaciones 740
 - recuperados, registros de anotaciones 740
- esquemas
 - nuevas bases de datos 404
- estructura COMPR_CB 618
- estructura COMPR_DB2INFO 619
- estructura COMPR_PIINFO 621
- estructura SQLCHAR
 - detalles 654
- estructura SQLEDBTERRITORYINFO 688
- estructura sqllob 701
- estructuras de datos
 - API de proveedor 588
 - COMPR_CB 618
 - COMPR_DB2INFO 619
 - COMPR_PIINFO 621
 - datos 613
 - DB2-INFO 604
 - db2DistMapStruct 634
 - db2HistData 635
 - db2LSN 641
 - INIT-OUTPUT 612
 - RETURN-CODE 614
 - SQL-DIR-ENTRY 642
 - SQLB-TBS-STATS 644
 - SQLB-TBSCONTQRY-DATA 646
 - SQLB-TBSPQRY-DATA 648
 - SQLCA 653
 - SQLCHAR 654
 - SQLDA 655
 - SQLDCOL 657
 - SQLE-ADDN-OPTIONS 660
 - SQLE-CLIENT-INFO 662
 - SQLE-CONN-SETTING 665
 - SQLE-NODE-LOCAL 668
 - SQLE-NODE-NPIPE 669

estructuras de datos (*continuación*)

- SQLE-NODE-STRUCT 670
- SQLE-NODE-TCPIP 671
- SQLEDBTERRITORYINFO 688
- SQLENINFO 689
- SQLETSDESC 672
- SQLFUPD 692
- sqllob 701
- SQLMA 702
- SQLOPT 706
- SQLU-LSN 708
- SQLU-MEDIA-LIST 709
- SQLU-RLOG-INFO 714
- SQLUPI 715
- SQLXA-XID 717
- VENDOR-INFO 608

exportación

- datos
 - API db2Export 96
 - modificadores de tipo de archivo 96

F

fastparse, modificador de tipo de archivo

- db2Load, API 205

forcein, modificador de tipo de archivo

- API db2Import 149
- db2Load, API 205

FORTTRAN, lenguaje

- aplicaciones
 - archivos de inclusión 35
 - manejo de punteros 467, 468, 469

forzar aplicación, API 426

funciones

- plugin de cliente

- API db2secClientAuthPluginInit 562
- API db2secClientAuthPluginTerm 564
- API db2secDoesAuthIDExist 565
- API db2secFreeInitInfo 566
- API db2secFreeToken 567
- API db2secGenerateInitialCred 568
- API db2secGetAuthIDs 570
- API db2secGetDefaultLoginContext 572
- API db2secProcessServerPrincipalName 574
- API db2secRemapUserid 575
- API db2secServerAuthPluginInit 577
- API db2secServerAuthPluginTerm 580
- API db2secValidatePassword 581

- plugin de grupo

- API db2secDoesGroupExist 546
- API db2secFreeErrorMsg 547
- API db2secFreeGroupListMemory 548
- API db2secGetGroupsForUser 549
- API db2secGroupPluginInit 553
- API db2secPluginTerm 555

G

generatedignore, modificador de tipo de archivo

- API db2Import 149
- db2load, API 205

generatedmissing, modificador de tipo de archivo

- API db2Import 149
- db2load, API 205

generatedoverride, modificador de tipo de archivo

- db2load, API 205

gestor de bases de datos

- registros de anotaciones 744

gestor de campos largos, registros de anotaciones 738

gestores de transacciones

- registros de anotaciones 728

GSS-API

- plugin de autenticación 584

guías de aprendizaje

- determinación de problemas 781

- lista 780

- pureXML 780

- resolución de problemas 781

I

ID de usuario

- de dos componentes 537

identificador de transacción, registros de anotaciones 726

identificadores de registro (RID)

- registros de anotaciones 726

identityignore, modificador de tipo de archivo

- API db2Import 149
- db2Load, API 205

identitymissing, modificador de tipo de archivo

- API db2Import 149
- db2Load, API 205

identityoverride, modificador de tipo de archivo

- db2Load, API 205

implieddecimal, modificador de tipo de archivo

- API db2Import 149
- db2Load, API 205

importación

- a base de datos remota 149

- a jerarquía que no existe 149

- a tabla que no existe 149

- a tablas con tipo 149

- acceso a base de datos mediante DB2 Connect 149

- archivo en tabla de base de datos 149

- archivos PC/IXF

- de varias partes 149

- consideraciones sobre página de códigos 149

- modificadores de tipo de archivo 149

- restricciones 149

indexfreespace, modificador de tipo de archivo

- API db2Import 205

indexixf, modificador de tipo de archivo

- API db2Import 149

indexschema, modificador de tipo de archivo

- API db2Import 149

inicializar tabla, registro de anotaciones 744

iniciar exploración de registros de histórico de base de datos,

- API 141

inicio de carga, registro de anotaciones

- detalles 740

INIT-INPUT, estructura 610

INIT-OUTPUT, estructura 612

insertar registro, registro de anotaciones 744

insertar registro en registro de anotaciones de página vacía 744

K

keepblanks, modificador de tipo de archivo

- API db2Import 149
- db2Load, API 205

L

LDAP
API actualizar servidor 200
API Actualizar servidor alternativo para base de datos 203
API desregistrar servidor 193
API registrar servidor 194
lenguaje C/C++
aplicaciones
archivos de inclusión 35
lenguaje REXX
API
llamada al CLP de DB2 485
sintaxis 485
liberar memoria de autoconfigure, API 53
libre de retrotracción, registro de anotaciones 728
lista pendiente global, registro de anotaciones 728
lista pendiente local, registro de anotaciones 728
lobsinfile, modificador de tipo de archivo
API db2Export 96
API db2Import 149
db2Load, API 205

M

manejadores de señales
API instalar manejador de señales 442
API interrumpir 440
manejo de punteros
copia de datos entre áreas de memoria 469
eliminar la referencia de direcciones 468
obtención de direcciones de variables 467
mensajes de error
plugins de seguridad 526
recuperación
API sqlaintp 351
vinculación 348
migración
base de datos
API sqlemgdb 443
registros de anotaciones 740
modificador de tipo de archivo de página de códigos 205
modificadores de tipo de archivo
API de exportación 96
API de importación 149
Load, API 205
movimiento de datos
entre bases de datos 149
múltiples peticiones concurrentes
cambio de niveles de aislamiento 487

N

niveles de aislamiento
cambiar 487
nochecklengths, modificador de tipo de archivo
API db2Import 149
db2Load, API 205
nodefaults, modificador de tipo de archivo
API db2Import 149
nodos
API abrir exploración de directorio DCS 438
directorios
API sqlectnd 412
entradas 446

nodos (*continuación*)
SOCKS
Estructura de datos `sqlc_node_struct` 670
Estructura de datos `sqlc_node_tcpip` 671
nodoubledel, modificador de tipo de archivo
API db2Export 96
API db2Import 149
db2Load, API 205
noeofchar, modificador de tipo de archivo
API db2Import 149
db2Load, API 205
noheader, modificador de tipo de archivo
db2Load, API 205
norowwarnings, modificador de tipo de archivo
db2Load, API 205
notypeid, modificador de tipo de archivo
API db2Import 149
nullindchar, modificador de tipo de archivo
API db2Import 149
db2Load, API 205
números de secuencia de anotaciones cronológicas (LSN)
cabeceras 726
cabeceras de registro de anotaciones cronológicas del gestor de anotaciones cronológicas 726
cambios 27

O

obtener mensaje de error, API
detalles 351

P

páginas de códigos
API db2Export 96
API db2Import 149
paquetes
creación
API sqlabndx 348
volver a crear 356
plugins
Autenticación GSS-API 584
contraseña, autenticación 556
gestión de bases de datos 511
ID, autenticación 556
recuperación de grupos 544
seguridad
API 527, 541
códigos de retorno 523
denominación, convenios de 536
desplegar 512, 513, 515
ejemplos 587
mensajes de error 526
restricciones (autenticación GSS-API) 586
restricciones (bibliotecas de plugins) 518
restricciones (resumen) 520
versiones 539
plugins de base de datos de proveedor 511
plugins de terceros 511
privilegios
bases de datos
otorgados durante la creación de la base de datos 404
procesador de línea de mandatos (CLP)
llamada a las API desde aplicaciones REXX 485

- productos de proveedor
 - API
 - copias de seguridad 588
 - restauraciones 588
 - DATA, estructura 613
 - INIT-INPUT, estructura 610
- productos de proveedor de copia de seguridad y restauración
 - visión general 511, 588
- programa de utilidad, registros de anotaciones
 - resumen 740
- programa de utilidad de carga
 - modificadores de tipo de archivo 205
- Protocolo de autenticación Kerberos
 - ejemplos 587
 - plugins
 - desplegar 515

Q

Quién debe utilizar este manual ix

R

- reclen, modificador de tipo de archivo
 - API db2Import 149
 - db2Load, API 205
- recuperación en avance
 - API db2Rollforward 296
- redistribución de datos
 - grupos de particiones de base de datos 472
- registro de anotaciones cronológicas de adición de campo largo 738
- registro de anotaciones cronológicas de alteración de tabla con adición de columnas 744
- registro de anotaciones cronológicas de atributo de alteración de tabla 744
- registro de anotaciones cronológicas de información de la partición 766
- registro de anotaciones cronológicas de las sentencias DDL 766
- registro de anotaciones cronológicas de no actualización de campo largo 738
- Registro de anotaciones cronológicas de preparación de subordinador de MPP 728
- registro de anotaciones cronológicas de supresión de campo largo 738
- registro de anotaciones de activación no registrada inicialmente 744
- registro de anotaciones de confirmación de coordinador de MPP 728
- registro de anotaciones de confirmación de subordinador de MPP 728
- registro de anotaciones de preparación de TM 728
- registro de anotaciones de preparación de XA 728
- registro de anotaciones del atributo de deshacer alteración de columna 744
- registro de datos de usuario formateado, registro de anotaciones 744
- registros de anotaciones
 - activación no registrada inicialmente 744
 - actualizar registro de estado de partición 744
 - actualizar registros 744
 - avance del espacio de tablas hasta finalización de punto en el tiempo 740
 - avance del espacio de tablas hasta inicio de punto en el tiempo 740
- registros de anotaciones (*continuación*)
 - cabeceras 726
 - cabeceras de gestor de anotaciones cronológicas 726
 - cambiar 744
 - cancelación heurística 728
 - cancelación normal 728
 - cargar lista pendiente 740
 - compensación de inicio de supresión de carga 740
 - confirmación de coordinador de MPP 728
 - confirmación de subordinador de MPP 728
 - confirmación heurística 728
 - confirmación normal 728
 - creación 744
 - delete 744
 - descartar
 - índices 744
 - tablas 744
 - deshacer adición de columnas 744
 - deshacer registro de anotaciones cronológicas de las sentencias DDL 766
 - deshacer registro de estado de partición 744
 - deshacer renombrar esquema 744
 - deshacer renombrar tabla 744
 - detalles de la tabla 744
 - espacio de tablas recuperado 740
 - fin de migración del catálogo del sistema 740
 - final de copia de seguridad 740
 - gestor de campos largos 738
 - gestor de datos 744
 - gestor de programas de utilidad 740
 - gestor de transacciones 728
 - inicializar tablas 744
 - inicio de carga 740
 - inicio de migración del catálogo del sistema 740
 - inicio de supresión de carga de tabla 740
 - insertar registros 744
 - libre de retrotracción 728
 - lista 722
 - lista pendiente global 728
 - lista pendiente local 728
 - preparación de subordinador de MPP 728
 - preparación de TM 728
 - preparación de XA 728
 - registro de anotaciones cronológicas de información de la partición 766
 - registro de anotaciones cronológicas de las sentencias DDL 766
 - registro de no actualización de campo largo 738
 - renombrar esquema 744
 - renombrar tabla 744
 - reorganizar tabla 744
 - resumen 722
 - retrotraer actualización de registro 744
 - retrotraer adición de columnas 744
 - retrotraer creación de tabla 744
 - retrotraer descarte de tabla 744
 - retrotraer inserción 744
 - retrotraer registro de estado de partición 744
 - retrotraer supresión de registro 744
 - suprimir
 - campo largo 738
 - gestor de datos 744
 - sustitución de importación (truncar) 744
- rendimiento
 - tablas
 - reorganizar 265
- renombrar registro de anotaciones de esquema 744

- renombrar registro de anotaciones de tabla 744
- reorganizar tabla, registro de anotaciones 744
- resolución de problemas
 - guías de aprendizaje 781
 - información en línea 781
 - plugins de seguridad 540
- retrotraer actualización de registro, registro de anotaciones 744
- retrotraer adición de columnas, registro de anotaciones 744
- retrotraer creación de tabla, registro de anotaciones 744
- retrotraer descarte de tabla, registro de anotaciones 744
- retrotraer inserción, registro de anotaciones 744
- retrotraer inserción de registro en registro de anotaciones de página vacía 744
- retrotraer supresión de registro, registro de anotaciones 744
- retrotraer supresión de registro en registro de anotaciones de página vacía 744
- RETURN-CODE, estructura 614

S

- secuencias de clasificación
 - definidas por el usuario 404
- seguridad
 - ejemplos 587
 - ID de usuario y contraseña 511
 - plugins
 - API 541, 546, 547, 548, 549, 553, 555, 562, 564, 565, 566, 567, 568, 570, 572, 574, 575, 577, 580
 - API (GSS-API) 584
 - API (ID de usuario/contraseña) 556
 - API (recuperación de grupos) 544
 - API (versiones) 539
 - API para validar contraseñas 581
 - bibliotecas 535
 - cargar 516, 530
 - códigos de retorno 523
 - consideraciones para 32 bits 540
 - consideraciones para 64 bits 540
 - denominación 536
 - desarrollar 530
 - desplegar 512, 513, 515, 520, 530
 - determinación de problemas 540
 - GSS-API (despliegue) 513
 - GSS-API (restricciones) 586
 - habilitar 530
 - inicialización 516
 - mensajes de error 526
 - restricciones para bibliotecas 518
 - secuencia de llamada 527
 - soporte para ID de usuario de dos componentes 537
 - SQLCODES 540
 - SQLSTATES 540
 - visión general 530
- sentencias SQL
 - ayuda
 - visualización 776
- simultaneidad
 - API cambiar el nivel de aislamiento 487
- sistemas principales
 - conexiones soportadas por DB2 Connect
 - API sqlcledad 429
- SOCKS
 - nodos
 - utilización 670, 671
- SQL-DIR-ENTRY, estructura 642
- SQLB-TBS-STATS, estructura 644

- SQLB-TBSCONTQRY-DATA, estructura 646
- SQLB-TBSPQRY-DATA, estructura 648
- sqlbstsc, API 375
- SQLCA
 - detalles 653
 - recuperación de mensajes de error 31, 351, 470
- SQLCODE
 - valores 31
- SQLDA
 - visión general 655
- SQLDCOL, estructura 657
- sqlc_activate_db, API 381
- SQLC-ADDN-OPTIONS, estructura 660
- SQLC-CLIENT-INFO, estructura 662
- SQLC-CONN-SETTING, estructura 665
- sqlc_deactivate_db, API 384
- SQLC-NODE-LOCAL, estructura 668
- SQLC-NODE-NPIPE, estructura 669
- SQLC-NODE-STRUCT, estructura 670
- SQLC-NODE-TCPIP, estructura 671
- sqlc_tcp, API 390
- SQLCDBDESCEXT, estructura 681
- sqlc_dgne, API 88
- sqlc_lgne, API 446
- SQLCENINFO, estructura 689
- sqlc_eryc, API 450
- sqlc_eset, API 457
- SQLCETSDESC, estructura 672
- SQLCUPD, estructura 692
- SQLCMA, estructura 702
- SQLCLOPT, estructura 706
- SQLCSTATE
 - API 31
 - recuperación de mensajes 470
- SQLC-LSN, estructura 708
- SQLC-MEDIA-LIST, estructura 709
- SQLC-RLOG-INFO, estructura 714
- sqlc_ludrtd, API 472
- sqlc_luimpr, API 149
- SQLC-LUPI, estructura 715
- sqlc_luvint, API 598
- SQLC-WARN
 - mensajes 31
- SQLCXA-XID, estructura 717
- supervisión de instantáneas
 - adición de peticiones 42
- suprimir registro, registro de anotaciones 744
- suprimir registro en registro de anotaciones de página vacía 744
- sustitución de importación (truncar), registro de anotaciones 744

T

- tablas
 - exportación a archivos 96
 - importación de archivos 149
 - inicio de supresión de carga, registro de anotaciones 740
- TCP/IP
 - SOCKS 670, 671
- terminación
 - anormal 80
- terminación anormal
 - API db2DatabaseRestart 80
- términos y condiciones
 - publicaciones 781

- timeformat, modificador de tipo de archivo
 - API db2Import 149
 - db2Load, API 205
- timestampformat, modificador de tipo de archivo
 - API db2import 149
 - db2load, API 205
- totalreespace, modificador de tipo de archivo
 - db2Load, API 205
- transacciones dudosas
 - retrotracción mediante API 499

U

- usedefaults, modificador de tipo de archivo
 - API db2Import 149
 - db2Load, API 205

V

- VENDOR-INFO, estructura 608
- vinculación
 - errores 404
 - programas de aplicación con bases de datos 348
 - valores por omisión 348

Z

- zoned decimal, modificador de tipo de archivo
 - db2Load, API 205



SC11-8067-00



Spine information:

IBM DB2 10.1 para Linux, UNIX y Windows

Consulta de las API administrativas

