IBM DB2 Information Integrator

Consulta de la API del desarrollador de reiniciadores

Versión 8

IBM DB2 Information Integrator

Consulta de la API del desarrollador de reiniciadores

Versión 8

Antes de utilizar esta información y el producto al que da soporte, asegúrese de leer la información general bajo el apartado “Avisos” en la página 249.

Este manual es la traducción del original inglés *IBM DB2 Information Integrator Wrapper Developer's API Reference Version 8.*

Puede solicitar publicaciones de IBM en línea o a través del representante de IBM de su localidad:

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

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

Contenido

Acerca de este manual

Este manual proporciona información de consulta acerca de las clases de API que puede utilizar cuando desarrolla un reiniciador para una fuente de datos. Se presenta una visión general de cada clase con descripciones generales e información de utilización. Las funciones de miembro (y los constructores y destructores, si es pertinente) de cada clase se listan con la finalidad, la sintaxis, los valores de retorno y los argumentos de entrada y salida necesarios correspondientes. Una vez que ha desarrollado un reiniciador para una fuente de datos, puede utilizar la fuente de datos en un sistema de bases de datos federadas.

A quién va dirigido este manual

Este manual está pensado para los DBA y los desarrolladores de reiniciadores que utilizan las API con DB2 Information Integrator que ofrece IBM.

Convenios y terminología utilizados en este manual

Convenios de resaltado:

En el presente manual se utilizan los siguientes convenios de resaltado:

Tipo negrita

Señala mandatos y controles de interfaz gráfica de usuario (GUI) (tales como nombres de campos, nombres de pulsadores y opciones de menú). El tipo negrita se utiliza para designar notas, restricciones, prerequisitos y recomendaciones.

`Tipo Monoespaciado`

Señala texto que escribe el usuario, nombres de archivos y ejemplos de código. El tipo monoespaciado también se utiliza para sentencias de SQL o nombres de parámetros de mandatos de DB2.

Tipo Cursiva

Señala sentencias de SQL o valores de parámetros de mandatos de DB2 que el usuario sustituye por un valor adecuado. Los ejemplos de sentencias de SQL o de mandatos de DB2 utilizan el tipo cursiva para

valores de parámetros de ejemplo. El tipo cursiva se utiliza para enfatizar palabras, para identificar nuevos términos y para señalar títulos de documentos.

TIPO MAYÚSCULAS

Señala los nombres de mandatos de DB2 y sentencias de SQL y sus palabras clave. El tipo mayúsculas también se utiliza para nombres de tipos de datos, opciones y acrónimos.

Métodos y clases de C++

La secciones siguientes describen las clases que puede utilizar con la API de C++. Estas clases incluyen:

- Clases de catálogo
- Clases de reiniciador
- Clases de servidor
- Clases de usuario
- Clases de apodo
- Clases de conexión remota
- Clases de operación
- Clases de petición
- Clases de datos
- Clases de programas de utilidad de reiniciador

Se presenta una visión general de cada clase con descripciones generales e información de utilización. Las funciones de miembro (y los constructores y destructores, si es pertinente) de cada clase se listan con la finalidad, la sintaxis, los valores de retorno y los argumentos de entrada y salida necesarios correspondientes.

Clases de catálogo para la API C++

La tabla siguiente describe cada clase de catálogo para la API C++.

Tabla 1. Clases de catálogo

Nombre de clase	Descripción
Catalog_Option	La clase que representa una opción de catálogo con nombre, de un solo valor o de varios valores. Entre éstas, se distinguen las subclases. No cree instancias ni subclases de esta clase.
Wrapper_Info	Subclase de Catalog_Info que contiene información de catálogo para un reiniciador.
Server_Info	La clase que encapsula la información de catálogo para un objeto de servidor (datos de las sentencias CREATE SERVER y ALTER SERVER).
User_Info	Clase que encapsula la información de catálogo para una correlación de usuario de las sentencias CREATE USER MAPPING y ALTER USER MAPPING.

Tabla 1. Clases de catálogo (continuación)

Nombre de clase	Descripción
Column_Info	Clase que encapsula la información de catálogo para una columna de un apodo. Esta clase incluye información estadística de columna.
Nickname_Info	Clase que encapsula una definición de apodo del catálogo e incluye definiciones de columna e índice.

Consulta relacionada:

- "Clase Catalog_Option" en la página 2
- "Clase Wrapper_Info" en la página 4
- "Clase Server_Info" en la página 13
- "Clase User_Info" en la página 25
- "Clase Column_Info" en la página 32
- "Clase Nickname_Info" en la página 54

Clase Catalog_Option

Este tema describe la clase Catalog_Option y proporciona detalles para cada función de miembro.

Visión general

La clase Catalog_Option representa una opción de catálogo con nombre, de un solo valor o de varios valores. Entre éstas, se distinguen las subclases.

La clase Catalog_Option es una de las clases de catálogo para la API C++.

Uso El servidor federado DB2 crea una instancia de las opciones de catálogo con nombre para cada opción que se especifica en el catálogo o en una sentencia DDL (Data Definition Language - Lenguaje de definición de datos). El reiniciador puede crear una instancia de estas opciones de catálogo con nombre (mediante el método add_option() de una subclase Catalog_Info) para añadir o cambiar valores de opción.

Archivo

sqlqg_catalog.h

Miembros de datos

Ninguno.

Funciones de miembro

La tabla siguiente describe cada función de miembro de la clase Catalog_Option. Después de la tabla, se describe cada función más detalladamente.

Tabla 2. Funciones de miembro para la clase Catalog_Option

Función de miembro	Descripción
get_name	Recuperar el nombre de opción.
get_action	Recuperar la acción (ADD, SET, DROP o ninguna) para esta opción.
get_value	Recuperar el valor de opción.
get_value	Recuperar el valor y la longitud de opción.

Función get_name

Finalidad

Recuperar el nombre de opción.

Sintaxis

```
sqluint8* get_name ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Nombre de opción terminado por nulo.

Función get_action

Finalidad

Recuperar la acción (ADD, SET, DROP o ninguna) para esta opción.

Sintaxis

```
Catalog_Option::Action get_action ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Acción.

Función get_value

Finalidad

Recuperar el valor de opción.

Sintaxis

```
virtual sqluint8* get_value ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Valor de opción terminado por nulo.

Función get_value

Finalidad

Recuperar el valor y la longitud de opción.

Sintaxis

```
virtual sqluint8* get_value (sqlint32* a_length)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 3. Argumentos de salida para la función de miembro get_value

Nombre	Tipo de datos	Descripción
a_length	sqlint32*	Longitud del valor de opción.

Valor de retorno

Valor de opción terminado por nulo.

Consulta relacionada:

• "Clases de catálogo para la API C++" en la página 1

Clase Wrapper_Info

Este tema describe la clase Wrapper_Info y proporciona detalles para el constructor y las funciones de miembro.

Visión general

La clase Wrapper_Info es una subclase de Catalog_Info y contiene información de catálogo para un reiniciador.

La clase Wrapper_Info es una de las clases de catálogo para la API C++.

Uso El servidor federado DB2 crea una instancia de esta clase para que contenga información de una sentencia CREATE WRAPPER o del catálogo de DB2 Information Integrator. El reiniciador crea una instancia de esta clase cuando se añade información durante las operaciones CREATE WRAPPER o ALTER WRAPPER.

Archivo
sqlqg_catalog.h

Miembros de datos
Ninguno.

Constructores y funciones de miembro

Las tablas siguientes describen el constructor y las funciones de miembro de la clase Wrapper_Info. El constructor y las funciones se describen más detalladamente después de las tablas.

Tabla 4. Constructores para la clase Wrapper_Info

Constructor	Descripción
Wrapper_Info	Construir una instancia de Wrapper_Info.

Tabla 5. Funciones de miembro para la clase Wrapper_Info

Función de miembro	Descripción
get_wrapper_name	Recuperar el nombre de reiniciador.
get_corelib	Recuperar el nombre de la biblioteca de reiniciadores.
set_type	Establecer el tipo de reiniciador.
get_type	Recuperar el tipo de reiniciador.
set_version	Establecer la versión de código de reiniciador.
get_version	Recuperar la versión de reiniciador, si existe.
copy	Duplicar un objeto Wrapper_Info.
add_option	Añadir una opción en la información de catálogo.
drop_option	Suprimir una opción de una clase Catalog_Info.

Tabla 5. Funciones de miembro para la clase Wrapper_Info (continuación)

Función de miembro	Descripción
get_option	Recuperar una opción de catálogo por nombre.
get_first_option	Recuperar un puntero a la primera opción de la cadena de opciones.
get_next_option	Recuperar la siguiente opción de la cadena de opciones.

Constructor Wrapper_Info

Finalidad

Construir una instancia de Wrapper_Info.

Sintaxis

```
Wrapper_Info ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Función get_wrapper_name

Finalidad

Recuperar el nombre de reiniciador.

Sintaxis

```
sqlint32 get_wrapper_name (sqluint8** a_name)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 6. Argumentos de salida para la función de miembro get_wrapper_name

Nombre	Tipo de datos	Descripción
a_name	sqluint8**	Puntero a la serie de nombre de reiniciador terminada por nulo.

Valor de retorno

Código de retorno. El valor es 0, si existe el nombre. El valor es SQLQG_NOVALUE, si no existe el nombre.

Función get_corelib

Finalidad

Recuperar el nombre de la biblioteca de reiniciadores. Este nombre es el nombre base de la biblioteca de reiniciadores de la sentencia CREATE WRAPPER.

Sintaxis

```
sqlint32 get_corelib (sqluint8** a_lib_name)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 7. Argumentos de salida para la función de miembro get_corelib

Nombre	Tipo de datos	Descripción
a_lib_name	sqluint8**	Puntero a la serie de nombre de biblioteca terminada por nulo.

Valor de retorno

Código de retorno. El valor es 0, si existe el nombre. El valor es SQLQG_NOVALUE, si no existe el nombre.

Función set_type

Finalidad

Establecer el tipo de reiniciador.

Sintaxis

```
void set_type (sqluint8 a_wrapper_type)
```

Argumentos de entrada

Tabla 8. Argumentos de entrada para la función de miembro set_type

Nombre	Tipo de datos	Descripción
a_wrapper_type	sqluint8	Tipo de reiniciador (debe ser N).

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función get_type

Finalidad

Recuperar el tipo de reiniciador.

Sintaxis

```
sqlint32 get_type (sqluint8* a_wrapper_type)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 9. Argumentos de salida para la función de miembro get_type

Nombre	Tipo de datos	Descripción
a_wrapper_type	sqluint8*	Tipo de reiniciador.

Valor de retorno

Código de retorno. El valor es 0, si existe el tipo. El valor es SQLQG_NOVALUE, si no existe el tipo.

Función set_version

Finalidad

Establecer la versión de código de reiniciador.

Sintaxis

```
void set_version (sqlint32 a_wrapper_version)
```

Argumentos de entrada

Tabla 10. Argumentos de entrada para la función de miembro set_version

Nombre	Tipo de datos	Descripción
a_wrapper_version	sqlint32	Versión de código de reiniciador.

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función get_version

Finalidad

Recuperar la versión de reiniciador, si existe.

Sintaxis

```
sqlint32 get_version (sqlint32* a_wrapper_version)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 11. Argumentos de salida para la función de miembro get_version

Nombre	Tipo de datos	Descripción
a_wrapper_version	sqlint32*	Versión de código de reiniciador.

Valor de retorno

Código de retorno. El valor es 0, si existe la versión. El valor es SQLQG_NOVALUE, si no existe la versión.

Función copy

Finalidad

Duplicar un objeto Wrapper_Info.

Sintaxis

```
sqlint32 copy (Wrapper_Info** a_new_wrapper_info)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 12. Argumentos de salida para la función de miembro copy

Nombre	Tipo de datos	Descripción
a_new_wrapper_info	Wrapper_Info**	Puntero al Wrapper_Info recién asignado.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función add_option

Dónde se define

Catalog_Info

Finalidad

Añadir una opción en la información de catálogo.

Uso El reiniciador puede invocar esta función de miembro cuando se añaden opciones generadas por el reiniciador al catálogo durante el proceso de las sentencias CREATE WRAPPER y ALTER WRAPPER.

Sintaxis

```
sqlint32 add_option (sqluint8*  a_opt_name,
                     sqlint32   a_name_len,
                     sqluint8*  a_opt_value,
                     sqlint32   a_value_len,
                     Catalog_Option::Action a_action
                     = Catalog_Option::sqlqg_None,
                       char*    a_option_type = "")
```

Argumentos de entrada

Tabla 13. Argumentos de entrada para la función de miembro add_option

Nombre	Tipo de datos	Descripción
a_opt_name	sqluint8*	Nombre de opción (no terminado por nulo).
a_name_len	sqlint32	Longitud del nombre de opción.
a_opt_value	sqluint8*	Valor de opción (no terminado por nulo).
a_value_len	sqlint32	Longitud del valor de opción.
a_action	Catalog_Option::Action	Acción para esta opción (ADD, SET, DROP o ninguna).
a_option_type	char*	Señal que se utiliza en el mensaje de error SQLN1884 si se trata de una opción duplicada. Para opciones de reiniciador, utilice la constante SQLQG_WRAPPER_OPTION que está definida en el archivo de cabecera sqlqg_misc.h.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función drop_option

Dónde se define

Catalog_Info

Finalidad

Suprimir una opción de una clase Catalog_Info. Esta función de miembro no elimina la opción del catálogo. La opción se elimina del catálogo cuando se añade una opción para un objeto delta Catalog_Info con una acción de Catalog_Option::sqlqg_Drop.

Sintaxis

```
sqlint32 drop_option (Catalog_Option* a_option)
```

Argumentos de entrada

Tabla 14. Argumentos de entrada para la función de miembro drop_option

Nombre	Tipo de datos	Descripción
a_option	Catalog_Option*	Opción que se debe eliminar.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_option

Dónde se define

Catalog_Info

Finalidad

Recuperar una opción de catálogo por nombre.

Uso El argumento de salida `option` es nulo si no se encuentra la opción.

Sintaxis

```
sqlint32 get_option (sqluint8*        a_opt_name,
                     sqlint32         a_name_len,
                     Catalog_Option** a_option)
```

Argumentos de entrada

Tabla 15. Argumentos de entrada para la función de miembro get_option

Nombre	Tipo de datos	Descripción
a_opt_name	sqluint8*	Nombre de opción (no terminado por nulo).
a_name_len	sqlint32	Longitud del nombre de opción.

Argumentos de salida

Tabla 16. Argumentos de salida para la función de miembro get_option

Nombre	Tipo de datos	Descripción
a_option	Catalog_Option**	Opción encontrada.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha encontrado la opción.

Función get_first_option

Dónde se define

Catalog_Info

Finalidad

Recuperar un puntero a la primera opción de la cadena de opciones.

Sintaxis

```
Catalog_Option* get_first_option ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Puntero a la primera opción de la cadena de opciones. El valor es nulo si la cadena está vacía.

Función get_next_option

Dónde se define

Catalog_Info

Finalidad

Recuperar la siguiente opción de la cadena de opciones.

Sintaxis

```
Catalog_Option* get_next_option (Catalog_Option* a_current_option)
```

Argumentos de entrada

Tabla 17. Argumentos de entrada para la función de miembro get_next_option

Nombre	Tipo de datos	Descripción
a_current_option	Catalog_Option*	Opción actual.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la siguiente opción de la cadena de opciones. El valor es nulo si está al final.

Consulta relacionada:

- "Clases de catálogo para la API C++" en la página 1

Clase Server_Info

Este tema describe la clase Server_Info y proporciona detalles para los constructores y las funciones de miembro.

Visión general

La clase Server_Info encapsula la información de catálogo para un objeto de servidor (datos de las sentencias CREATE SERVER y ALTER SERVER).

La clase Server_Info es una de las clases de catálogo para la API C++.

Uso El servidor federado DB2 crea una instancia de esta clase para que contenga información de una sentencia CREATE SERVER o ALTER SERVER o para que contenga información del catálogo de DB2 Information Integrator. El reiniciador también crea una instancia de esta clase cuando se añade información durante las operaciones de sentencia CREATE SERVER o ALTER SERVER.

Archivo
sqlqg_catalog.h

Miembros de datos
Ninguno.

Constructores y funciones de miembro

Las tablas siguientes describen los constructores y las funciones de miembro de la clase Server_Info. Los constructores y las funciones se describen más detalladamente después de las tablas.

Tabla 18. Constructores para la clase Server_Info

Constructor	Descripción
Server_Info	Construir un objeto Server_Info vacío.
Server_Info	Construir un objeto Server_Info con parámetros especificados.

Tabla 19. Funciones de miembro para la clase Server_Info

Función de miembro	Descripción
add_option	Añadir una opción de un solo valor a la información de catálogo.
drop_option	Suprimir una opción de una clase Catalog_Info.
get_option	Recuperar una opción de catálogo por nombre.
get_first_option	Recuperar un puntero a la primera opción de la cadena de opciones.
get_next_option	Recuperar la siguiente opción de la cadena de opciones.
get_basic_info	Recuperar información básica de un objeto Server_Info.
get_server_name	Recuperar el nombre de servidor de Server_Info, si el nombre es válido.
set_server_type	Establecer el tipo del servidor.
get_server_type	Recuperar el tipo de servidor de Server_Info, si el tipo es válido.
set_server_version	Establecer la versión del servidor.
get_server_version	Recuperar la versión de servidor de Server_Info, si la versión es válida.
get_wrapper_name	Recuperar el nombre de reiniciador de Server_Info, si el nombre es válido.
merge	Fusionar un objeto delta Server_Info en el objeto actual.
copy	Duplicar un objeto Server_Info.

Constructor Server_Info

Finalidad

Construir un objeto Server_Info vacío.

Sintaxis

```
Server_Info ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Ninguno.

Constructor Server_Info

Finalidad
Construir un objeto Server_Info con parámetros especificados.

Sintaxis

```
Server_Info (sqlint32*   a_rc,
             sqluint8*   a_server_name,
             sqlint32    a_name_len,
             sqluint8*   a_server_type,
             sqlint32    a_type_len,
             sqluint8*   a_server_version,
             sqlint32    a_ver_len,
             sqluint8*   a_server_wrapper,
             sqlint32    a_wrap_len);
```

Argumentos de entrada

Tabla 20. Argumentos de entrada para el constructor Server_Info

Nombre	Tipo de datos	Descripción
a_server_name	sqluint8*	Nombre de servidor (no terminado por nulo).
a_name_len	sqlint32	Longitud del nombre de servidor.
a_server_type	sqluint8*	Tipo de servidor (no terminado por nulo).
a_type_len	sqlint32	Longitud del tipo de servidor.
a_server_version	sqluint8*	Versión de servidor (no terminada por nulo).
a_ver_len	sqlint32	Longitud de la versión de servidor.
a_server_wrapper	sqluint8*	Nombre de reiniciador (no terminado por nulo).
a_wrap_len	sqlint32	Longitud del nombre de reiniciador.

Argumentos de salida

Tabla 21. Argumentos de salida para el constructor Server_info

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Puntero al código de retorno; 0 indica que la operación se ha realizado satisfactoriamente.

Valor de retorno

Ninguno.

Función add_option

Dónde se define

Catalog_Info

Finalidad

Añadir una opción de un solo valor a la información de catálogo.

Uso

El reiniciador puede invocar esta función de miembro cuando se añaden opciones generadas por el reiniciador al catálogo durante el proceso de la sentencia CREATE SERVER o ALTER SERVER.

Sintaxis

```
sqlint32 add_option (sqluint8*  a_opt_name,
                     sqlint32   a_opt_name_len,
                     sqluint8*  a_opt_value,
                     sqlint32   a_opt_value_len,
                     Catalog_Option::Action     a_act
                   = Catalog_Option::sqlqg_None,
                     char*      a_opt_type = "")
```

Argumentos de entrada

Tabla 22. Argumentos de entrada para la función de miembro add_option

Nombre	Tipo de datos	Descripción
a_opt_name	sqluint8*	Nombre de opción (no terminado por nulo).
a_opt_name_len	sqlint32	Longitud del nombre de opción.
a_opt_value	sqluint8*	Valor de opción (no terminado por nulo).
a_opt_value_len	sqlint32	Longitud del valor de opción.
a_act	Catalog_Option::Action	Acción para esta opción (ADD, SET, DROP o ninguna).

Tabla 22. Argumentos de entrada para la función de miembro add_option (continuación)

Nombre	Tipo de datos	Descripción
a_opt_type	char*	Señal que se utiliza en el mensaje de error SQLN1884 si se trata de una opción duplicada. Utilice la constante SQLQG_SERVER_OPTION que está definida en el archivo de cabecera sqlqg_misc.h.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función drop_option

Dónde se define

Catalog_Info

Finalidad

Suprimir una opción de una clase Catalog_Info. Esta función de miembro no elimina la opción del catálogo. La opción se elimina cuando se añade una opción para un objeto delta Catalog_Info con una acción de Catalog_Option::sqlqg_Drop.

Sintaxis

```
sqlint32 drop_option (Catalog_Option* a_option)
```

Argumentos de entrada

Tabla 23. Argumentos de entrada para la función de miembro drop_option

Nombre	Tipo de datos	Descripción
a_option	Catalog_Option*	Opción que se debe eliminar.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_option

Dónde se define
Catalog_Info

Finalidad
Recuperar una opción de catálogo por nombre.

Uso El argumento de salida `option` es nulo si no se encuentra la opción.

Sintaxis

```
sqlint32 get_option (sqluint8*        a_opt_name,
                     sqlint32         a_name_len,
                     Catalog_Option** a_option)
```

Argumentos de entrada

Tabla 24. Argumentos de entrada para la función de miembro get_option

Nombre	Tipo de datos	Descripción
a_opt_name	sqluint8*	Nombre de opción (no terminado por nulo).
a_name_len	sqlint32	Longitud del nombre de opción.

Argumentos de salida

Tabla 25. Argumentos de salida para la función de miembro get_option

Nombre	Tipo de datos	Descripción
a_option	Catalog_Option**	Opción encontrada.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha encontrado la opción.

Función get_first_option

Dónde se define
Catalog_Info

Finalidad
Recuperar un puntero a la primera opción de la cadena de opciones.

Sintaxis

```
Catalog_Option* get_first_option ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la primera opción de la cadena de opciones. El valor es nulo si la cadena está vacía.

Función get_next_option

Dónde se define
Catalog_Info

Finalidad
Recuperar la siguiente opción de la cadena de opciones.

Sintaxis

```
Catalog_Option* get_next_option (Catalog_Option* a_current_option)
```

Argumentos de entrada

Tabla 26. Argumentos de entrada para la función de miembro get_next_option

Nombre	Tipo de datos	Descripción
a_current_option	Catalog_Option*	Opción actual.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la siguiente opción de la cadena. El valor es nulo si está al final.

Función get_basic_info

Finalidad
Recuperar información básica de un objeto Server_Info. A diferencia de otras rutinas de miembro get_XXX, la función de miembro get_basic_info devuelve SQLQG_ERROR y anota cronológicamente un error si alguno de los elementos de datos no es válido.

Sintaxis

```
sqlint32 get_basic_info (sqluint8** a_server_name,
                         sqluint8** a_server_type,
                         sqluint8** a_server_version,
                         sqluint8** a_server_wrapper)
```

Argumentos de entrada
Ninguno.

Argumentos de salida

Tabla 27. Argumentos de salida para la función de miembro get_basic_info

Nombre	Tipo de datos	Descripción
a_server_name	sqluint8**	Puntero al nombre de servidor terminado por nulo.
a_server_type	sqluint8**	Puntero al tipo de servidor terminado por nulo.
a_server_version	sqluint8**	Puntero a la versión de servidor terminada por nulo
a_server_wrapper	sqluint8**	Puntero al nombre de reiniciador terminado por nulo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_server_name

Finalidad

Recuperar el nombre de servidor de Server_Info, si el nombre es válido.

Sintaxis

```
sqlint32 get_server_name (sqluint8** a_server_name)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 28. Argumentos de salida para la función de miembro get_server_name

Nombre	Tipo de datos	Descripción
a_server_name	sqluint8**	Puntero a un nombre de servidor terminado por nulo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el nombre de servidor.

Función set_server_type

Finalidad

Establecer el tipo del servidor.

Uso El reiniciador puede invocar esta función de miembro durante el proceso de la sentencia CREATE SERVER o ALTER SERVER para proporcionar un valor por omisión cuando no existe ningún valor por omisión en la sentencia original.

Sintaxis

```
sqlint32 set_server_type (sqluint8* a_server_type,
                          sqlint32  a_server_type_len)
```

Argumentos de entrada

Tabla 29. Argumentos de entrada para la función de miembro set_server_type

Nombre	Tipo de datos	Descripción
a_server_type	sqluint8*	Tipo de servidor (no terminado por nulo).
a_server_type_len	sqlint32	Longitud de tipo de servidor.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_server_type

Finalidad

Recuperar el tipo de servidor de Server_Info, si el tipo es válido.

Sintaxis

```
sqlint32 get_server_type (sqluint8** a_server_type)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 30. Argumentos de salida para la función de miembro get_server_type

Nombre	Tipo de datos	Descripción
a_server_type	sqluint8**	Puntero a un tipo de servidor terminado por nulo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el tipo de servidor.

Función set_server_version

Finalidad

Establecer la versión del servidor.

Uso El reiniciador puede invocar esta función de miembro durante el proceso de la sentencia CREATE SERVER o ALTER SERVER para proporcionar una versión de servidor por omisión si no se ha especificado una versión de servidor por omisión en la DDL.

Sintaxis

```
sqlint32 set_server_version (sqluint8* a_server_version,
                             sqlint32  a_server_version_len)
```

Argumentos de entrada

Tabla 31. Argumentos de entrada para la función de miembro set_server_version

Nombre	Tipo de datos	Descripción
a_server_version	sqluint8*	Versión de servidor (no terminada por nulo).
a_server_version_len	sqlint32	Longitud de la versión de servidor.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_server_version

Finalidad

Recuperar la versión de servidor de Server_Info, si la versión es válida.

Sintaxis

```
sqlint32 get_server_version (sqluint8** a_server_version)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 32. Argumentos de salida para la función de miembro get_server_version

Nombre	Tipo de datos	Descripción
a_server_version	sqluint8**	Puntero a una versión de servidor terminada por nulo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido la versión de servidor.

Función get_wrapper_name

Finalidad

Recuperar el nombre de reiniciador de Server_Info, si el nombre es válido.

Sintaxis

```
sqlint32 get_wrapper_name (sqluint8** a_wrapper_name)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 33. Argumentos de salida para la función de miembro get_wrapper_name

Nombre	Tipo de datos	Descripción
a_wrapper_name	sqluint8**	Puntero a un nombre de reiniciador terminado por nulo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha

realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el nombre de reiniciador.

Función merge

Finalidad

Fusionar un objeto delta Server_Info en el objeto actual.

Sintaxis

```
sqlint32 merge (Server_Info* a_delta_info)
```

Argumentos de entrada

Tabla 34. Argumentos de entrada para la función de miembro merge

Nombre	Tipo de datos	Descripción
a_delta_info	Server_Info*	Objeto delta.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función copy

Finalidad

Duplicar un objeto Server_Info.

Sintaxis

```
sqlint32 copy (Server_Info** a_new_server_info)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 35. Argumentos de salida para la función de miembro copy

Nombre	Tipo de datos	Descripción
a_new_server_info	Server_Info**	Puntero al nuevo objeto Server_Info.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Consulta relacionada:

- "Clases de catálogo para la API C++" en la página 1

Clase User_Info

Este tema describe la clase User_Info y proporciona detalles para el constructor y las funciones de miembro.

Visión general

La clase User_Info encapsula la información de catálogo para una correlación de usuario de las sentencias CREATE USER MAPPING y ALTER USER MAPPING.

La clase User_Info es una de las clases de catálogo para la API C++.

Uso El servidor federado DB2 crea una instancia de esta clase para que contenga información de una sentencia CREATE USER MAPPING o ALTER USER MAPPING o para que contenga información del catálogo de DB2 Information Integrator. El reiniciador crea una instancia de esta clase cuando se añade información durante el proceso de la sentencia CREATE USER MAPPING o ALTER USER MAPPING.

Archivo
sqlqg_catalog.h

Miembros de datos
Ninguno.

Constructores y funciones de miembro

Las tablas siguientes describen el constructor y las funciones de miembro de la clase User_Info. El constructor y las funciones se describen más detalladamente después de las tablas.

Tabla 36. Constructores para la clase User_Info

Constructor	Descripción
User_Info	Construir un objeto User_Info (vacío) por omisión.

Tabla 37. Funciones de miembro para la clase User_Info

Función de miembro	Descripción
get_server_name	Recuperar el nombre de servidor para esta correlación de usuario.
get_authid	Recuperar el ID de autorización para esta correlación de usuario.

Tabla 37. Funciones de miembro para la clase User_Info (continuación)

Función de miembro	Descripción
merge	Fusionar un objeto delta User_Info en el objeto actual.
copy	Crear una copia del objeto User_Info actual.
add_option	Añadir una opción de un solo valor a la información de catálogo.
drop_option	Suprimir una opción de una clase Catalog_Info.
get_option	Recuperar una opción de catálogo por nombre.
get_first_option	Recuperar un puntero a la primera opción de la cadena de opciones.
get_next_option	Recuperar la siguiente opción de la cadena de opciones.

Constructor de User_Info

Finalidad

Construir un objeto User_Info (vacío) por omisión.

Sintaxis

```
User_Info ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función get_server_name

Finalidad

Recuperar el nombre de servidor para esta correlación de usuario.

Sintaxis

```
sqlint32 get_server_name (sqluint8** a_server_name)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 38. Argumentos de salida para la función de miembro get_server_name

Nombre	Tipo de datos	Descripción
a_server_name	sqluint8**	Puntero a un nombre de servidor terminado por nulo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que el nombre de servidor no es válido o no se ha establecido.

Función get_authid

Finalidad

Recuperar el ID de autorización para esta correlación de usuario.

Sintaxis

```
sqlint32 get_authid (sqluint8** a_authid)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 39. Argumentos de salida para la función de miembro get_authid

Nombre	Tipo de datos	Descripción
a_authid	sqluint8**	Puntero al ID de autorización terminado por nulo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que el ID no es válido o no se ha establecido.

Función merge

Finalidad

Fusionar un objeto delta User_Info en el objeto actual.

Sintaxis

```
sqlint32 merge (User_Info* a_delta_info)
```

Argumentos de entrada

Tabla 40. Argumentos de entrada para la función de miembro merge

Nombre	Tipo de datos	Descripción
a_delta_info	User_Info*	Objeto delta que se debe fusionar.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función copy

Finalidad

Crear una copia del objeto User_Info actual.

Sintaxis

```
sqlint32 copy (User_Info** a_new_user_info)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 41. Argumentos de salida para la función de miembro copy

Nombre	Tipo de datos	Descripción
a_new_user_info	User_Info**	Puntero al objeto duplicado.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función add_option

Finalidad

Añadir una opción de un solo valor a la información de catálogo.

Uso El reiniciador puede invocar esta función de miembro durante el proceso de la sentencia CREATE USER MAPPING o ALTER USER MAPPING para añadir opciones generadas por el reiniciador.

Sintaxis

```
sqlint32 add_option (sqluint8*  a_opt_name,
                     sqlint32   a_opt_name_len,
                     sqluint8*  a_opt_value,
                     sqlint32   a_opt_value_len,
                     Catalog_Option::Action     a_act
                      = Catalog_Option::sqlqg_None,
                     char*      option_type = "")
```

Argumentos de entrada

Tabla 42. Argumentos de entrada para la función de miembro add_option

Nombre	Tipo de datos	Descripción
a_opt_name	sqluint8*	Nombre de opción (no terminado por nulo).
a_opt_name_len	sqlint32	Longitud del nombre de opción.
a_opt_value	sqluint8*	Valor de opción (no terminado por nulo).
a_opt_value_len	sqlint32	Longitud del valor de opción.
a_act	Catalog_Option::Action	Acción para esta opción (ADD, SET, DROP o ninguna).
a_opt_type	char*	Señal que se utiliza en un mensaje de error SQLN1884 si se trata de una opción duplicada. Utilice la constante SQLQG_USER_OPTION que está definida en el archivo de cabecera sqlqg_misc.h.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función drop_option

Dónde se define

Catalog_Info

Finalidad

Suprimir una opción de una clase Catalog_Info. Esta función de miembro no elimina la opción del catálogo. La opción se elimina cuando se añade una opción para un objeto delta Catalog_Info con una acción de Catalog_Option::sqlqg_Drop.

Sintaxis

```
sqlint32 drop_option (Catalog_Option* a_option)
```

Argumentos de entrada

Tabla 43. Argumentos de entrada para la función de miembro drop_option

Nombre	Tipo de datos	Descripción
a_option	Catalog_Option*	Opción que se debe eliminar.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_option

Dónde se define

Catalog_Info

Finalidad

Recuperar una opción de catálogo por nombre.

Uso El argumento de salida `option` es nulo si no se encuentra la opción.

Sintaxis

```
sqlint32 get_option (sqluint8*        a_opt_name,
                     sqlint32         a_name_len,
                     Catalog_Option** a_option)
```

Argumentos de entrada

Tabla 44. Argumentos de entrada para la función de miembro get_option

Nombre	Tipo de datos	Descripción
a_opt_name	sqluint8*	Nombre de opción (no terminado por nulo).
a_name_len	sqlint32	Longitud del nombre de opción.

Argumentos de salida

Tabla 45. Argumentos de salida para la función de miembro get_option

Nombre	Tipo de datos	Descripción
a_option	Catalog_Option**	Opción encontrada.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha encontrado la opción.

Función get_first_option

Dónde se define
Catalog_Info

Finalidad
Recuperar un puntero a la primera opción de la cadena de opciones.

Sintaxis

```
Catalog_Option* get_first_option ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la primera opción de la cadena de opciones. El valor es nulo si la cadena está vacía.

Función get_next_option

Dónde se define
Catalog_Info

Finalidad
Recuperar la siguiente opción de la cadena de opciones.

Sintaxis

```
Catalog_Option* get_next_option (Catalog_Option* a_current_option)
```

Argumentos de entrada

Tabla 46. Argumentos de entrada para la función de miembro get_next_option

Nombre	Tipo de datos	Descripción
a_current_option	Catalog_Option*	Opción actual.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la siguiente opción de la cadena. El valor es nulo si está al final.

Consulta relacionada:

- "Clases de catálogo para la API C++" en la página 1

Clase Column_Info

Este tema describe la clase Column_Info y proporciona detalles para el constructor y las funciones de miembro.

Visión general

La clase Column_Info encapsula la información de catálogo para una columna de un apodo. Esta clase incluye información estadística de columna.

La clase Column_Info es una de las clases de catálogo para la API C++.

Uso El servidor federado DB2 crea una instancia de esta clase para que contenga información de una sentencia CREATE NICKNAME o ALTER NICKNAME o para que contenga información del catálogo de DB2 Information Integrator. El reiniciador crea una instancia de esta clase cuando se añade información durante las operaciones de sentencia CREATE NICKNAME o ALTER NICKNAME.

Archivo
sqlqg_catalog.h

Miembros de datos
Ninguno.

Constructores y funciones de miembro

Las tablas siguientes describen el constructor y las funciones de miembro de la clase Column_Info. El constructor y las funciones se describen más detalladamente después de las tablas.

Tabla 47. Constructores para la clase Column_Info

Constructor	Descripción
Column_Info	Construir un objeto Column_Info (vacío) por omisión.

Tabla 48. Funciones de miembro para la clase Column_Info

Función de miembro	Descripción
set_column_name	Establecer el nombre para la columna.
get_column_name	Recuperar el nombre de esta columna.

Tabla 48. Funciones de miembro para la clase Column_Info (continuación)

Función de miembro	Descripción
get_new_column_name	Recuperar el nuevo nombre de columna que se ha especificado en una sentencia ALTER COLUMN que incluye una cláusula ALTER (o SET) COLUMN para redenominar la columna.
set_column_type	Establecer el tipo para la columna.
get_column_type	Recuperar el tipo de columna local de Server_Info, si el tipo es válido.
set_for_bit_data	Establecer el distintivo FOR BIT DATA para la columna.
get_for_bit_data	Recuperar el distintivo FOR BIT DATA para la columna, si el distintivo es válido.
set_column_ID	Establecer el ID (la posición) para la columna.
get_column_ID	Recuperar el ID (la posición) de la columna, si el ID es válido.
set_org_length	Establecer la longitud máxima para la columna.
get_org_length	Recuperar la longitud de columna máxima de Column_Info, si el valor es válido.
set_org_scale	Establecer la escala numérica para la columna.
get_org_scale	Recuperar la escala numérica de Column_Info, si el valor es válido.
set_nulls	Establecer el distintivo de nulos permitidos para la columna.
get_nulls	Recuperar el distintivo de nulos permitidos de Column_Info, si el valor es válido.
set_avg_length	Establecer el promedio de longitud para la columna.
get_avg_length	Recuperar el promedio de longitud de columna de Column_Info, si el valor es válido.
set_high2key	Establecer el segundo valor más alto para la columna.
get_high2key	Recuperar el segundo valor más alto de Column_Info, si el valor es válido.

Tabla 48. Funciones de miembro para la clase Column_Info (continuación)

Función de miembro	Descripción
set_low2key	Establecer el segundo valor más bajo para la columna.
get_low2key	Recuperar el segundo valor más bajo de Column_Info, si el valor es válido.
get_default	Recuperar el valor por omisión de Column_Info, si el valor es válido.
set_colcard	Establecer la cardinalidad para la columna.
get_colcard	Recuperar la cardinalidad de Column_Info, si el valor es válido.
set_codepage1	Establecer la página de códigos para la columna.
get_codepage1	Recuperar la página de códigos de Column_Info, si el valor es válido.
set_codepage2	Establecer la página de códigos para la columna.
get_codepage2	Recuperar la página de códigos de Column_Info, si el valor es válido.
merge	Fusionar un objeto delta Column_Info en el objeto actual.
copy	Duplicar un objeto Column_Info.
add_option	Añadir una opción de un solo valor a la información de catálogo.
drop_option	Suprimir una opción de una clase Catalog_Info.
get_option	Recuperar una opción de catálogo por nombre.
get_first_option	Recuperar un puntero a la primera opción de la cadena de opciones.
get_next_option	Recuperar la siguiente opción de la cadena de opciones.

Constructor Column_Info

Finalidad

Construir un objeto Column_Info (vacío) por omisión.

Sintaxis

```
Column_Info ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Ninguno.

Función set_column_name

Finalidad
Establecer el nombre para la columna.

Uso No utilice esta función de miembro para cambiar el nombre de una columna.

Sintaxis

```
sqlint32 set_column_name (sqluint8* a_column_name,
                          sqlint32  a_column_name_len)
```

Argumentos de entrada

Tabla 49. Argumentos de entrada para la función de miembro set_column_name

Nombre	Tipo de datos	Descripción
a_column_name	sqluint8*	Nombre de columna (no terminado por nulo).
a_column_name_len	sqlint32	Longitud del nombre de columna.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_column_name

Finalidad
Recuperar el nombre de esta columna.

Sintaxis

```
sqlint32 get_column_name (sqluint8** a_column_name)
```

Argumentos de entrada
Ninguno.

Argumentos de salida

Tabla 50. Argumentos de salida para la función de miembro get_column_name

Nombre	Tipo de datos	Descripción
a_column_name	sqluint8**	Puntero a un nombre de columna terminado por nulo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que el nombre de columna no es válido o no se ha establecido.

Función get_new_column_name

Finalidad

Recuperar el nuevo nombre de columna que se ha especificado en una sentencia ALTER COLUMN que incluye una cláusula ALTER (o SET) COLUMN para redenominar la columna.

Sintaxis

```
sqlint32 get_new_column_name (sqluint8** a_new_col_name)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 51. Argumentos de salida para la función de miembro get_new_column_name

Nombre	Tipo de datos	Descripción
a_new_col_name	sqluint8**	Puntero a un nuevo nombre de columna terminado por nulo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el nuevo nombre de columna.

Función set_column_type

Finalidad

Establecer el tipo para la columna.

Sintaxis

```
sqlint32 set_column_type (sqluint8* a_column_type,
                          sqlint32  a_column_type_len)
```

Argumentos de entrada

Tabla 52. Argumentos de entrada para la función de miembro set_column_type

Nombre	Tipo de datos	Descripción
a_column_type	sqluint8*	Tipo de columna (no terminado por nulo).
a_column_type_len	sqlint32	Longitud del tipo de columna.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_column_type

Finalidad

Recuperar el tipo de columna local de Server_Info, si el tipo es válido.

Sintaxis

```
sqlint32 get_column_type (sqluint8** a_column_type)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 53. Argumentos de salida para la función de miembro get_column_type

Nombre	Tipo de datos	Descripción
a_column_type	sqluint8**	Puntero al tipo de columna terminado por nulo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el tipo.

Función set_for_bit_data

Finalidad

Establecer el distintivo FOR BIT DATA para la columna.

Sintaxis

```
void set_for_bit_data (sqluint8 a_for_bit_data)
```

Argumentos de entrada

Tabla 54. Argumentos de entrada para la función de miembro set_for_bit_data

Nombre	Tipo de datos	Descripción
a_for_bit_data	sqluint8	Distintivo FOR BIT DATA (el distintivo debe ser 'Y' o 'N').

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función get_for_bit_data

Finalidad

Recuperar el distintivo FOR BIT DATA para la columna, si el distintivo es válido.

Sintaxis

```
sqlint32 get_for_bit_data (sqluint8* a_for_bit_data)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 55. Argumentos de salida para la función de miembro get_for_bit_data

Nombre	Tipo de datos	Descripción
a_for_bit_data	sqluint8*	El distintivo, que debe ser 'Y' o 'N'.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el distintivo.

Función set_column_ID

Finalidad

Establecer el ID (la posición) para la columna.

Uso No utilice esta función de miembro para cambiar el orden de las columnas en un apodo.

Sintaxis

```
sqlint32 set_column_ID (sqlint16 a_column_ID)
```

Argumentos de entrada

Tabla 56. Argumentos de entrada para la función de miembro set_column_ID

Nombre	Tipo de datos	Descripción
a_column_ID	sqlint16	ID (posición) de columna.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_column_ID

Finalidad
Recuperar el ID (la posición) de la columna, si el ID es válido.

Sintaxis

```
sqlint32 get_column_ID (sqluint16* a_column_ID)
```

Argumentos de entrada
Ninguno.

Argumentos de salida

Tabla 57. Argumentos de salida para la función de miembro get_column_ID

Nombre	Tipo de datos	Descripción
a_column_ID	sqluint16*	ID de columna.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el ID.

Función set_org_length

Finalidad
Establecer la longitud máxima (en bytes) para la columna.

Sintaxis

```
void set_org_length (sqlint32 a_org_length)
```

Argumentos de entrada

Tabla 58. Argumentos de entrada para la función de miembro set_org_length

Nombre	Tipo de datos	Descripción
a_org_length	sqlint32	Longitud máxima de la columna.

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función get_org_length

Finalidad

Recuperar la longitud de columna máxima (en bytes) de Column_Info, si el valor es válido.

Sintaxis

```
sqlint32 get_org_length (sqlint32* a_org_length)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 59. Argumentos de salida para la función de miembro get_org_length

Nombre	Tipo de datos	Descripción
a_org_length	sqlint32*	Longitud máxima de la columna.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función set_org_scale

Finalidad

Establecer la escala numérica para la columna.

Sintaxis

```
void set_org_scale (sqlint16 a_org_scale)
```

Argumentos de entrada

Tabla 60. Argumentos de entrada para la función de miembro set_org_scale

Nombre	Tipo de datos	Descripción
a_org_scale	sqlint16	Escala.

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función get_org_scale

Finalidad

Recuperar la escala numérica de Column_Info, si el valor es válido.

Sintaxis

```
sqlint32 get_org_scale (sqlint16* a_org_scale)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 61. Argumentos de salida para la función de miembro get_org_scale

Nombre	Tipo de datos	Descripción
a_org_scale	sqlint16*	Escala numérica.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función set_nulls

Finalidad

Establecer el distintivo de nulos permitidos para la columna.

Sintaxis

```
void set_nulls (sqluint8 a_nulls)
```

Argumentos de entrada

Tabla 62. Argumentos de entrada para la función de miembro set_nulls

Nombre	Tipo de datos	Descripción
a_nulls	sqluint8	Distintivo de nulos permitidos. El distintivo debe ser 'Y' o 'N'.

Argumentos de salida
Ninguno.

Valor de retorno
Ninguno.

Función get_nulls

Finalidad
Recuperar el distintivo de nulos permitidos de Column_Info, si el valor es válido.

Sintaxis

```
sqlint32 get_nulls (sqluint8* a_nulls)
```

Argumentos de entrada
Ninguno.

Argumentos de salida

Tabla 63. Argumentos de salida para la función de miembro get_nulls

Nombre	Tipo de datos	Descripción
a_nulls	sqluint8*	Distintivo de nulos permitidos de la columna.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función set_avg_length

Finalidad
Establecer el promedio de longitud (en bytes) para la columna.

Uso El reiniciador establece el promedio de longitud de una columna durante el proceso de la sentencia CREATE NICKNAME o ALTER

NICKNAME. El optimizador de DB2 utiliza esta información de promedio de longitud cuando desarrolla un plan de optimización de consultas.

Sintaxis

```
void set_avg_len (sqlint32 a_avg_len)
```

Argumentos de entrada

Tabla 64. Argumentos de entrada para la función de columna set_avg_length

Nombre	Tipo de datos	Descripción
a_avg_len	sqlint32	Promedio de longitud de la columna.

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función get_avg_length

Finalidad

Recuperar el promedio de longitud de columna (en bytes) de Column_Info, si el valor es válido.

Sintaxis

```
sqlint32 get_avg_length (sqlint32* a_avg_len)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 65. Argumentos de salida para la función de miembro get_avg_length

Nombre	Tipo de datos	Descripción
a_avg_len	sqlint32*	Promedio de longitud de la columna.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función set_high2key

Finalidad

Establecer el segundo valor más alto para la columna.

Uso El reiniciador puede establecer el segundo valor más alto de una columna durante el proceso de la sentencia CREATE NICKNAME o ALTER NICKNAME. El optimizador de DB2 puede utilizar este segundo valor más alto o el valor más alto cuando desarrolla un plan de optimización de consultas.

Sintaxis

```
sqlint32 set_high2key (sqluint8* a_high2key,
                       sqlint32  a_high2key_len)
```

Argumentos de entrada

Tabla 66. Argumentos de entrada para la función de miembro set_high2key

Nombre	Tipo de datos	Descripción
a_high2key	sqluint8*	Segundo valor más alto de serie de caracteres (no terminado por nulo).
a_high2key_len	sqlint32	Longitud del valor.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_high2key

Finalidad

Recuperar el segundo valor más alto de Column_Info, si el valor es válido.

Sintaxis

```
sqlint32 get_high2key (sqluint8** a_high2key)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 67. Argumentos de salida para la función de miembro get_high2key

Nombre	Tipo de datos	Descripción
a_high2key	sqluint8**	Segundo valor más alto de la columna.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función set_low2key

Finalidad

Establecer el segundo valor más bajo para la columna.

Uso El reiniciador puede establecer el segundo valor más bajo de una columna durante el proceso de la sentencia CREATE NICKNAME o ALTER NICKNAME. El optimizador de DB2 puede utilizar este segundo valor más bajo o el valor más bajo cuando desarrolla un plan de optimización de consultas.

Sintaxis

```
sqlint32 set_low2key (sqluint8* a_low2key,
                      sqlint32  a_low2key_len)
```

Argumentos de entrada

Tabla 68. Argumentos de entrada para la función de miembro set_low2key

Nombre	Tipo de datos	Descripción
a_low2key	sqluint8*	Segundo valor más bajo de serie de caracteres (no terminado por nulo).
a_low2key_len	sqlint32	Longitud del valor.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_low2key

Finalidad

Recuperar el segundo valor más bajo de Column_Info, si el valor es válido.

Sintaxis

```
sqlint32 get_low2key (sqluint8** a_low2key)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 69. Argumentos de salida para la función de miembro get_low2key

Nombre	Tipo de datos	Descripción
a_low2key	sqluint8**	Segundo valor más bajo de la columna.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función get_default

Finalidad

Recuperar el valor por omisión de Column_Info, si el valor es válido.

Sintaxis

```
sqlint32 get_default (sqluint8** a_default)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 70. Argumentos de salida para la función de miembro get_default

Nombre	Tipo de datos	Descripción
a_default	sqluint8**	Valor por omisión de la columna.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función set_colcard

Finalidad

Establecer la cardinalidad para la columna.

Uso El reiniciador establece la cardinalidad de la columna (si se conoce) durante el proceso de la sentencia CREATE NICKNAME o ALTER NICKNAME. El optimizador de DB2 utiliza esta información cuando genera un plan de rendimiento óptimo. Para las columnas con valores diferenciados (no duplicados), la cardinalidad de la columna debe ser igual que la cardinalidad del apodo. El optimizador de DB2 genera un error si la cardinalidad de la columna es mayor que la cardinalidad del apodo.

Sintaxis

```
void set_colcard (sqlint64 a_colcard)
```

Argumentos de entrada

Tabla 71. Argumentos de entrada para la función de miembro set_colcard

Nombre	Tipo de datos	Descripción
a_colcard	sqlint64	Cardinalidad de la columna.

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función get_colcard

Finalidad

Recuperar la cardinalidad de Column_Info, si el valor es válido.

Sintaxis

```
sqlint32 get_colcard (sqlint64* a_colcard)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 72. Argumentos de salida para la función de miembro get_colcard

Nombre	Tipo de datos	Descripción
a_colcard	sqlint64*	Cardinalidad de la columna.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función set_codepage1

Finalidad

Establecer la página de códigos para la columna.

Sintaxis

```
void set_codepage1 (sqlint16 a_codepage1)
```

Argumentos de entrada

Tabla 73. Argumentos de entrada para la función de miembro set_codepage1

Nombre	Tipo de datos	Descripción
a_codepage1	sqlint16	Página de códigos de la columna.

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función get_codepage1

Finalidad

Recuperar la página de códigos de Column_Info, si el valor es válido.

Sintaxis

```
sqlint32 get_codepage1 (sqlint16* a_codepage1)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 74. Argumentos de salida para la función de miembro get_codepage1

Nombre	Tipo de datos	Descripción
a_codepage1	sqlint16*	Página de códigos de la columna.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función set_codepage2

Finalidad

Establecer la página de códigos para la columna.

Sintaxis

```
void set_codepage2 (sqlint16 a_codepage2)
```

Argumentos de entrada

Tabla 75. Argumentos de entrada para la función de miembro set_codepage2

Nombre	Tipo de datos	Descripción
a_codepage2	sqlint16	Página de códigos de la columna.

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función get_codepage2

Finalidad

Recuperar la página de códigos de Column_Info, si el valor es válido.

Sintaxis

```
sqlint32 get_codepage2 (sqlint16* a_codepage2)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 76. Argumentos de salida para la función de miembro get_codepage2

Nombre	Tipo de datos	Descripción
a_codepage2	sqlint16*	Página de códigos de la columna.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función merge

Finalidad

Fusionar un objeto delta Column_Info en el objeto actual.

Sintaxis

```
sqlint32 merge (Column_Info* a_delta_info)
```

Argumentos de entrada

Tabla 77. Argumentos de entrada para la función de miembro merge

Nombre	Tipo de datos	Descripción
a_delta_info	Column_Info*	Datos que se deben fusionar.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función copy

Finalidad

Duplicar un objeto Column_Info.

Sintaxis

```
sqlint32 copy (Column_Info** a_new_column_info)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 78. Argumentos de salida para la función de miembro copy

Nombre	Tipo de datos	Descripción
a_new_column_info	Column_Info**	Puntero al objeto duplicado.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función add_option

Dónde se define

Catalog_Info

Finalidad

Añadir una opción de un solo valor a la información de catálogo.

Uso El reiniciador invoca esta función de miembro cuando se añaden opciones generadas por el reiniciador al catálogo durante el proceso de la sentencia CREATE NICKNAME o ALTER NICKNAME.

Sintaxis

```
sqlint32 add_option (sqluint8*   a_opt_name,
                     sqlint32    a_opt_name_len,
                     sqluint8*   a_opt_value,
                     sqlint32    a_opt_value_len,
          Catalog_Option::Action  a_act
                    = Catalog_Option::sqlqg_None,
                     char*       a_opt_type = "")
```

Argumentos de entrada

Tabla 79. Argumentos de entrada para la función de miembro add_option

Nombre	Tipo de datos	Descripción
a_opt_name	sqluint8*	Nombre de opción (no terminado por nulo).
a_opt_name_len	sqlint32	Longitud del nombre de opción.
a_opt_value	sqluint8*	Valor de opción (no terminado por nulo).
a_opt_value_len	sqlint32	Longitud del valor de opción.
a_act	Catalog_Option::Action	Acción para esta opción (ADD, SET, DROP o ninguna).
a_opt_type	char*	Señal que se utiliza en el mensaje de error SQLN1884 si se trata de una opción duplicada. Utilice la constante SQLQG_COLUMN_OPTION que está definida en el archivo de cabecera sqlqg_misc.h.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función drop_option

Dónde se define
Catalog_Info

Finalidad
Suprimir una opción de una clase Column_Info. Esta función de miembro no elimina la opción del catálogo. La opción se elimina cuando se añade una opción para un objeto delta Catalog_Info con una acción de Catalog_Option::sqlqg_Drop.

Sintaxis

```
sqlint32 drop_option (Catalog_Option* a_option)
```

Argumentos de entrada

Tabla 80. Argumentos de entrada para la función de miembro drop_option

Nombre	Tipo de datos	Descripción
a_option	Catalog_Option*	Opción que se debe eliminar.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_option

Dónde se define
Catalog_Info

Finalidad
Recuperar una opción de catálogo por nombre.

Uso El argumento de salida `a_option` es nulo si no se encuentra la opción.

Sintaxis

```
sqlint32 get_option (sqluint8*        a_opt_name,
                     sqlint32         a_name_len,
                     Catalog_Option** a_option)
```

Argumentos de entrada

Tabla 81. Argumentos de entrada para la función de miembro get_option

Nombre	Tipo de datos	Descripción
a_opt_name	sqluint8*	Nombre de opción (no terminado por nulo).
a_name_len	sqlint32	Longitud del nombre de opción.

Argumentos de salida

Tabla 82. Argumentos de salida para la función de miembro get_option

Nombre	Tipo de datos	Descripción
a_option	Catalog_Option**	Opción encontrada.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha encontrado la opción.

Función get_first_option

Dónde se define

Catalog_Info

Finalidad

Recuperar un puntero a la primera opción de la cadena de opciones.

Sintaxis

```
Catalog_Option* get_first_option ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Puntero a la primera opción de la cadena. El valor es nulo si la cadena está vacía.

Función get_next_option

Dónde se define
Catalog_Info

Finalidad
Recuperar la siguiente opción de la cadena de opciones.

Sintaxis

```
Catalog_Option* get_next_option (Catalog_Option* a_current_option)
```

Argumentos de entrada

Tabla 83. Argumentos de entrada para la función de miembro get_next_option

Nombre	Tipo de datos	Descripción
a_current_option	Catalog_Option*	Opción actual.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la siguiente opción de la cadena. El valor es nulo si está al final.

Consulta relacionada:

- "Clases de catálogo para la API C++" en la página 1

Clase Nickname_Info

Este tema describe la clase Nickname_Info y proporciona detalles para el constructor y las funciones de miembro.

Visión general

La clase Nickname_Info encapsula una definición de apodo del catálogo e incluye definiciones de columna.

La clase Nickname_Info es una de las clases de catálogo para la API C++.

Uso El servidor federado DB2 crea una instancia de esta clase para que contenga información de una sentencia CREATE NICKNAME o ALTER NICKNAME o para que contenga información del catálogo de DB2 Information Integrator. El reiniciador crea una instancia de esta clase cuando se añade información durante las operaciones de sentencia CREATE NICKNAME o ALTER NICKNAME.

Archivo
sqlqg_catalog.h

Miembros de datos
Ninguno.

Constructores y funciones de miembro

Las tablas siguientes describen el constructor y las funciones de miembro de la clase Nickname_Info. El constructor y las funciones se describen más detalladamente después de las tablas.

Tabla 84. Constructores para la clase Nickname_Info

Constructor	Descripción
Nickname_Info	Construir un objeto Nickname_Info (vacío) por omisión.

Tabla 85. Funciones de miembro para la clase Nickname_Info

Función de miembro	Descripción
get_local_schema	Recuperar el nombre de esquema local para el apodo.
get_nickname	Recuperar el nombre local para el apodo.
get_server_name	Recuperar el nombre de servidor para el apodo.
set_card	Establecer la cardinalidad para el apodo.
get_card	Recuperar la cardinalidad de apodo, si el valor es válido.
set_npages	Establecer el valor de npages para el apodo.
get_npages	Recuperar las estadísticas de npages de apodo, si el valor es válido.
set_fpages	Establecer las estadísticas de fpages para el apodo.
get_fpages	Recuperar las estadísticas de fpages de apodo, si el valor es válido.
set_overflow	Establecer las estadísticas de desbordamiento (overflow) para el apodo.
get_overflow	Recuperar las estadísticas de desbordamiento de apodo, si el valor es válido.
insert_column	Añadir una definición de columna a la información de apodo.
replace_column	Sustituir una definición de columna en la información de apodo.

Tabla 85. Funciones de miembro para la clase Nickname_Info (continuación)

Función de miembro	Descripción
get_number_columns	Recuperar el número de columnas para el apodo.
get_first_column	Recuperar un puntero al primer objeto Column_Info para el apodo.
get_next_column	Recuperar un puntero al siguiente objeto Column_Info.
get_column	Recuperar una descripción de columna por nombre (local).
merge	Fusionar un objeto delta Nickname_Info en el objeto actual.
copy	Duplicar un objeto Nickname_Info.
add_option	Añadir una opción de un solo valor a la información de catálogo.
drop_option	Suprimir una opción de una clase Catalog_Info.
get_option	Recuperar una opción de catálogo por nombre.
get_first_option	Recuperar un puntero a la primera opción de la cadena de opciones.
get_next_option	Recuperar la siguiente opción de la cadena de opciones.

Constructor Nickname_Info

Finalidad

Construir un objeto Nickname_Info (vacío) por omisión.

Sintaxis

```
Nickname_Info ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función get_local_schema

Finalidad

Recuperar el nombre de esquema local para el apodo.

Sintaxis

```
sqlint32 get_local_schema (sqluint8** a_local_schema)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 86. Argumentos de salida para la función de miembro get_local_schema

Nombre	Tipo de datos	Descripción
a_local_schema	sqluint8**	Puntero al nombre de esquema local terminado por nulo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función get_nickname

Finalidad

Recuperar el nombre local para el apodo.

Sintaxis

```
sqlint32 get_nickname (sqluint8** a_nickname)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 87. Argumentos de salida para la función de miembro get_nickname

Nombre	Tipo de datos	Descripción
a_nickname	sqluint8**	Puntero al nombre local terminado por nulo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función get_server_name

Finalidad

Recuperar el nombre de servidor para el apodo.

Sintaxis

```
sqlint32 get_server_name (sqluint8** a_server_name)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 88. Argumentos de salida para la función de miembro get_server_name

Nombre	Tipo de datos	Descripción
a_server_name	sqluint8**	Puntero al nombre de servidor terminado por nulo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función set_card

Finalidad

Establecer la cardinalidad para el apodo.

Uso El reiniciador puede invocar esta función de miembro durante el proceso de la sentencia CREATE NICKNAME o ALTER NICKNAME con el fin de especificar una cardinalidad inicial para un apodo. Se puede utilizar el método set_colcard del objeto Column_Info para establecer cardinalidades de columna individuales. Para columnas con valores exclusivos, la cardinalidad de columna debe ser igual a la cardinalidad de apodo. La cardinalidad de columna no puede ser mayor que la cardinalidad de nombre.

Sintaxis

```
void set_card (sqlint64 a_card)
```

Argumentos de entrada

Tabla 89. Argumentos de entrada para la función de miembro set_card

Nombre	Tipo de datos	Descripción
a_card	sqlint64	Cardinalidad para el apodo.

Argumentos de salida
Ninguno.

Valor de retorno
Ninguno.

Función get_card

Finalidad
Recuperar la cardinalidad de apodo, si el valor es válido.

Sintaxis

```
sqlint32 get_card (sqlint64* a_card)
```

Argumentos de entrada
Ninguno.

Argumentos de salida

Tabla 90. Argumentos de salida para la función de miembro get_card

Nombre	Tipo de datos	Descripción
a_card	sqlint64*	Cardinalidad de apodo.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función set_npages

Finalidad
Establecer el valor de npages para el apodo.

Sintaxis

```
void set_npages (sqlint32 a_npages)
```

Argumentos de entrada

Tabla 91. Argumentos de entrada para la función de miembro set_npages

Nombre	Tipo de datos	Descripción
a_npages	sqlint32	Valor de npages para el apodo.

Argumentos de salida
Ninguno.

Valor de retorno
Ninguno.

Función get_npages

Finalidad
Recuperar las estadísticas de npages de apodo, si el valor es válido.

Sintaxis

```
sqlint32 get_npages (sqlint32* a_npages)
```

Argumentos de entrada
Ninguno.

Argumentos de salida

Tabla 92. Argumentos de salida para la función de miembro get_npages

Nombre	Tipo de datos	Descripción
a_npages	sqlint32*	Estadísticas de npages de apodo.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función set_fpages

Finalidad
Establecer las estadísticas de fpages para el apodo.

Sintaxis

```
void set_fpages (sqlint32 a_fpages)
```

Argumentos de entrada

Tabla 93. Argumentos de entrada para la función de miembro set_fpages

Nombre	Tipo de datos	Descripción
a_fpages	sqlint32	Estadísticas de fpages para el apodo.

Argumentos de salida
Ninguno.

Valor de retorno
Ninguno.

Función get_fpages

Finalidad

Recuperar las estadísticas de fpages de apodo, si el valor es válido.

Sintaxis

```
sqlint32 get_fpages (sqlint32* a_fpages)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 94. Argumentos de salida para la función de miembro get_fpages

Nombre	Tipo de datos	Descripción
a_fpages	sqlint32*	Estadísticas de fpages de apodo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función set_overflow

Finalidad

Establecer las estadísticas de desbordamiento (overflow) para el apodo.

Sintaxis

```
void set_overflow (sqlint32 a_overflow)
```

Argumentos de entrada

Tabla 95. Argumentos de entrada para la función de miembro set_overflow

Nombre	Tipo de datos	Descripción
a_overflow	sqlint32	Estadísticas de desbordamiento para el apodo.

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función get_overflow

Finalidad

Recuperar las estadísticas de desbordamiento de apodo, si el valor es válido.

Sintaxis

```
sqlint32 get_overflow (sqlint32* a_overflow)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 96. Argumentos de salida para la función de miembro get_overflow

Nombre	Tipo de datos	Descripción
a_overflow	sqlint32*	Estadísticas de desbordamiento de apodo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha establecido el valor.

Función insert_column

Finalidad

Añadir una definición de columna a la información de apodo.

Uso El objeto Column_Info debe estar asignado en la pila (utilizando new). Esta rutina toma el control del puntero. Si se establece el ID de columna para el argumento a_new_col_info, se produce un error.

Sintaxis

```
sqlint32 insert_column (Column_Info* a_new_col_info)
```

Argumentos de entrada

Tabla 97. Argumentos de entrada para la función de miembro insert_column

Nombre	Tipo de datos	Descripción
a_new_col_info	Column_Info*	Objeto Column_Info que se debe añadir.

Argumentos de salida

Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función replace_column

Finalidad
Sustituir una definición de columna en la información de apodo. Dos definiciones de columna se consideran iguales si utilizan el mismo ID.

Uso Si no se establece el ID de columna en el argumento a_new_col_info, se produce un error.

Sintaxis

```
sqlint32 replace_column (Column_Info* a_new_col_info)
```

Argumentos de entrada

Tabla 98. Argumentos de entrada para la función de miembro replace_column

Nombre	Tipo de datos	Descripción
a_new_col_info	Column_Info*	Column_Info que se debe sustituir.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_number_columns

Finalidad
Recuperar el número de columnas para el apodo.

Sintaxis

```
sqlint16 get_number_columns ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Número de columnas.

Función get_first_column

Finalidad

Recuperar un puntero al primer objeto Column_Info para el apodo.

Uso Las columnas se mantienen ordenadas por el ID.

Sintaxis

```
Column_Info* get_first_column ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Puntero al primer objeto Column_Info. El valor es nulo si la lista está vacía.

Función get_next_column

Finalidad

Recuperar un puntero al siguiente objeto Column_Info.

Uso Las columnas se mantienen ordenadas por el ID.

Sintaxis

```
Column_Info* get_next_column (Column_Info* a_cur_col_info)
```

Argumentos de entrada

Tabla 99. Argumentos de entrada para la función de miembro get_next_column

Nombre	Tipo de datos	Descripción
a_cur_col_info	Column_Info*	Columna actual.

Argumentos de salida

Ninguno.

Valor de retorno

Puntero al siguiente Column_Info. El valor es nulo si está al final de la lista.

Función get_column

Finalidad

Recuperar una descripción de columna por nombre (local).

Sintaxis

```
sqlint32 get_column (sqluint8*     a_col_name,
                     sqlint32      a_col_name_len,
                     Column_Info** a_col_info)
```

Argumentos de entrada

Tabla 100. Argumentos de entrada para la función de miembro get_column

Nombre	Tipo de datos	Descripción
a_col_name	sqluint8*	Nombre de columna (no terminado por nulo).
a_col_name_len	sqlint32	Longitud del nombre de columna.

Argumentos de salida

Tabla 101. Argumentos de salida para la función de miembro get_column

Nombre	Tipo de datos	Descripción
a_col_info	Column_Info**	Puntero al objeto Column_Info.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha encontrado el nombre de columna.

Función merge

Finalidad

Fusionar un objeto delta Nickname_Info en el objeto actual.

Sintaxis

```
sqlint32 merge (Nickname_Info* a_delta_info)
```

Argumentos de entrada

Tabla 102. Argumentos de entrada para la función de miembro merge

Nombre	Tipo de datos	Descripción
a_delta_info	Nickname_Info*	Datos que se deben fusionar.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función copy

Finalidad

Duplicar un objeto Nickname_Info.

Sintaxis

```
sqlint32 copy (Nickname_Info** a_new_idx_info)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 103. Argumentos de salida para la función de miembro copy

Nombre	Tipo de datos	Descripción
a_new_idx_info	Nickname_Info**	Puntero al objeto duplicado.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función add_option

Dónde se define

Catalog_Info

Finalidad

Añadir una opción de un solo valor a la información de catálogo.

Uso El reiniciador puede invocar esta función de miembro cuando se añaden opciones generadas por el reiniciador al catálogo durante el proceso de la sentencia CREATE NICKNAME o ALTER NICKNAME.

Sintaxis

```
sqlint32 add_option (sqluint8*   a_opt_name,
                     sqlint32    a_opt_name_len,
                     sqluint8*   a_opt_value,
                     sqlint32    a_opt_value_len,
          Catalog_Option::Action a_act
                     = Catalog_Option::sqlqg_None,
                     char*       a_opt_type = "")
```

Argumentos de entrada

Tabla 104. Argumentos de entrada para la función de miembro add_option

Nombre	Tipo de datos	Descripción
a_opt_name	sqluint8*	Nombre de opción (no terminado por nulo).
a_opt_name_len	sqlint32	Longitud del nombre de opción.
a_opt_value	sqluint8*	Valor de opción (no terminado por nulo).
a_opt_value_len	sqlint32	Longitud del valor de opción.
a_act	Catalog_Option::Action	Acción para esta opción (ADD, SET, DROP o ninguna).
a_opt_type	char*	Señal que se debe utilizar en el mensaje de error SQLN1884 si se trata de una opción duplicada. Utilice la constante SQLQG_NICKNAME_OPTION que está definida en el archivo de cabecera sqlqg_misc.h.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función drop_option

Dónde se define

Catalog_Info

Finalidad

Suprimir una opción de una clase Catalog_Info.

Uso Esta función de miembro no elimina la opción del catálogo. La opción se elimina del catálogo cuando se añade una opción para un objeto delta Catalog_Info con una acción de Catalog_Option::sqlqg_Drop.

Sintaxis

```
sqlint32 drop_option (Catalog_Option* a_option)
```

Argumentos de entrada

Tabla 105. Argumentos de entrada para la función de miembro drop_option

Nombre	Tipo de datos	Descripción
a_option	Catalog_Option*	Opción que se debe eliminar.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_option

Dónde se define

Catalog_Info

Finalidad

Recuperar una opción de catálogo por nombre.

Uso El argumento de salida `a_option` es nulo si no se encuentra la opción.

Sintaxis

```
sqlint32 get_option (sqluint8*        a_opt_name,
                     sqlint32         a_name_len,
                     Catalog_Option** a_option)
```

Argumentos de entrada

Tabla 106. Argumentos de entrada para la función de miembro get_option

Nombre	Tipo de datos	Descripción
a_opt_name	sqluint8*	Nombre de opción (no terminado por nulo).
a_name_len	sqlint32	Longitud del nombre de opción.

Argumentos de salida

Tabla 107. Argumentos de salida para la función de miembro get_option

Nombre	Tipo de datos	Descripción
a_option	Catalog_Option**	Opción encontrada.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. SQLQG_NOVALUE indica que no se ha encontrado la opción.

Función get_first_option

Dónde se define
Catalog_Info

Finalidad
Recuperar un puntero a la primera opción de la cadena de opciones.

Sintaxis

```
Catalog_Option* get_first_option ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la primera opción de la cadena. El valor es nulo si la cadena está vacía.

Función get_next_option

Dónde se define
Catalog_Info

Finalidad
Recuperar la siguiente opción de la cadena de opciones.

Sintaxis

```
Catalog_Option* get_next_option (Catalog_Option* a_current_option)
```

Argumentos de entrada

Tabla 108. Argumentos de entrada para la función de miembro get_next_option

Nombre	Tipo de datos	Descripción
a_current_option	Catalog_Option*	Opción actual.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la siguiente opción de la cadena. El valor es nulo si está al final.

Consulta relacionada:
- "Clases de catálogo para la API C++" en la página 1

Clases de reiniciador para la API C++

La tabla siguiente describe cada clase de reiniciador para la API C++.

Tabla 109. Clases de reiniciador

Nombre de clase	Descripción
Unfenced_Generic_Wrapper	Esta clase representa el reiniciador en el espacio de proceso desprotegido (fiable) y es responsable de la verificación de los datos de catálogo.
Fenced_Generic_Wrapper	Esta clase representa el reiniciador en el espacio de proceso protegido.

Consulta relacionada:
- "Clase Unfenced_Generic_Wrapper" en la página 70
- "Clase Fenced_Generic_Wrapper" en la página 79

Clase Unfenced_Generic_Wrapper

Este tema describe la clase Unfenced_Generic_Wrapper y proporciona detalles para el constructor, el destructor y las funciones de miembro.

Visión general

La clase Unfenced_Generic_Wrapper representa el reiniciador en el espacio de proceso desprotegido (fiable) y verifica los datos de catálogo.

La clase Unfenced_Generic_Wrapper es una de las clases de reiniciador para la API C++.

Uso El reiniciador debe implementar una subclase de la clase Unfenced_Generic_Wrapper. La función UnfencedWrapper_Hook específica de reiniciador crea una instancia de esta clase.

Archivo
sqlqg_unfenced_generic_wrapper.h

Miembros de datos

La tabla siguiente lista los miembros de datos que puede utilizar con la clase Unfenced_Generic_Wrapper.

Tabla 110. Miembros de datos para la clase Unfenced_Generic_Wrapper

Nombre	Tipo de datos	Descripción
m_info	Wrapper_Info*	Contiene la información de catálogo para el reiniciador.
m_name	sqluint8*	Puntero a la serie de caracteres terminada por nulo que contiene el nombre del reiniciador.
m_corelib	sqluint8*	Puntero a la serie de caracteres terminada por nulo que contiene el nombre de la biblioteca de reiniciador. Este nombre es la biblioteca de reiniciador que se utiliza en la sentencia CREATE WRAPPER. Este nombre no corresponde a la biblioteca que se utiliza durante el proceso de sentencia.
m_version	sqlint32	Versión del código de reiniciador.

Constructores, destructores y funciones de miembro

Las tablas siguientes describen el constructor, el destructor y las funciones de miembro de la clase Unfenced_Generic_Wrapper. El constructor, el destructor y las funciones se describen más detalladamente después de las tablas.

Tabla 111. Constructores para la clase Unfenced_Generic_Wrapper

Constructor	Descripción
Unfenced_Generic_Wrapper	Construir una instancia de la clase Unfenced_Generic_Wrapper.

Tabla 112. Destructores para la clase Unfenced_Generic_Wrapper

Destructor	Descripción
~Unfenced_Generic_Wrapper	Destructor para la clase Unfenced_Generic_Wrapper.

Tabla 113. Funciones de miembro para la clase Unfenced_Generic_Wrapper

Función de miembro	Descripción
verify_my_register_wrapper_info	Verificar la información de catálogo cuando se somete la sentencia CREATE WRAPPER.
verify_my_alter_wrapper_info	Verificar la información de catálogo cuando se somete la sentencia ALTER WRAPPER.
get_name	Devolver un puntero a la serie de caracteres terminada por nulo que contiene el nombre de reiniciador.
get_corelib	Devolver un puntero a la serie de caracteres terminada por nulo que contiene el nombre de biblioteca de reiniciador.
get_version	Devolver la versión del código de reiniciador.
get_info	Devolver un puntero al objeto de información de catálogo de reiniciador.
initialize_my_wrapper	Inicializar el estado de objeto de reiniciador a partir del objeto de información de catálogo.
create_server	Crear una instancia de la subclase apropiada de Servidor para el reiniciador.

Constructor Unfenced_Generic_Wrapper

Finalidad

Construir una instancia de Unfenced_Generic_Wrapper.

Sintaxis

```
Unfenced_Generic_Wrapper (sqlint32* a_rc,
                          sqlint32  a_wrapper_version = 0)
```

Argumentos de entrada

Tabla 114. Argumentos de entrada para el constructor Unfenced_Generic_Wrapper

Nombre	Tipo de datos	Descripción
a_wrapper_version	sqlint32	Versión del código de reiniciador.

Argumentos de salida

Tabla 115. Argumentos de entrada para el constructor Unfenced_Generic_Wrapper

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Puntero al retorno. El valor debe ser 0 para que la operación se realice satisfactoriamente.

Valor de retorno

Ninguno.

Destructor ~Unfenced_Generic_Wrapper

Finalidad

Destructor para la clase Unfenced_Generic_Wrapper.

Sintaxis

```
~Unfenced_Generic_Wrapper ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función verify_my_register_wrapper_info

Finalidad

Verificar la información de catálogo cuando se somete la sentencia CREATE WRAPPER.

Uso El reiniciador puede implementar esta función de miembro en la subclase específica de reiniciador de Unfenced_Generic_Wrapper si se soportan opciones de reiniciador específicas de reiniciador.

El reiniciador comprueba si se ha asignado un objeto delta antes de asignar uno él mismo.

Sintaxis

```
virtual sqlint32 verify_my_register_wrapper_info
                (Wrapper_Info*  a_wrapper_info,
                 Wrapper_Info** a_delta_info)
```

Argumentos de entrada

Tabla 116. Argumentos de entrada para la función de miembro verify_my_register_wrapper_info

Nombre	Tipo de datos	Descripción
a_wrapper_info	Wrapper_Info*	Información de la sentencia CREATE WRAPPER.

Argumentos de salida

Tabla 117. Argumentos de salida para la función de miembro verify_my_register_wrapper_info

Nombre	Tipo de datos	Descripción
a_delta_info	Wrapper_Info**	Información añadida por verify_my.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función verify_my_alter_wrapper_info

Finalidad

Verificar la información de catálogo cuando se somete la sentencia ALTER WRAPPER.

Uso

El reiniciador puede implementar esta función de miembro en la subclase específica de reiniciador de Unfenced_Generic_Wrapper si se soportan opciones de reiniciador específicas de reiniciador.

El reiniciador comprueba si se ha asignado un objeto delta antes de asignar uno él mismo.

Sintaxis

```
virtual sqlint32 verify_my_alter_wrapper_info
                 (Wrapper_Info*  a_wrapper_info,
                  Wrapper_Info** a_delta_info)
```

Argumentos de entrada

Tabla 118. Argumentos de entrada para la función de miembro verify_my_alter_wrapper_info

Nombre	Tipo de datos	Descripción
a_wrapper_info	Wrapper_Info*	Información de la sentencia ALTER WRAPPER.

Argumentos de salida

Tabla 119. Argumentos de salida para la función de miembro verify_my_alter_wrapper_info

Nombre	Tipo de datos	Descripción
a_delta_info	Wrapper_Info**	Información añadida por verify_my.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_name

Dónde se define

Reiniciador

Finalidad

Devolver un puntero a la serie de caracteres terminada por nulo que contiene el nombre de reiniciador.

Sintaxis

```
sqluint8* get_name()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Puntero a la serie de caracteres.

Función get_corelib

Dónde se define

Reiniciador

Finalidad

Devolver un puntero a la serie de caracteres terminada por nulo que contiene el nombre de biblioteca de reiniciador. Este nombre es la biblioteca de reiniciador que se utiliza en la sentencia CREATE WRAPPER. Este nombre no corresponde a la biblioteca que se utiliza durante el proceso de sentencia.

Sintaxis

```
sqluint8* get_corelib()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la serie de caracteres.

Función get_version

Dónde se define
Reiniciador

Finalidad
Devolver la versión del código de reiniciador.

Sintaxis

```
sqlint32 get_version()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Versión de reiniciador.

Función get_info

Dónde se define
Reiniciador

Finalidad
Devolver un puntero al objeto de información de catálogo de reiniciador.

Sintaxis

```
Wrapper_Info* get_info()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al objeto de información de catálogo.

Función initialize_my_wrapper

Dónde se define
Reiniciador

Finalidad
Inicializar el estado de objeto de reiniciador a partir del objeto de información de catálogo. La versión por omisión no realiza ninguna acción.

Uso Esta función de miembro se puede implementar en la subclase específica de reiniciador de Unfenced_Generic_Wrapper si se soportan opciones de reiniciador específicas de reiniciador.

Sintaxis

```
virtual sqlint32 initialize_my_wrapper (Wrapper_Info* a_wrapper_info)
```

Argumentos de entrada

Tabla 120. Argumentos de entrada para la función de miembro initialize_my_wrapper

Nombre	Tipo de datos	Descripción
a_wrapper_info	Wrapper_Info*	Objeto de información de catálogo.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función create_server

Dónde se define
Reiniciador

Finalidad
Crear una instancia de la subclase apropiada de Servidor para el reiniciador.

Uso El reiniciador debe implementar esta función de miembro en la subclase específica de reiniciador de Unfenced_Generic_Wrapper.

Sintaxis

```
virtual Server* create_server (sqluint8* a_server_name,
                               sqlint32* a_rc)
```

Argumentos de entrada

Tabla 121. Argumentos de entrada para la función de miembro create_server

Nombre	Tipo de datos	Descripción
a_server_name	sqluint8*	Nombre terminado por nulo del servidor.

Argumentos de salida

Tabla 122. Argumentos de salida para la función de miembro create_server

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Puntero al código de retorno; 0 indica que la operación se ha realizado satisfactoriamente.

Valor de retorno

Puntero al objeto de servidor recién creado o nulo.

Funciones asociadas

La clase Unfenced_Generic_Wrapper no utiliza la función UnfencedWrapper_Hook, pero esta función está directamente asociada con dicha clase.

Función UnfencedWrapper_Hook

Finalidad

Función de enganche que permite al servidor federado DB2 crear instancias de la subclase específica de reiniciador de Unfenced_Generic_Wrapper. El reiniciador debe implementar esta función y dicha función se debe exportar del módulo de reiniciador desprotegido.

Sintaxis

```
extern "C" UnfencedWraper* UnfencedWrapper_Hook()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Subclase de Unfenced_Generic_Wrapper de la que se han creado instancias o nulo si se produce un error.

Consulta relacionada:

• "Clases de reiniciador para la API C++" en la página 70

Clase Fenced_Generic_Wrapper

Este tema describe la clase Fenced_Generic_Wrapper y proporciona detalles para el constructor, el destructor y las funciones de miembro.

Visión general

La clase Fenced_Generic_Wrapper representa el reiniciador del espacio de proceso protegido.

La clase Fenced_Generic_Wrapper es una de las clases de reiniciador para la API C++.

Uso El reiniciador debe implementar una subclase de Fenced_Generic_Wrapper. La función FencedWrapper_Hook específica de reiniciador crea una instancia de esta clase.

Archivo

sqlqg_fenced_generic_wrapper.h

Miembros de datos

La tabla siguiente lista los miembros de datos que puede utilizar con la clase Fenced_Generic_Wrapper.

Tabla 123. Miembros de datos para la clase Fenced_Generic_Wrapper

Nombre	Tipo de datos	Descripción
m_info	Wrapper_Info*	Contiene la información de catálogo para el reiniciador.
m_name	sqluint8*	Puntero a la serie de caracteres que contiene el nombre del reiniciador.
m_corelib	sqluint8*	Puntero a la serie de caracteres que contiene el nombre de la biblioteca de reiniciador.
m_version	sqlint32	Versión del código de reiniciador.

Constructores, destructores y funciones de miembro

Las tablas siguientes describen el constructor, el destructor y las funciones de miembro de la clase Fenced_Generic_Wrapper. El constructor, el destructor y las funciones se describen más detalladamente después de las tablas.

Tabla 124. Constructores para la clase Fenced_Generic_Wrapper

Constructor	Descripción
FencedGeneric_Wrapper	Construir una instancia de la clase Fenced_Generic_Wrapper.

Tabla 125. Destructores para la clase Fenced_Generic_Wrapper

Destructor	Descripción
~FencedGeneric_Wrapper	Destructor para la clase Fenced_Generic_Wrapper.

Tabla 126. Funciones de miembro para la clase Fenced_Generic_Wrapper

Función de miembro	Descripción
get_name	Devolver un puntero a la serie de caracteres terminada por nulo que contiene el nombre de reiniciador.
get_corelib	Devolver un puntero a la serie de caracteres terminada por nulo que contiene el nombre de biblioteca de reiniciador.
get_version	Devolver la versión del código de reiniciador.
get_info	Devolver un puntero al objeto de información de catálogo de reiniciador.
initialize_my_wrapper	Inicializar el estado de objeto de reiniciador a partir del objeto de información de catálogo.
create_server	Crear una instancia de la subclase apropiada de Servidor para el reiniciador.

Constructor FencedGeneric_Wrapper

Finalidad

Construir una instancia de la clase Fenced_Generic_Wrapper.

Sintaxis

```
FencedGeneric_Wrapper (sqlint32* a_rc,
                       sqlint32  a_wrapper_version = 0)
```

Argumentos de entrada

Tabla 127. Argumentos de entrada para el constructor FencedGeneric_Wrapper

Nombre	Tipo de datos	Descripción
a_wrapper_version	sqlint32	Versión del código de reiniciador.

Argumentos de salida

Tabla 128. Argumentos de salida para el constructor FencedGeneric_Wrapper

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Puntero al código de retorno. Este valor debe ser 0 para que la operación se realice satisfactoriamente.

Valor de retorno

Ninguno.

Destructor ~FencedGeneric_Wrapper

Finalidad

Destructor para la clase Fenced_Generic_Wrapper.

Sintaxis

```
~FencedGeneric_Wrapper ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función get_name

Dónde se define

Reiniciador

Finalidad

Devolver un puntero a la serie de caracteres terminada por nulo que contiene el nombre de reiniciador.

Sintaxis

```
sqluint8* get_name()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la serie de caracteres.

Función get_corelib

Dónde se define
Reiniciador

Finalidad
Devolver un puntero a la serie de caracteres terminada por nulo que contiene el nombre de biblioteca de reiniciador.

Sintaxis

```
sqluint8* get_corelib()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la serie de caracteres.

Función get_version

Dónde se define
Reiniciador

Finalidad
Devolver la versión del código de reiniciador.

Sintaxis

```
sqlint32 get_version()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Versión de reiniciador.

Función get_info

Dónde se define
Reiniciador

Finalidad
Devolver un puntero al objeto de información de catálogo de reiniciador.

Sintaxis

```
Wrapper_Info* get_info()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al objeto de información de catálogo.

Función initialize_my_wrapper

Dónde se define
Reiniciador

Finalidad
Inicializar el estado de objeto de reiniciador a partir del objeto de información de catálogo. La versión por omisión no realiza ninguna acción.

Uso El reiniciador puede implementar la subclase específica de reiniciador de Fenced_Generic_Wrapper si se soportan opciones de reiniciador específicas de reiniciador.

Sintaxis

```
virtual sqlint32 initialize_my_wrapper (Wrapper_Info* a_wrapper_info)
```

Argumentos de entrada

Tabla 129. Argumentos de entrada para la función de miembro initialize_my_wrapper

Nombre	Tipo de datos	Descripción
a_wrapper_info	Wrapper_Info*	Objeto de información de catálogo.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función create_server

Dónde se define
Reiniciador

Finalidad
Crear una instancia de la subclase apropiada de Servidor para el reiniciador.

Uso El reiniciador debe alterar temporalmente esta función de miembro en la subclase específica de reiniciador de Fenced_Generic_Wrapper

Sintaxis

```
virtual Server* create_server (sqluint8* a_server_name,
                               sqlint32* a_rc)
```

Argumentos de entrada

Tabla 130. Argumentos de entrada para la función de miembro create_server

Nombre	Tipo de datos	Descripción
a_server_name	sqluint8*	Nombre terminado por nulo del servidor.

Argumentos de salida

Tabla 131. Argumentos de salida para la función de miembro create_server

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Puntero al código de retorno; 0 indica que la operación se ha realizado satisfactoriamente.

Valor de retorno
Puntero al objeto de servidor recién creado o nulo.

Funciones asociadas

La clase Fenced_Generic_Wrapper no utiliza la función FencedWrapper_Hook, pero esta función está directamente asociada con dicha clase.

Función FencedWrapper_Hook

Finalidad
Función de enganche que permite al servidor federado DB2 crear

instancias de la subclase específica de reiniciador de Fenced_Generic_Wrapper. El reiniciador debe implementar esta función y dicha función se debe exportar del módulo de reiniciador protegido.

Sintaxis

```
extern "C" FencedWraper* FencedWrapper_Hook()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Subclase de Fenced_Generic_Wrapper de la que se han creado instancias o nulo si se produce un error.

Consulta relacionada:

• "Clases de reiniciador para la API C++" en la página 70

Clases de servidor para la API C++

La tabla siguiente describe cada clase de servidor para la API C++.

Tabla 132. Clases de servidor

Nombre de clase	Descripción
Unfenced_Generic_Server	La clase Unfenced_Generic_Server es una subclase de la clase Unfenced_Server y es la clase base abstracta para toda la funcionalidad de servidor que opera en el espacio de proceso desprotegido (fiable). La clase Unfenced_Generic_Server es responsable de validar la información de catálogo de las sentencias CREATE SERVER y ALTER SERVER.
Fenced_Generic_Server	La clase Fenced_Generic_Server es una subclase de la clase Server y es la clase base abstracta para toda la funcionalidad de servidor que opera en el espacio de proceso protegido (no fiable). La clase Fenced_Generic_Server es responsable de crear conexiones remotas y apodos.

Consulta relacionada:

• "Clase Unfenced_Generic_Server" en la página 86
• "Clase Fenced_Generic_Server" en la página 97

Clase Unfenced_Generic_Server

Este tema describe la clase Unfenced_Generic_Server y proporciona detalles para el constructor y las funciones de miembro.

Visión general

La clase Unfenced_Generic_Server es una subclase de la clase Unfenced_Server y es la clase base abstracta para toda la funcionalidad de servidor que opera en el espacio de proceso desprotegido (fiable). Esta clase es responsable de validar la información de catálogo de las sentencias CREATE SERVER y ALTER SERVER.

La clase Unfenced_Generic_Server es una de las clases de servidor para la API C++.

Uso El reiniciador debe implementar una subclase de Unfenced_Generic_Server. El reiniciador crea una instancia de la clase Unfenced_Generic_Server en el método create_server() de la subclase específica de reiniciador de la clase Unfenced_Generic_Wrapper.

Archivo

sqlqg_unfenced_generic_server.h

Miembros de datos

La tabla siguiente lista los miembros de datos que puede utilizar con la clase Unfenced_Generic_Server.

Tabla 133. Miembros de datos para la clase Unfenced_Generic_Server

Nombre	Tipo de datos	Descripción
info	Server_Info*	Información de catálogo para el servidor.
name	sqluint8*	Serie de caracteres terminada por nulo que contiene el nombre de servidor.
kind	server_kind	Clase de servidor que debe ser Server::generic_kind.
wrapper	Wrapper*	Puntero al objeto de reiniciador asociado que es propietario de este servidor.

Constructores y funciones de miembro

Las tablas siguientes describen el constructor y las funciones de miembro de la clase Unfenced_Generic_Server. El constructor y las funciones se describen más detalladamente después de las tablas.

Tabla 134. Constructores para la clase Unfenced_Generic_Server

Constructor	Descripción
Unfenced_Generic_server	Construir una instancia de la clase Unfenced_Generic_Server.

Tabla 135. Funciones de miembro para la clase Unfenced_Generic_Server

Función de miembro	Descripción
create_remote_user	Crear una instancia de una subclase apropiada de Remote_User.
plan_request	Analizar un plan propuesto y determinar qué parte, si hay alguna, que se puede incorporar a la fuente de datos.
get_selectivity	Calcular la selectividad de una lista de predicados.
create_reply	Crear una instancia de un objeto de respuesta.
get_name	Devolver un puntero al nombre de servidor terminado por nulo.
get_type	Recuperar de la información de catálogo el tipo de servidor terminado por nulo tal como se ha especificado en la sentencia CREATE SERVER.
get_version	Recuperar la versión de servidor terminada por nulo como se ha especificado en la sentencia CREATE SERVER.
get_info	Devolver un puntero al objeto de información de catálogo almacenada.
initialize_my_server	Inicializar las instancias de servidor de la información de catálogo válida.
create_nickname	Crear una instancia de una subclase apropiada de un apodo para este servidor.
is_reserved_server_option	Identificar una opción que está soportada por DB2 pero ignorada por el reiniciador.
verify_my_register_server_info	Validar información que se proporciona en la sentencia CREATE SERVER.
verify_my_alter_server_info	Validar información que se proporciona en la sentencia ALTER SERVER.

Constructor Unfenced_Generic_server

Finalidad

Construir una instancia de la clase Unfenced_Generic_Server.

Sintaxis

```
Unfenced_Generic_server (sqluint8*        a_server_name,
                         UnfencedWrapper* a_wrapper,
                         sqlint32*        a_rc)
```

Argumentos de entrada

Tabla 136. Argumentos de entrada para el constructor Unfenced_Generic_server

Nombre	Tipo de datos	Descripción
a_server_name	sqluint8*	Serie terminada por nulo que contiene el nombre del servidor.
a_wrapper	UnfencedWrapper*	Puntero al objeto de reiniciador que es propietario de este objeto de servidor.

Argumentos de salida

Tabla 137. Argumentos de salida para el constructor Unfenced_Generic_server

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Puntero al código de retorno; 0 indica que la operación se ha realizado satisfactoriamente.

Valor de retorno

Ninguno.

Función create_remote_user

Finalidad

Crear una instancia de una subclase apropiada de Remote_User.

Uso El reiniciador puede implementar esta función de miembro en la subclase de servidor desprotegida específica de reiniciador. Se debe implementar esta función de miembro si se implementa una subclase específica de reiniciador de la clase Unfenced_Generic_User.

Sintaxis

```
virtual Remote_User* create_remote_user (sqluint8* a_user_name,
                                         sqlint32* a_rc)
```

Argumentos de entrada

Tabla 138. Argumentos de entrada para la función de miembro create_remote_user

Nombre	Tipo de datos	Descripción
a_user_name	sqluint8*	Serie terminada por nulo que contiene el nombre del usuario.

Argumentos de salida

Tabla 139. Argumentos de salida para la función de miembro create_remote_user

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Puntero al código de retorno; 0 indica que la operación se ha realizado satisfactoriamente.

Valor de retorno

Puntero al objeto Remote_User.

Función plan_request

Finalidad

Analizar un plan propuesto y determinar qué parte, si hay alguna, que se puede incorporar a la fuente de datos.

Uso El reiniciador debe implementar esta función de miembro en la subclase de servidor desprotegida específica de reiniciador.

Sintaxis

```
virtual sqlint32 plan_request (Request* a_req,
                               Reply**  a_rpl) = 0
```

Argumentos de entrada

Tabla 140. Argumentos de entrada para la función de miembro plan_request

Nombre	Tipo de datos	Descripción
a_req	Request*	Puntero al objeto de petición que describe el plan propuesto.

Argumentos de salida

Tabla 141. Argumentos de salida para la función de miembro plan_request

Nombre	Tipo de datos	Descripción
a_rpl	Reply**	Puntero a la cadena de objetos de respuesta construidos por plan_request.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_selectivity

Finalidad

Calcular la selectividad de una lista de predicados.

Uso El reiniciador puede implementar esta función de miembro en la subclase de servidor desprotegida específica de reiniciador. La versión por omisión utiliza fórmulas de DB2 Information Integrator incorporadas para calcular la selectividad. El reiniciador utiliza estas fórmulas y las dependencias conocidas entre expresiones de predicado.

Sintaxis

```
virtual sqlint32 get_selectivity (Predicate_List* a_pl,
                                  float*          a_selectivity)
```

Argumentos de entrada

Tabla 142. Argumentos de entrada para la función de miembro get_selectivity

Nombre	Tipo de datos	Descripción
a_pl	Predicate_List*	Lista de predicados.

Argumentos de salida

Tabla 143. Argumentos de salida para la función de miembro get_selectivity

Nombre	Tipo de datos	Descripción
a_selectivity	float*	Selectividad que se expresa como un valor de 0,0 a 1,0.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función create_reply

Finalidad

Crear una instancia de un objeto de respuesta.

Uso El reiniciador puede implementar esta función de miembro en la subclase de servidor desprotegida específica de reiniciador y se debe implementar esta función de miembro si se utiliza una subclase Reply específica de reiniciador. Este método se altera temporalmente si el reiniciador implementa su propio modelo de coste creando una subclase de la respuesta.

Sintaxis

```
virtual sqlint32 create_reply (Request* a_req,
                               Reply**  a_rpl)
```

Argumentos de entrada

Tabla 144. Argumentos de entrada para la función de miembro create_reply

Nombre	Tipo de datos	Descripción
a_req	Request*	Puntero a un objeto de petición.

Argumentos de salida

Tabla 145. Argumentos de salida para la función de miembro create_reply

Nombre	Tipo de datos	Descripción
a_rpl	Reply**	Puntero al objeto de respuesta.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_name

Dónde se define

Servidor

Finalidad

Devolver un puntero al nombre de servidor terminado por nulo.

Sintaxis

```
sqluint8* get_name()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al nombre de servidor.

Función get_type

Dónde se define
Servidor

Finalidad
Recuperar de la información de catálogo el tipo de servidor terminado por nulo tal como se ha especificado en la sentencia CREATE SERVER.

Uso Este método sólo es válido *después* de que se haya inicializado el objeto de servidor. Si no se especifica el tipo de servidor en el catálogo, este método anota cronológicamente un error. Sin embargo, no se genera ningún error de usuario.

Sintaxis

```
sqluint8* get_type()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al tipo de servidor.

Función get_version

Dónde se define
Servidor

Finalidad
Recuperar la versión de servidor terminada por nulo como se ha especificado en la sentencia CREATE SERVER.

Uso Este método sólo es válido después de que se haya inicializado el objeto de servidor. Si no se especifica el tipo de servidor en el catálogo, este método anota cronológicamente un error. Sin embargo, no se genera ningún error de usuario.

Sintaxis

```
sqluint8* get_version()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la versión de servidor.

Función get_info

Dónde se define
Servidor

Finalidad
Devolver un puntero al objeto de información de catálogo almacenada.

Uso Este método sólo es válido después de la inicialización del servidor.

Sintaxis

```
Server_Info* get_info()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al objeto de información de catálogo.

Función initialize_my_server

Dónde se define
Servidor

Finalidad
Inicializar las instancias de servidor de la información de catálogo válida.

Uso El reiniciador puede implementar esta función de miembro en la subclase de servidor desprotegida específica de reiniciador y esta función de miembro debe implementarse si se soportan opciones de servidor específicas de reiniciador.

Sintaxis

```
virtual sqlint32 initialize_my_server (Server_Info* a_server_info)
```

Argumentos de entrada

Tabla 146. Argumentos de entrada para la función de miembro initialize_my_server

Nombre	Tipo de datos	Descripción
a_server_info	Server_Info*	Información de catálogo validada.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función create_nickname

Dónde se define
Servidor

Uso El reiniciador debe implementar esta función de miembro en la subclase de servidor desprotegida específica de reiniciador.

Finalidad
Crear una instancia de una subclase apropiada de un apodo para este servidor.

Sintaxis

```
virtual Nickname* create_nickname (sqluint8* a_schema_name,
                                   sqluint8* a_nickname_name,
                                   sqlint32* a_rc)
```

Argumentos de entrada

Tabla 147. Argumentos de entrada para la función de miembro create_nickname

Nombre	Tipo de datos	Descripción
a_schema_name	sqluint8*	Serie terminada por nulo que contiene el nombre de esquema local (DB2) del apodo.
a_nickname_name	sqluint8*	Serie terminada por nulo que contiene el nombre de apodo local (DB2).

Argumentos de salida

Tabla 148. Argumentos de salida para la función de miembro create_nickname

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Puntero al código de retorno; un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Valor de retorno

Puntero al objeto de apodo del que se ha creado una instancia.

Función is_reserved_server_option

Dónde se define

UnfencedServer

Finalidad

Identificar una opción que está soportada por DB2 pero ignorada por el reiniciador.

Sintaxis

```
virtual sqlint32 is_reserved_server_option (sqluint8* a_op_name)
```

Argumentos de entrada

Tabla 149. Argumentos de entrada para la función de miembro is_reserved_server_option

Nombre	Tipo de datos	Descripción
a_op_name	sqluint8*	Serie terminada por nulo que contiene un nombre de opción.

Argumentos de salida

Ninguno.

Valor de retorno

Un valor de 0 indica que la opción no es una opción de servidor definida por DB2. Un valor distinto de cero indica que la opción es una opción de servidor definida por DB2.

Función verify_my_register_server_info

Dónde se define
UnfencedServer

Finalidad
Validar información que se proporciona en la sentencia CREATE SERVER.

Uso El reiniciador puede implementar esta función de miembro en la subclase de servidor desprotegida específica de reiniciador y esta función de miembro debe implementarse si se soportan opciones de servidor específicas de reiniciador. El reiniciador debe comprobar si ya se ha asignado un objeto a_delta_info antes de asignar el suyo propio.

Sintaxis

```
virtual sqlint32 verify_my_register_server_info
        (Server_Info*  a_server_info,
         Server_Info** a_delta_info)
```

Argumentos de entrada

Tabla 150. Argumentos de entrada para la función de miembro verify_my_register_server_info

Nombre	Tipo de datos	Descripción
a_server_info	Server_Info*	Información de la sentencia CREATE SERVER.

Argumentos de salida

Tabla 151. Argumentos de salida para la función de miembro verify_my_register_server_info

Nombre	Tipo de datos	Descripción
a_delta_info	Server_Info**	Información adicional proporcionada por el reiniciador.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función verify_my_alter_server_info

Dónde se define
UnfencedServer

Finalidad

Validar información que se proporciona en la sentencia ALTER SERVER.

Uso El reiniciador puede implementar esta función de miembro en la subclase de servidor desprotegida específica de reiniciador y esta función de miembro debe implementarse si se soportan opciones de servidor específicas de reiniciador. El reiniciador debe comprobar si ya se ha asignado un objeto a_delta_info antes de asignar el suyo propio.

Sintaxis

```
virtual sqlint32 verify_my_alter_server_info
        (Server_Info*  a_server_info,
         Server_Info** a_delta_info)
```

Argumentos de entrada

Tabla 152. Argumentos de entrada para la función de miembro verify_my_alter_server_info

Nombre	Tipo de datos	Descripción
a_server_info	Server_Info*	Información de una sentencia ALTER SERVER.

Argumentos de salida

Tabla 153. Argumentos de salida para la función de miembro verify_my_alter_server_info

Nombre	Tipo de datos	Descripción
a_delta_info	Server_Info**	Información adicional proporcionada por el reiniciador.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Consulta relacionada:

- "Clases de servidor para la API C++" en la página 85

Clase Fenced_Generic_Server

Este tema describe la clase Fenced_Generic_Server y proporciona detalles para el constructor y las funciones de miembro.

Visión general

La clase Fenced_Generic_Server es una subclase de la clase Server y es la clase base abstracta para toda la funcionalidad de servidor que opera en el espacio de proceso protegido (no fiable). Esta clase es responsable de crear conexiones remotas y apodos.

La clase Fenced_Generic_Server es una de las clases de servidor para la API C++.

Uso El reiniciador debe implementar una subclase de Fenced_Generic_Server. El reiniciador crea una instancia de esta clase en el método create_server() de la subclase específica de reiniciador de Fenced_Generic_Wrapper.

Archivo

sqlqg_fenced_genserver.h

Miembros de datos

La tabla siguiente lista los miembros de datos que puede utilizar con la clase Fenced_Generic_Server.

Tabla 154. Miembros de datos para la clase Fenced_Generic_Server

Nombre	Tipo de datos	Descripción
info	Server_Info*	Información de catálogo para el servidor.
name	sqluint8*	Serie de caracteres terminada por nulo que contiene el nombre de servidor.
kind	server_kind	Clase de servidor que debe ser Server::generic_kind.
wrapper	Wrapper*	Puntero al objeto de reiniciador asociado que es propietario de este servidor.

Constructores y funciones de miembro

Las tablas siguientes describen el constructor y las funciones de miembro de la clase Fenced_Generic_Server. El constructor y las funciones se describen más detalladamente después de las tablas.

Tabla 155. Constructores para la clase Fenced_Generic_Server

Constructor	Descripción
Fenced_Generic_Server	Constructor para esta clase.

Tabla 156. Funciones de miembro para la clase Fenced_Generic_Server

Función de miembro	Descripción
create_remote_connection	Crear una instancia de una subclase apropiada de Remote_Connection.
get_name	Devolver un puntero al nombre de servidor terminado por nulo.
get_type	Recuperar de la información de catálogo el tipo de servidor terminado por nulo que se especifica en la sentencia CREATE SERVER.
get_version	Recuperar la versión de servidor terminada por nulo que se especifica en la sentencia CREATE SERVER.
get_info	Devolver un puntero al objeto de información de catálogo almacenada.
initialize_my_server	Inicializar las instancias de servidor de la información de catálogo válida.
create_remote_user	Crear una instancia de una subclase apropiada de Remote_User para este reiniciador.
create_nickname	Crear una instancia de una subclase apropiada de Nickname para este servidor.

Constructor Fenced_Generic_Server

Finalidad

Constructor para esta clase.

Sintaxis

```
Fenced_Generic_Server (sqluint8* a_server_name,
                       FencedWrapper* wrapper,
                       sqlint32* rc)
```

Argumentos de entrada

Tabla 157. Argumentos de entrada para el constructor Fenced_Generic_Server

Nombre	Tipo de datos	Descripción
a_server_name	sqluint8*	Nombre del servidor.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función create_remote_connection

Dónde se define
FencedServer

Finalidad
Crear una instancia de una subclase apropiada de Remote_Connection.

Uso El reiniciador debe implementar esta función de miembro en la subclase de servidor protegida específica de reiniciador.

Sintaxis

```
virtual sqlint32 create_remote_connection
                 (FencedRemote_User*  a_user,
                  Remote_connection** a_connection)
```

Argumentos de entrada

Tabla 158. Argumentos de entrada para la función de miembro create_remote_connection

Nombre	Tipo de datos	Descripción
a_user	FencedRemote_User*	Puntero al objeto que representa el usuario que se utiliza para la conexión.

Argumentos de salida

Tabla 159. Argumentos de salida para la función de miembro create_remote_connection

Nombre	Tipo de datos	Descripción
a_connection	Remote_Connection**	Puntero al objeto de conexión remota.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_name

Dónde se define
Servidor

Finalidad
Devolver un puntero al nombre de servidor terminado por nulo.

Sintaxis

```
sqluint8* get_name()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al nombre de servidor.

Función get_type

Dónde se define
Servidor

Finalidad
Recuperar de la información de catálogo el tipo de servidor terminado por nulo que se especifica en la sentencia CREATE SERVER.

Uso Este método sólo es válido después de que se haya inicializado el objeto de servidor (es decir, cuando se ejecuta initialize_server). Si no se especifica el tipo de servidor en el catálogo, al llamar este método se anota cronológicamente un error. Sin embargo, no se genera ningún error de usuario.

Sintaxis

```
sqluint8* get_type()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al tipo de servidor.

Función get_version

Dónde se define
Servidor

Finalidad
Recuperar la versión de servidor terminada por nulo que se especifica en la sentencia CREATE SERVER.

Uso Este método sólo es válido después de que se haya inicializado el objeto de servidor (es decir, cuando se ejecuta initialize_server). Si no

se especifica el tipo de servidor en el catálogo, al llamar este método se anota cronológicamente un error. Sin embargo, no se genera ningún error de usuario.

Sintaxis

```
sqluint8* get_version()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la versión de servidor.

Función get_info

Dónde se define
Servidor

Finalidad
Devolver un puntero al objeto de información de catálogo almacenada.

Uso Este método sólo es válido después de la inicialización del servidor.

Sintaxis

```
Server_Info* get_info()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al objeto de información de catálogo.

Función initialize_my_server

Dónde se define
Servidor

Finalidad
Inicializar las instancias de servidor de la información de catálogo válida.

Uso El reiniciador puede implementar esta función de miembro en la

subclase de servidor protegida específica de reiniciador. Se debe implementar esta función de miembro si se soportan opciones de servidor específicas de reiniciador.

Sintaxis

```
virtual sqlint32 initialize_my_server (Server_Info* a_server_info)
```

Argumentos de entrada

Tabla 160. Argumentos de entrada para la función de miembro initialize_my_server

Nombre	Tipo de datos	Descripción
a_server_info	Server_Info*	Información de catálogo validada.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función create_remote_user

Dónde se define

Servidor

Finalidad

Crear una instancia de una subclase apropiada de Remote_User para este reiniciador.

Uso El reiniciador puede implementar esta función de miembro en la subclase de servidor protegida específica de reiniciador. Se debe implementar esta función de miembro si se utiliza una subclase Fenced_Generic_User específica de reiniciador.

Sintaxis

```
virtual Remote_User* create_remote_user (sqluint8* a_user_name,
                                         sqlint32* a_rc)
```

Argumentos de entrada

Tabla 161. Argumentos de entrada para la función de miembro create_remote_user

Nombre	Tipo de datos	Descripción
a_user_name	sqluint8*	Nombre del usuario remoto.

Argumentos de salida

Tabla 162. Argumentos de salida para la función de miembro create_remote_user

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Puntero al código de retorno; 0 indica que la operación se ha realizado satisfactoriamente.

Valor de retorno

Puntero al Remote_User del que se ha creado una instancia.

Función create_nickname

Dónde se define

Servidor

Finalidad

Crear una instancia de una subclase apropiada de Nickname para este servidor.

Uso

El reiniciador debe implementar esta función de miembro en la subclase de servidor protegida específica de reiniciador.

Sintaxis

```
virtual Nickname* create_nickname (sqluint8* a_schema_name,
                                   sqluint8* a_nickname_name,
                                   sqlint32* a_rc)
```

Argumentos de entrada

Tabla 163. Argumentos de entrada para la función de miembro create_nickname

Nombre	Tipo de datos	Descripción
a_schema_name	sqluint8*	Serie terminada por nulo que contiene el nombre de esquema local (DB2) del apodo.
a_nickname_name	sqluint8*	Serie terminada por nulo que contiene el nombre de apodo local (DB2).

Argumentos de salida

Tabla 164. Argumentos de salida para la función de miembro create_nickname

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Puntero al código de retorno; un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Valor de retorno

Puntero al objeto Nickname del que se ha creado una instancia.

Consulta relacionada:

- "Clases de servidor para la API C++" en la página 85

Clases de usuario para la API C++

La tabla siguiente describe cada clase de usuario para la API C++.

Tabla 165. Clases de usuario

Nombre de clase	Descripción
Unfenced_Generic_User	Esta clase representa una correlación de usuario en el espacio de proceso desprotegido (fiable). Esta clase es responsable de validar la información de las sentencias CREATE USER MAPPING y ALTER USER MAPPING.
Fenced_Generic_User	Esta clase representa una correlación de usuario en el espacio de proceso protegido (no fiable).

Consulta relacionada:

- "Clase Unfenced_Generic_User" en la página 105
- "Clase Fenced_Generic_User" en la página 111

Clase Unfenced_Generic_User

Este tema describe la clase Unfenced_Generic_User y proporciona detalles para el constructor, el destructor y las funciones de miembro.

Visión general

La clase Unfenced_Generic_User representa una correlación de usuario en el espacio de proceso desprotegido (fiable). Esta clase es responsable de validar la información de las sentencias CREATE USER MAPPING y ALTER USER MAPPING.

La clase Unfenced_Generic_User es una de las clases de usuario para la API C++.

Uso el reiniciador debe implementar una subclase de Unfenced_Generic_User si se utilizan opciones específicas de reiniciador para la sentencia CREATE USER MAPPING o ALTER USER MAPPING. Esta clase sólo se utiliza para la validación de opciones y el reiniciador crea una instancia de la misma en el método create_remote_user() de la subclase específica de reiniciador de Unfenced_Generic_Server.

Archivo

sqlqg_unfenced_generic_user.h

Miembros de datos

La tabla siguiente lista los miembros de datos que puede utilizar con la clase Unfenced_Generic_User.

Tabla 166. Miembros de datos para la clase Unfenced_Generic_User

Nombre	Tipo de datos	Descripción
local_name	sqluint8*	Nombre de usuario (local) terminado por nulo.
server	Server*	Puntero al objeto de Servidor que es propietario de este usuario.
info	User_Info*	Copia local de la información de catálogo. Este miembro de datos sólo es válido después de initialize_my_user.

Constructores, destructores y funciones de miembro

Las tablas siguientes describen el constructor, el destructor y las funciones de miembro de la clase Unfenced_Generic_User. El constructor, el destructor y las funciones se describen más detalladamente después de las tablas.

Tabla 167. Constructores para la clase Unfenced_Generic_User

Constructor	Descripción
Unfenced_Generic_User	Construir una instancia de Unfenced_Generic_User.

Tabla 168. Destructores para la clase Unfenced_Generic_User

Destructor	Descripción
~Unfenced_Generic_User	Destruir una instancia de Unfenced_Generic_User.

Tabla 169. Funciones de miembro para la clase Unfenced_Generic_User

Función de miembro	Descripción
get_info	Devolver un puntero a la copia local de User_Info.
is_reserved_user_option	Verificar que una opción de catálogo especificada es una opción reservada de DB2 para una correlación de usuario.
initialize_my_user	Inicializar el estado del objeto a partir de la información de catálogo válida.
verify_my_register_user_info	Validar información de la sentencia CREATE USER MAPPING.
verify_my_alter_user_info	Validar información de la sentencia ALTER USER MAPPING.

Constructor Unfenced_Generic_User

Finalidad

Construir una instancia de Unfenced_Generic_User.

Sintaxis

```
Unfenced_Generic_User (sqluint8*       a_user_name,
                       UnfencedServer* a_server,
                       sqlint32*       a_rc)
```

Argumentos de entrada

Tabla 170. Argumentos de entrada para el constructor Unfenced_Generic_User

Nombre	Tipo de datos	Descripción
a_user_name	sqluint8*	Nombre de usuario (local) terminado por nulo.
a_server	UnfencedServer*	Puntero al objeto de Servidor que es propietario de este usuario.

Argumentos de salida

Tabla 171. Argumentos de salida para el constructor Unfenced_Generic_User

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Puntero al objeto de Servidor que es propietario de este usuario.

Valor de retorno

Ninguno.

Destructor ~Unfenced_Generic_User

Finalidad

Destruir una instancia de Unfenced_Generic_User.

Sintaxis

```
virtual ~Unfenced_Generic_User ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función get_info

Finalidad

Devolver un puntero a la copia local de User_Info.

Sintaxis

```
User_Info* get_info ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Puntero al objeto User_Info.

Función is_reserved_user_option

Finalidad

Verificar que una opción de catálogo especificada es una opción reservada de DB2 para una correlación de usuario.

Sintaxis

```
virtual sqluint32 is_reserved_user_option (sqluint8* a_opt_name)
```

Argumentos de entrada

Tabla 172. Argumentos de entrada para la función de miembro is_reserved_user_option

Nombre	Tipo de datos	Descripción
a_opt_name	sqluint8*	Nombre de opción terminado por nulo que se debe comprobar.

Argumentos de salida

Ninguno.

Valor de retorno

Un valor de 0 indica que la opción no es una opción reservada de DB2.

Función initialize_my_user

Finalidad

Inicializar el estado del objeto a partir de la información de catálogo válida.

Uso El reiniciador puede implementar esta función de miembro en la subclase de usuario desprotegida específica de reiniciador.

Sintaxis

```
virtual sqlint32 initialize_my_user (User_Info* a_user_info)
```

Argumentos de entrada

Tabla 173. Argumentos de entrada para la función de miembro initialize_my_user

Nombre	Tipo de datos	Descripción
a_user_info	User_Info*	Información de catálogo validada.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función verify_my_register_user_info

Finalidad

Validar información de la sentencia CREATE USER MAPPING.

Uso El reiniciador puede implementar esta función de miembro en la subclase de usuario desprotegida específica de reiniciador. Se debe implementar esta función de miembro si se soportan opciones de correlación de usuario específicas de reiniciador.

El reiniciador comprueba si se ha asignado un objeto a_delta_info antes de asignar uno él mismo.

Sintaxis

```
virtual sqlint32 verify_my_register_user_info (User_Info*  a_user_info,
                                               User_Info** a_delta_info)
```

Argumentos de entrada

Tabla 174. Argumentos de entrada para la función de miembro verify_my_register_user_info

Nombre	Tipo de datos	Descripción
a_user_info	User_Info*	Información de una sentencia CREATE USER MAPPING.

Argumentos de salida

Tabla 175. Argumentos de salida para la función de miembro verify_my_register_user_info

Nombre	Tipo de datos	Descripción
a_delta_info	User_Info**	Información adicional proporcionada por el reiniciador.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función verify_my_alter_user_info

Finalidad

Validar información de la sentencia ALTER USER MAPPING.

Uso El reiniciador puede implementar esta función de miembro en la subclase de usuario desprotegida específica de reiniciador. Se debe implementar esta función de miembro si se soportan opciones de correlación de usuario específicas de reiniciador.

El reiniciador comprueba si se ha asignado un objeto a_delta_info antes de asignar uno él mismo.

Sintaxis

```
virtual sqlint32 verify_my_alter_user_info (User_Info*  a_user_info,
                                            User_Info** a_delta_info)
```

Argumentos de entrada

Tabla 176. Argumentos de entrada para la función de miembro verify_my_alter_user_info

Nombre	Tipo de datos	Descripción
a_user_info	User_Info*	Información de la sentencia ALTER USER MAPPING.

Argumentos de salida

Tabla 177. Argumentos de salida para la función de miembro verify_my_alter_user_info

Nombre	Tipo de datos	Descripción
a_delta_info	User_Info**	Información adicional proporcionada por el reiniciador.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Consulta relacionada:

- "Clases de usuario para la API C++" en la página 105

Clase Fenced_Generic_User

Este tema describe la clase Fenced_Generic_User y proporciona detalles para el constructor, el destructor y las funciones de miembro.

Visión general

La clase Fenced_Generic_User representa una correlación de usuario en el espacio de proceso protegido (no fiable).

La clase Fenced_Generic_User es una de las clases de usuario para la API C++.

Uso El reiniciador crea una instancia de esta clase en el método create_remote_user() de la subclase específica de reiniciador de Fenced_Generic_Server. El reiniciador implementa una subclase de Fenced_Generic_User si se soportan opciones de correlación de usuario específicas de reiniciador.

Archivo

sqlqg_fenced_generic_user.h

Miembros de datos

La tabla siguiente lista los miembros de datos que puede utilizar con la clase Fenced_Generic_User.

Tabla 178. Miembros de datos para la clase Fenced_Generic_User

Nombre	Tipo de datos	Descripción
local_name	sqluint8*	Nombre de usuario (local) terminado por nulo.
server	Server*	Puntero al objeto de Servidor que es propietario de este usuario.
info	User_Info*	Copia local de la información de catálogo. Este miembro de datos sólo es válido después de initialize_my_user.

Constructores, destructores y funciones de miembro

Las tablas siguientes describen el constructor, el destructor y las funciones de miembro de la clase Fenced_Generic_User. El constructor, el destructor y las funciones se describen más detalladamente después de las tablas.

Tabla 179. Constructores para la clase Fenced_Generic_User

Constructor	Descripción
Fenced_Generic_User	Construir una instancia de Fenced_Generic_User.

Tabla 180. Destructores para la clase Fenced_Generic_User

Destructor	Descripción
~Fenced_Generic_User	Destruir una instancia de Fenced_Generic_User.

Tabla 181. Funciones de miembro para la clase Fenced_Generic_User

Función de miembro	Descripción
get_info	Devolver un puntero a la copia local de User_Info.
initialize_my_user	Inicializar el estado del objeto a partir de la información de catálogo válida.

Constructor Fenced_Generic_User

Finalidad

Construir una instancia de Fenced_Generic_User.

Sintaxis

```
Fenced_Generic_User (sqluint8*     a_local_user_name,
                     FencedServer* a_server,
                     sqlint32*     a_rc)
```

Argumentos de entrada

Tabla 182. Argumentos de entrada para el constructor Fenced_Generic_User

Nombre	Tipo de datos	Descripción
a_local_user_name	sqluint8*	Nombre de usuario (local) terminado por nulo.
a_server	FencedServer*	Puntero al objeto de Servidor que es propietario de este usuario.

Argumentos de salida

Tabla 183. Argumentos de salida para el constructor Fenced_Generic_User

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Puntero al código de retorno; 0 indica que la operación se ha realizado satisfactoriamente.

Valor de retorno

Ninguno.

Destructor ~Fenced_Generic_User

Finalidad
Destruir una instancia de Fenced_Generic_User.

Sintaxis

```
virtual ~Fenced_Generic_User ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Ninguno.

Función get_info

Finalidad
Devolver un puntero a la copia local de User_Info.

Sintaxis

```
User_Info* get_info ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al objeto User_Info.

Función initialize_my_user

Finalidad
Inicializar el estado del objeto a partir de la información de catálogo válida.

Uso El reiniciador puede implementar esta función de miembro en la subclase de usuario protegida específica de reiniciador y se debe implementar dicha clase si se soportan opciones de correlación de usuario específicas de reiniciador.

Sintaxis

```
virtual sqlint32 initialize_my_user (User_Info* a_user_info)
```

Argumentos de entrada

Tabla 184. Argumentos de entrada para la función de miembro initialize_my_user

Nombre	Tipo de datos	Descripción
a_user_info	User_Info*	Información de catálogo validada.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Consulta relacionada:

• "Clases de usuario para la API C++" en la página 105

Clases de apodo para la API C++

La tabla siguiente describe cada clase de apodo para la API C++.

Tabla 185. Clases de apodo

Nombre de clase	Descripción
Unfenced_Generic_Nickname	Esta clase representa un apodo en el espacio de proceso desprotegido (fiable). Esta clase es responsable de validar la información de las sentencias CREATE NICKNAME y ALTER NICKNAME.
Fenced_Generic_Nickname	Esta clase representa un apodo en el espacio de proceso protegido (no fiable). Esta clase es responsable de validar la información de la sentencia CREATE NICKNAME.

Consulta relacionada:

• "Clase Unfenced_Generic_Nickname" en la página 115
• "Clase Fenced_Generic_Nickname" en la página 125

Clase Unfenced_Generic_Nickname

Este tema describe la clase Unfenced_Generic_Nickname y proporciona detalles para el constructor y las funciones de miembro.

Visión general

La clase Unfenced_Generic_Nickname representa un apodo en el espacio de proceso desprotegido (fiable). Esta clase es responsable de validar la información de las sentencias CREATE NICKNAME y ALTER NICKNAME.

La clase Unfenced_Generic_Nickname es una de las clases de apodo para la API C++.

Uso El reiniciador debe crear una subclase de esta clase. El reiniciador crea una instancia de esta clase en el método create_nickname() de la subclase específica de reiniciador de Unfenced_Generic_Server.

Archivo

sqlqg_unfenced_generic_nickhame.h

Miembros de datos

La tabla siguiente lista los miembros de datos que puede utilizar con la clase Unfenced_Generic_Nickname.

Tabla 186. Miembros de datos para la clase Unfenced_Generic_Nickname

Nombre	Tipo de datos	Descripción
name	sqluint8*	Serie terminada por nulo que contiene el nombre (local) del apodo.
schema	sqluint8*	Serie terminada por nulo que contiene el nombre (local) del esquema.
server	Server*	Puntero al objeto de servidor que es propietario de este apodo.
m_cardinality	sqlint64	Cardinalidad.
m_advance_cost	sqlint32	Coste (tiempo en milisegundos) para buscar y cargar una fila.
m_setup_cost	sqlint32	Coste (tiempo en milisegundos) para realizar una sola configuración de una consulta.
m_submission_cost	sqlint32	Coste (tiempo en milisegundos) para someter una consulta (distinto de los costes de una sola configuración).

Constructores y funciones de miembro

Las tablas siguientes describen el constructor y las funciones de miembro de la clase Unfenced_Generic_Nickname. El constructor y las funciones se describen más detalladamente después de las tablas.

Tabla 187. Constructores para la clase Unfenced_Generic_Nickname

Constructor	Descripción
Unfenced_Generic_Nickname	Construir una instancia de Unfenced_Generic_Nickname.

Tabla 188. Funciones de miembro para la clase Unfenced_Generic_Nickname

Función de miembro	Descripción
get_local_name	Devolver puntero a un nombre de apodo local terminado por nulo.
get_local_schema	Devolver puntero a un nombre de esquema local terminado por nulo.
get_server	Devolver puntero a un objeto de servidor que la contiene.
is_reserved_nickname_option	Determinar si un nombre de opción concreto es una de las opciones de apodo incorporadas de DB2.
is_reserved_column_option	Determinar si un nombre de opción concreto es una de las opciones de columna incorporadas de DB2.
initialize_my_nickname	Inicializar el apodo con información válida del catálogo.
verify_my_register_nickname_info	Validar información de la sentencia CREATE NICKNAME.
verify_my_alter_nickname_info	Validar información de la sentencia ALTER NICKNAME.
get_card	Recuperar la cardinalidad para un apodo.
get_advance_cost	Recuperar el valor de la opción de apodo ADVANCE_COST que se utiliza en el modelo de coste por omisión.
get_setup_cost	Recuperar el valor de la opción de apodo SETUP_COST que se utiliza en el modelo de coste por omisión.
get_submission_cost	Recuperar el valor de la opción de apodo SUBMISSION_COST que se utiliza en el modelo de coste por omisión.

Constructor Unfenced_Generic_Nickname

Finalidad

Construir una instancia de Unfenced_Generic_Nickname.

Sintaxis

```
Unfenced_Generic_Nickname (sqluint8* a_schema,
           sqluint8*                a_nickname_name,
           Unfenced_Generic_Server* a_nickname_server,
           sqlint32*                a_rc)
```

Argumentos de entrada

Tabla 189. Argumentos de entrada para el constructor Unfenced_Generic_Nickname

Nombre	Tipo de datos	Descripción
a_schema	sqluint8*	Nombre de esquema terminado por nulo.
a_nickname_name	sqluint8*	Nombre de apodo (local) terminado por nulo.
a_nickname_server	Unfenced_Generic_Server*	Puntero a un objeto de servidor que lo contiene.

Argumentos de salida

Tabla 190. Argumentos de salida para el constructor Unfenced_Generic_Nickname

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Puntero al código de retorno; 0 indica que la operación se ha realizado satisfactoriamente.

Valor de retorno

Ninguno.

Función get_local_name

Finalidad

Devolver puntero a un nombre de apodo local terminado por nulo.

Sintaxis

```
sqluint8* get_local_name ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno
Nombre de apodo terminado por nulo.

Función get_local_schema

Finalidad
Devolver puntero a un nombre de esquema local terminado por nulo.

Sintaxis

```
sqluint8* get_local_schema ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Nombre de esquema terminado por nulo.

Función get_server

Finalidad
Devolver puntero a un objeto de servidor que la contiene.

Sintaxis

```
Server* get_server ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a un objeto de servidor que la contiene.

Función is_reserved_nickname_option

Finalidad
Determinar si un nombre de opción concreto es una de las opciones de apodo incorporadas de DB2.

Sintaxis

```
virtual sqluint32 is_reserved_nickname_option (sqluint8* a_opt_name)
```

Argumentos de entrada

Tabla 191. Argumentos de entrada para la función de miembro is_reserved_nickname_option

Nombre	Tipo de datos	Descripción
a_opt_name	sqluint8*	Serie de nombre de opción terminada por nulo.

Argumentos de salida

Ninguno.

Valor de retorno

Un valor de 0 indica que la opción no es una opción de apodo reservada. Un valor de retorno distinto de cero indica que la opción es una opción de apodo reservada.

Función is_reserved_column_option

Finalidad

Determinar si un nombre de opción concreto es una de las opciones de columna incorporadas de DB2.

Sintaxis

```
virtual sqluint32 is_reserved_column_option (sqluint8* a_opt_name)
```

Argumentos de entrada

Tabla 192. Argumentos de entrada para la función de miembro is_reserved_column_option

Nombre	Tipo de datos	Descripción
a_opt_name	sqluint8*	Serie de nombre de opción terminada por nulo.

Argumentos de salida

Ninguno.

Valor de retorno

Un valor de 0 indica que la opción no es una opción reservada. Un valor de retorno distinto de cero indica que la opción es una opción de columna reservada.

Función initialize_my_nickname

Finalidad

Inicializar el apodo con información válida del catálogo.

Uso El reiniciador puede implementar esta función de miembro en la subclase de apodo desprotegida específica de reiniciador.

Sintaxis

```
virtual sqlint32 initialize_my_nickname (Nickname_Info*  a_cat_info)
```

Argumentos de entrada

Tabla 193. Argumentos de entrada para la función de miembro initialize_my_nickname

Nombre	Tipo de datos	Descripción
a_cat_info	Nickname_Info*	Información del catálogo.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función verify_my_register_nickname_info

Finalidad
Validar información de la sentencia CREATE NICKNAME.

Uso El reiniciador puede implementar esta función de miembro en la subclase de apodo protegida específica de reiniciador. Se debe implementar este método o el mismo método de la clase protegida si se soportan las opciones de columna o apodo específicas de reiniciador. Dado que la función verify_my_register_nickname_info forma parte del espacio de proceso fiable, esta función de miembro no puede interactuar con la fuente de datos remota. Si es necesaria la interacción para verificar la información de apodo, se debe implementar la función de miembro verify_my_register_nickname_info de la clase Fenced_Generic_Nickname.

El reiniciador comprueba si se ha asignado anteriormente un objeto a_delta_info antes de asignar uno él mismo. Este método se llama después de llamar a Fenced_Generic_Nickname::verify_my_register_nickname_info.

Sintaxis

```
virtual sqlint32 verify_my_register_nickname_info
       (Nickname_Info*   a_nick_info,
        Nickname_Info**  a_delta_info)
```

Argumentos de entrada

Tabla 194. Argumentos de entrada para la función de miembro verify_my_register_nickname_info

Nombre	Tipo de datos	Descripción
a_nick_info	Nickname_Info*	Información de la sentencia CREATE NICKNAME.

Argumentos de salida

Tabla 195. Argumentos de salida para la función de miembro verify_my_register_nickname_info

Nombre	Tipo de datos	Descripción
a_delta_info	Nickname_Info**	Información que el reiniciador añade.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función verify_my_alter_nickname_info

Finalidad

Validar información de la sentencia ALTER NICKNAME.

Uso El reiniciador puede implementar esta función de miembro en la subclase de apodo específica de reiniciador. Se debe implementar esta función de miembro si se soportan las opciones de columna o apodo específicas de reiniciador. Dado que la función verify_my_alter_nickname_info forma parte del espacio de proceso fiable, esta función de miembro no puede interactuar con la fuente de datos remota.

El reiniciador comprueba si se ha asignado anteriormente un objeto a_delta_info antes de asignar uno él mismo.

Sintaxis

```
virtual sqlint32 verify_my_alter_nickname_info
       (Nickname_Info*  a_nick_info,
        Nickname_Info** a_delta_info)
```

Argumentos de entrada

Tabla 196. Argumentos de entrada para la función de miembro verify_my_alter_nickname_info

Nombre	Tipo de datos	Descripción
a_nick_info	Nickname_Info*	Información de la sentencia ALTER NICKNAME.

Argumentos de salida

Tabla 197. Argumentos de salida para la función de miembro verify_my_alter_nickname_info

Nombre	Tipo de datos	Descripción
a_delta_info	Nickname_Info**	Información que el reiniciador añade.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_card

Finalidad

Recuperar la cardinalidad para un apodo. La cardinalidad se almacena en el catálogo de sistema DB2 Information Integrator.

Sintaxis

```
void  get_card (sqlint64* a_cardinality) const
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 198. Argumentos de salida para la función de miembro get_card

Nombre	Tipo de datos	Descripción
a_cardinality	sqlint64*	Cardinalidad.

Valor de retorno

Ninguno.

Función get_advance_cost

Finalidad

Recuperar el valor de la opción de apodo ADVANCE_COST que se utiliza en el modelo de coste por omisión.

Sintaxis

```
sqlint32 get_advance_cost (void) const
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Valor de la opción de apodo ADVANCE_COST o valor por omisión si no se especifica la opción para el apodo.

Función get_setup_cost

Finalidad

Recuperar el valor de la opción de apodo SETUP_COST que se utiliza en el modelo de coste por omisión.

Sintaxis

```
sqlint32 get_setup_cost (void) const
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Valor de la opción de apodo SETUP_COST o valor por omisión si no se especifica la opción para el apodo.

Función get_submission_cost

Finalidad

Recuperar el valor de la opción de apodo SUBMISSION_COST que se utiliza en el modelo de coste por omisión.

Sintaxis

```
sqlint32 get_submission_cost (void) const
```

Argumentos de entrada

Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Valor de la opción de apodo SUBMISSION_COST o valor por omisión si no se especifica la opción para el apodo.

Consulta relacionada:

- "Clases de apodo para la API C++" en la página 115

Clase Fenced_Generic_Nickname

Este tema describe la clase Fenced_Generic_Nickname y proporciona detalles para el constructor y las funciones de miembro.

Visión general

La clase Fenced_Generic_Nickname representa un apodo en el espacio de proceso protegido (no fiable). Esta clase es responsable de validar la información de la sentencia CREATE NICKNAME.

La clase Fenced_Generic_Nickname es una de las clases de apodo para la API C++.

Uso El reiniciador debe crear una subclase de esta clase y crea una instancia de esta clase en el método create_nickname() de la subclase específica de reiniciador de Fenced_Generic_Server.

Archivo
sqlqg_fenced_generic_nickname.h

Miembros de datos

La tabla siguiente lista los miembros de datos que puede utilizar con la clase Fenced_Generic_Nickname.

Tabla 199. Miembros de datos para la clase Fenced_Generic_Nickname

Nombre	Tipo de datos	Descripción
name	sqluint8*	Serie terminada por nulo que contiene el nombre (local) del apodo.
schema	sqluint8*	Serie terminada por nulo que contiene el nombre (local) del esquema.
server	Server*	Puntero al objeto de servidor que es propietario de este apodo.

Constructores y funciones de miembro

Las tablas siguientes describen el constructor y las funciones de miembro de la clase Fenced_Generic_Nickname. El constructor y las funciones se describen más detalladamente después de las tablas.

Tabla 200. Constructores para la clase Fenced_Generic_Nickname

Constructor	Descripción
Fenced_Generic_Nickname	Construir una instancia de Fenced_Generic_Nickname.

Tabla 201. Funciones de miembro para la clase Fenced_Generic_Nickname

Función de miembro	Descripción
get_local_name	Devolver puntero al nombre de apodo local terminado por nulo.
get_local_schema	Devolver puntero al nombre de esquema local terminado por nulo.
get_server	Devolver puntero al objeto de servidor que la contiene.
is_reserved_nickname_option	Determinar si un nombre de opción concreto es una de las opciones de apodo incorporadas de DB2.
is_reserved_column_option	Determinar si un nombre de opción concreto es una de las opciones de columna incorporadas de DB2.
initialize_my_nickname	Inicializar el apodo con información válida del catálogo.
verify_my_register_nickname_info	Validar información de la sentencia CREATE NICKNAME.

Constructor Fenced_Generic_Nickname

Finalidad

Construir una instancia de Fenced_Generic_Nickname.

Sintaxis

```
Fenced_Generic_Nickname (sqluint8*             a_schema,
                         sqluint8*             a_name,
                         Fenced_Generic_Server* a_server,
                         sqlint32*             a_rc)
```

Argumentos de entrada

Tabla 202. Argumentos de entrada para el constructor Fenced_Generic_Nickname

Nombre	Tipo de datos	Descripción
a_schema	sqluint8*	Nombre de esquema terminado por nulo.
a_name	sqluint8*	Nombre de apodo (local) terminado por nulo.
a_server	Fenced_Generic_Server*	Puntero al objeto de servidor que lo contiene.

Argumentos de salida

Tabla 203. Argumentos de salida para el constructor Fenced_Generic_Nickname

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Puntero al código de retorno; 0 indica que la operación se ha realizado satisfactoriamente.

Valor de retorno

Ninguno.

Función get_local_name

Finalidad

Devolver puntero al nombre de apodo local terminado por nulo.

Sintaxis

```
sqluint8* get_local_name ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Nombre de apodo terminado por nulo.

Función get_local_schema

Finalidad

Devolver puntero al nombre de esquema local terminado por nulo.

Sintaxis

```
sqluint8* get_local_schema ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Nombre de esquema terminado por nulo.

Función get_server

Finalidad

Devolver puntero al objeto de servidor que la contiene.

Sintaxis

```
Server* get_server ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Puntero al objeto de servidor que la contiene.

Función is_reserved_nickname_option

Finalidad

Determinar si un nombre de opción concreto es una de las opciones de apodo incorporadas de DB2.

Sintaxis

```
virtual sqluint32 is_reserved_nickname_option (sqluint8* a_opt_name)
```

Argumentos de entrada

Tabla 204. Argumentos de entrada para la función de miembro is_reserved_nickname_option

Nombre	Tipo de datos	Descripción
a_opt_name	sqluint8*	Serie de nombre de opción terminada por nulo.

Argumentos de salida

Ninguno.

Valor de retorno

Un valor de 0 indica que la opción no es una opción reservada. Un valor de retorno distinto de cero indica que la opción es una opción de apodo reservada.

Función is_reserved_column_option

Finalidad

Determinar si un nombre de opción concreto es una de las opciones de columna incorporadas de DB2.

Sintaxis

```
virtual sqluint32 is_reserved_column_option (sqluint8* a_opt_name)
```

Argumentos de entrada

Tabla 205. Argumentos de entrada para la función de miembro is_reserved_column_option

Nombre	Tipo de datos	Descripción
a_opt_name	sqluint8*	Serie de nombre de opción terminada por nulo.

Argumentos de salida

Ninguno.

Valor de retorno

Un valor de 0 indica que la opción no es una opción reservada. Un valor de retorno distinto de cero indica que la opción es una opción de columna reservada.

Función initialize_my_nickname

Finalidad

Inicializar el apodo con información válida del catálogo.

Uso El reiniciador puede implementar esta función de miembro en la subclase de apodo protegida específica de reiniciador.

Sintaxis

```
virtual sqlint32 initialize_my_nickname (Nickname_Info*  a_nick_info)
```

Argumentos de entrada

Tabla 206. Argumentos de entrada para la función de miembro initialize_my_nickname

Nombre	Tipo de datos	Descripción
a_nick_info	Nickname_Info*	Información del catálogo.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función verify_my_register_nickname_info

Finalidad
Validar información de la sentencia CREATE NICKNAME.

Uso El reiniciador puede implementar esta función de miembro en la subclase de apodo protegida específica de reiniciador. Se debe implementar este método o el mismo método de la clase desprotegida si se soportan las opciones de columna o apodo específicas de reiniciador. Implemente este método si el reiniciador debe interactuar con la fuente de datos remota al verificar información de apodos.

El reiniciador comprueba si se ha asignado anteriormente un objeto a_delta_info antes de asignar uno él mismo. Este método se llama antes de llamar a Unfenced_Generic_Nickname::verify_my_register_nickname_info.

Sintaxis

```
virtual sqlint32 verify_my_register_nickname_info
                (Nickname_Info*  a_nick_info,
                 Nickname_Info** a_delta_info)
```

Argumentos de entrada

Tabla 207. Argumentos de entrada para la función de miembro verify_my_register_nickname_info

Nombre	Tipo de datos	Descripción
a_nick_info	Nickname_Info*	Información de la sentencia CREATE NICKNAME.

Argumentos de salida

Tabla 208. Argumentos de salida para la función de miembro verify_my_register_nickname_info

Nombre	Tipo de datos	Descripción
a_delta_info	Nickname_Info**	Información que el reiniciador añade.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Consulta relacionada:

- "Clases de apodo para la API C++" en la página 115

Clase Remote_Connection

Este tema describe la clase Remote_Connection y proporciona detalles para el constructor y las funciones de miembro.

Visión general

La clase Remote_Connection representa una conexión entre DB2 y la fuente de datos remota. Esta clase gestiona la conexión, crea los objetos de operación remotos y mantiene los objetos de operación remotos.

La clase Remote_Connection es una clase de conexión para la API C++.

Uso El reiniciador puede crear una subclase de esta clase para crear subclases de conexión específicas del reiniciador. El método create_remote_connection() de la subclase Fenced_Generic_Server específica del reiniciador crea una instancia de esta clase.

Archivo

sqlqg_connection.h

Miembros de datos

La tabla siguiente lista los miembros de datos que puede utilizar con la clase Remote_Connection.

Tabla 209. Miembros de datos para la clase Remote_Connection

Nombre	Tipo de datos	Descripción
kind	connection_kind	Clase de conexión (1 fase o 2 fases).
server	FencedServer*	Puntero al servidor que es propietario de esta conexión.
user	FencedRemote_User*	El usuario que está asociado con la conexión.
connect	unsigned short	Distintivo que indica que la conexión está activa.

Constructores y funciones de miembro

Las tablas siguientes describen el constructor y las funciones de miembro de la clase Remote_Connection. El constructor y las funciones se describen más detalladamente después de las tablas.

Tabla 210. Constructores para la clase Remote_Connection

Constructor	Descripción
Remote_Connection	Construir un objeto Remote_Connection.

Tabla 211. Funciones de miembro para la clase Remote_Connection

Función de miembro	Descripción
connect	Establecer una conexión con la fuente de datos remota.
disconnect	Anular una conexión con la fuente de datos remota.
is_connected	Indicar si la conexión está activa o inactiva.
create_remote_query	Crear una subclase apropiada de Remote_Query.
create_remote_passthru	Crear una subclase apropiada de Remote_Passthru.
get_server	Devolver un punto al servidor que la contiene.
get_user	Devolver un puntero al objeto Remote_User asociado.
connect	Indicar que una transacción se ha realizado satisfactoriamente y que, a continuación, la fuente de datos remota confirma la transacción.
rollback	Indicar que una transacción no se ha realizado satisfactoriamente y que, a continuación, la fuente de datos remota retrotrae la transacción.

Constructor Remote_Connection

Finalidad

Construir un objeto Remote_Connection.

Sintaxis

```
Remote_Connection (FencedServer*      remote_server,
                   FencedRemote_User* remote_user,
                   connection_kind    k=one_phase_kind,
                   sqlint32*          rc = 0)
```

Argumentos de entrada

Tabla 212. Argumentos de entrada para el constructor Remote_Connection

Nombre	Tipo de datos	Descripción
FencedServer*	remote_server	Servidor propietario de esta conexión.
remote_user	FencedRemote_User*	Usuario que está asociado con la conexión.
k	connection_kind	Clase de conexión.

Argumentos de salida

Tabla 213. Argumentos de salida para el constructor Remote_Connection

Nombre	Tipo de datos	Descripción
rc	sqlint32*	Puntero al código de retorno; 0 indica que la operación se ha realizado satisfactoriamente.

Valor de retorno

Ninguno.

Función connect

Finalidad

Establecer una conexión con la fuente de datos remota.

Uso DB2 Information Integrator invoca esta función de miembro para abrir una conexión con la fuente de datos remota. El reiniciador debe implementar esta función de miembro en la subclase Remote_Connection específica de reiniciador. El comportamiento por omisión es informar de un error.

Sintaxis

```
virtual sqlint32 connect ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función disconnect

Finalidad

Anular una conexión con la fuente de datos remota.

Uso DB2 Information Integrator invoca esta función de miembro para cerrar una conexión con la fuente de datos remota. El reiniciador debe implementar esta función de miembro en la subclase Remote_Connection específica de reiniciador. El comportamiento por omisión informa de un error.

Sintaxis

```
virtual sqlint32 disconnect ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función is_connected

Finalidad

Indicar si la conexión está activa o inactiva.

Sintaxis

```
unsigned short is_connected ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Un valor de TRUE, si la conexión está activa. De lo contrario, el valor de retorno es FALSE.

Función create_remote_query

Finalidad

Crear una subclase apropiada de Remote_Query.

Uso DB2 Information Integrator invoca esta función de miembro para crear

una instancia de la subclase Remote_Query específica de reiniciador. El reiniciador debe implementar esta función de miembro en la subclase Remote_Connection específica de reiniciador.

Sintaxis

```
virtual sqlint32 create_remote_query
                (Runtime_Operation* runtime_query,
                 Remote_Query**     query)
```

Argumentos de entrada

Tabla 214. Argumentos de entrada para la función de miembro create_remote_query

Nombre	Tipo de datos	Descripción
runtime_query	Runtime_Operation*	Estructura interna de DB2 que describe la operación.
query	Remote_Query**	Puntero al objeto Remote_Query asignado.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función create_remote_passthru

Finalidad

Crear una subclase apropiada de Remote_Passthru.

Uso DB2 Information Integrator invoca esta función de miembro para crear una instancia de la subclase Remote_Passthru específica de reiniciador. El reiniciador puede implementar esta función de miembro en la subclase Remote_Connection específica de reiniciador. Esta función de miembro sólo es necesaria si el reiniciador soporta la modalidad PASSTHRU.

Sintaxis

```
virtual sqlint32 create_remote_passthru
                (Runtime_Operation*   runtime_passthru,
                 Remote_Passthru**    query)
```

Argumentos de entrada

Tabla 215. Argumentos de entrada para la función de miembro create_remote_passthru

Nombre	Tipo de datos	Descripción
runtime_passthru	Runtime_Operation*	Estructura interna de DB2 que describe la operación.
query	Runtime_Passthru**	Puntero al objeto Remote_Passthru asignado.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_server

Finalidad
Devolver un punto al servidor que la contiene.

Sintaxis

```
FencedServer* get_server()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al servidor.

Función get_user

Finalidad
Devolver un puntero al objeto Remote_User asociado.

Sintaxis

```
FencedRemote_User* get_user ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al objeto de usuario remoto.

Función commit

Finalidad
Indicar que una transacción se ha realizado satisfactoriamente y que, a continuación, la fuente de datos remota confirma la transacción.

Uso DB2 Information Integrator invoca esta función de miembro para indicar que una transacción se ha realizado satisfactoriamente. El reiniciador debe implementar esta función de miembro en la subclase Remote_Connection específica de reiniciador. El comportamiento por omisión es informar de un error.

Sintaxis

```
sqlint32 commit ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función rollback

Finalidad
Indicar que una transacción no se ha realizado satisfactoriamente y que, a continuación, la fuente de datos remota retrotrae la transacción.

Uso DB2 Information Integrator invoca esta función de miembro para indicar que una transacción no se ha realizado satisfactoriamente y que la fuente de datos remota retrotrae la transacción. El reiniciador puede implementar esta función de miembro en la subclase Remote_Connection específica de reiniciador. El comportamiento por omisión es informar de un error.

Sintaxis

```
sqlint32  rollback ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Consulta relacionada:

- "Clases de servidor para la API C++" en la página 85

Clases de operación para la API C++

La tabla siguiente describe cada clase de operación para la API C++.

Tabla 216. Clases de operación

Nombre de clase	Descripción
Remote_Query	Esta clase encapsula la información que es necesaria para ejecutar una consulta en la fuente de datos remota. El reiniciador debe crear una subclase de esta clase para implementar la operación.
Remote_Passthru	Esta clase representa una operación de paso a través en una fuente de datos remota. Un reiniciador debe crear una subclase de la clase Remote_Passthru para soportar las operaciones de paso de través.

Consulta relacionada:

- "Clase Remote_Query" en la página 138
- "Clase Remote_Passthru" en la página 149

Clase Remote_Query

Este tema describe la clase Remote_Query y proporciona detalles para el constructor y las funciones de miembro.

Visión general

La clase Remote_Query encapsula la información que es necesaria para ejecutar una consulta en la fuente de datos remota. El reiniciador debe crear una subclase de esta clase para implementar la operación.

La clase Remote_Query es una de las clases de operación para la API C++.

Uso El reiniciador crea una instancia de esta clase en el método create_remote_query() de la subclase Remote_Connection específica de reiniciador.

Archivo
sqlqg_operation.h

Clase padre
Remote_Operation.

Miembros de datos
Ninguno.

Constructores y funciones de miembro

Las tablas siguientes describen el constructor y las funciones de miembro de la clase Remote_Query. El constructor y las funciones se describen más detalladamente después de las tablas.

Tabla 217. Constructores para la clase Remote_Query

Constructor	Descripción
Remote_Query	Construir un objeto Remote_Query.

Tabla 218. Funciones de miembro para la clase Remote_Query

Función de miembro	Descripción
get_connection	Recuperar un puntero a la conexión que es propietaria de esta operación.
get_server	Recuperar un puntero al servidor que es propietario de esta operación.
get_input_data	Recuperar un puntero a la lista de valores de entrada para la operación.
get_output_data	Recuperar un puntero a la lista de almacenamientos intermedios de datos de salida.
get_exec_desc	Recuperar un puntero y la longitud para el descriptor de ejecución de esta consulta.
open	Abrir una consulta.
reopen	Volver a abrir una consulta.
fetch	Buscar y cargar una fila individual.
close	Cerrar una consulta.
report_eof	Informar de una condición de fin de archivo durante una búsqueda y carga.
get_status	Recuperar el estado de consulta actual.
set_status	Establecer el estado de consulta actual.
fetch_lob	Recuperar datos LOB en un almacenamiento intermedio LOB especial que proporciona DB2.

Tabla 218. Funciones de miembro para la clase Remote_Query (continuación)

Función de miembro	Descripción
set_lob_next	Suspender el proceso de las columnas no LOB e iniciar el proceso de una o más columnas LOB.
lob_data_ready	Indica que el reiniciador ha transferido una sección de datos LOB al almacenamiento intermedio.

Constructor Remote_Query

Finalidad

Construir un objeto Remote_Query.

Sintaxis

```
Remote_Query (Remote_Connection* a_active_connection,
              Runtime_Operation* a_runtime_info,
              sqlint32*          a_rc)
```

Argumentos de entrada

Tabla 219. Argumentos de entrada para el constructor Remote_Query

Nombre	Tipo de datos	Descripción
a_active_connection	Remote_Connection*	Conexión para esta consulta.
a_runtime_info	Runtime_Operation*	Datos internos para esta consulta.

Argumentos de salida

Tabla 220. Argumentos de salida para el constructor Remote_Query

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Código de retorno; 0 indica que la operación se ha realizado satisfactoriamente.

Valor de retorno

Ninguno.

Función get_connection

Dónde se define

Remote_Operation

Finalidad
Recuperar un puntero a la conexión que es propietaria de esta operación.

Sintaxis

```
Remote_Connection* get_connection ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al objeto Remote_Connection.

Función get_server

Dónde se define
Remote_Operation

Finalidad
Recuperar un puntero al servidor que es propietario de esta operación.

Sintaxis

```
Fenced_Generic_Server* get_server ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al objeto de servidor.

Función get_input_data

Dónde se define
Remote_Operation

Finalidad
Recuperar un puntero a la lista de valores de entrada para la operación.

Sintaxis

```
Runtime_Data_List* get_input_data ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la lista de valores de datos.

Función get_output_data

Dónde se define
Remote_Operation

Finalidad
Recuperar un puntero a la lista de almacenamientos intermedios de datos de salida.

Sintaxis

```
Runtime_Data_List* get_output_data ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la lista de almacenamientos intermedios de datos.

Función get_exec_desc

Dónde se define
Remote_Operation

Finalidad
Recuperar un puntero y la longitud para el descriptor de ejecución de esta consulta.

Sintaxis

```
void get_exec_desc (void** a_exec_desc,
                    int*   a_exec_desc_len)
```

Argumentos de entrada
Ninguno.

Argumentos de salida

Tabla 221. Argumentos de salida para la función de miembro get_exec_desc

Nombre	Tipo de datos	Descripción
a_exec_desc	void**	Puntero al descriptor de ejecución.
a_exec_desc_len	int*	Longitud del descriptor.

Valor de retorno
Ninguno.

Función open

Finalidad
Abrir una consulta.

Uso DB2 Information Integrator invoca esta función de miembro para iniciar una consulta. El reiniciador debe implementar esta función de miembro en la subclase Remote_Query específica de reiniciador.

Sintaxis

```
sqlint32 open ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función reopen

Finalidad
Volver a abrir una consulta

Uso DB2 Information Integrator invoca esta función de miembro para iniciar una consulta con valores de parámetro nuevos. El reiniciador debe implementar esta función de miembro en la subclase Remote_Query específica de reiniciador.

Sintaxis

```
sqlint32 reopen (sqlint16 a_status)
```

Argumentos de entrada

Tabla 222. Argumentos de entrada para la función de miembro reopen

Nombre	Tipo de datos	Descripción
a_status	sqlint16	No se utiliza.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función fetch

Finalidad
Buscar y cargar una fila individual.

Uso DB2 Information Integrator invoca esta función de miembro para buscar y cargar una fila de datos no LOB de una consulta remota. El reiniciador debe implementar esta función de miembro en la subclase Remote_Query específica de reiniciador.

Sintaxis

```
sqlint32 fetch ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función close

Finalidad
Cerrar una consulta.

Uso DB2 Information Integrator invoca esta función de miembro para cerrar un cursor de consulta. El reiniciador debe implementar esta función de miembro en la subclase Remote_Query específica de reiniciador.

Sintaxis

```
sqlint32 close (sqlint16 a_status)
```

Argumentos de entrada

Tabla 223. Argumentos de entrada para la función de miembro close

Nombre	Tipo de datos	Descripción
a_status	sqlint16	Estado (SQLQG_CLOSE_EOS, SQLQG_CLOSE_EOA o SQLQG_CLOSE_EOQ). SQLQG_CLOSE_EOS indica que el reiniciador deja el estado de forma que se pueda llamar a la función reopen(). SQLQG_CLOSE_EOA indica que el reiniciador puede finalizar todo el proceso necesario. SQLQG_CLOSE_EOQ no se utiliza.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función report_eof

Finalidad

Informar de una condición de fin de archivo durante una búsqueda y carga.

Uso Esta función de miembro la invoca la clase Remote_Query específica de reiniciador durante el método fetch() para indicar que se ha buscado y cargado la última fila.

Sintaxis

```
sqlint32 report_eof ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno
Código de retorno. Este código se devuelve de fetch().

Función get_status

Finalidad
Recuperar el estado de consulta actual.

Sintaxis

```
sqluint8 get_status ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Estado (SQLQG_UNREADY, SQLQG_READY, SQLQG_OPEN o SQLQG_EOF).

Función set_status

Finalidad
Establecer el estado de consulta actual.

Sintaxis

```
void set_status (sqluint8 a_new_status )
```

Argumentos de entrada

Tabla 224. Argumentos de entrada para la función de miembro set_status

Nombre	Tipo de datos	Descripción
a_new_status	sqluint8	Estado (SQLQG_UNREADY, SQLQG_READY, SQLQG_OPEN o SQLQG_EOF).

Argumentos de salida
Ninguno.

Valor de retorno
Ninguno.

Función fetch_lob

Finalidad

Recuperar datos LOB en un almacenamiento intermedio LOB especial que proporciona DB2.

Sintaxis

```
virtual sqlint32 fetch_lob (unsigned char* a_buffer,
                            squint32       a_buffer_size,
                            squint32       a_bytes_written)
```

Argumentos de entrada

Tabla 225. Argumentos de entrada para la función de miembro fetch_lob

Nombre	Tipo de datos	Descripción
a_buffer	unsigned char*	Dirección del almacenamiento intermedio de datos LOB proporcionado por DB2.
a_buffer_size	squint32	Tamaño (en bytes) del almacenamiento intermedio.
a_bytes_written	squint32	Número total de bytes que las llamadas anteriores de la función fetch_lob han grado para la columna actual.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función set_lob_next

Finalidad

Suspender el proceso de las columnas no LOB e iniciar el proceso de una o más columnas LOB. Esta función se llama desde la función fetch.

Sintaxis

```
void set_lob_next (void)
```

Argumentos de entrada

Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Ninguno.

Función lob_data_ready

Finalidad
Indica que el reiniciador ha transferido una sección de datos LOB al almacenamiento intermedio.

Sintaxis

```
sqlint32 lob_data_ready (sqlint32          a_col_number,
                         sqluint32         a_bytes_ready,
                         sqlqg_lob_status  a_status,
                         sqlqg_lob_intent  a_next)
```

Argumentos de entrada

Tabla 226. Argumentos de entrada para la función de miembro lob_data_ready

Nombre	Tipo de datos	Descripción
a_col_number	sqlint32	Columna de la que el reiniciador ha tomado datos para grabarlos en el almacenamiento intermedio LOB.
a_bytes_ready	sqluint32	Número de bytes que el reiniciador ha copiado en el almacenamiento intermedio LOB.
a_status	sqlqg_lob_status	Tipo de datos enumerado que indica el estado. El valor es LOB_LAST si el reiniciador está procesando la última sección de la columna. De lo contrario, el valor es LOB_MORE.
a_next	sqlqg_lob_intent	Tipo de datos enumerado que indica si la siguiente llamada del reiniciador es una llamada a la función fetch o a la función fetch_lob. Los valores válidos son LOB_NEXT, NON_LOB_NEXT y ROW_DONE.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Consulta relacionada:

- "Clases de operación para la API C++" en la página 138

Clase Remote_Passthru

Este tema describe la clase Remote_Passthru y proporciona detalles para el constructor y las funciones de miembro.

Visión general

La clase Remote_Passthru representa una operación de paso a través en una fuente de datos remota. Un reiniciador debe crear una subclase de la clase Remote_Passthru para soportar las operaciones de paso de través.

La clase Remote_Passthru es una de las clases de operación para la API C++.

Uso El método create_remote_passthru() de la subclase Remote_Connection específica de reiniciador crea una instancia de esta clase.

Archivo
sqlqg_operation.h

Clase padre
Remote_Operation

Miembros de datos
Ninguno.

Constructores y funciones de miembro

Las tablas siguientes describen el constructor y las funciones de miembro de la clase Remote_Passthru. El constructor y las funciones se describen más detalladamente después de las tablas.

Tabla 227. Constructores para la clase Remote_Passthru

Constructor	Descripción
Remote_Passthru	Construir un objeto Remote_Passthru.

Tabla 228. Funciones de miembro para la clase Remote_Passthru

Función de miembro	Descripción
get_connection	Recuperar un puntero a la conexión que es propietaria de esta operación.
get_server	Recuperar un puntero al servidor que es propietario de esta operación.
get_input_data	Recuperar un puntero a la lista de valores de entrada para la operación.
get_output_data	Recuperar un puntero a la lista de almacenamientos intermedios de datos de salida.
get_SQL_statement	Recuperar la sentencia para la ejecución de paso de través.
report_eof	Informar de una condición de fin de archivo durante una búsqueda y carga.
prepare	Preparar una operación remota de paso a través.
describe	Describir una operación remota de paso a través.
execute	Ejecutar una operación remota de paso a través.
open	Abrir un cursor para una operación remota de paso a través.
fetch	Buscar y cargar una fila en una operación remota de paso a través.
close	Cerrar un cursor para una operación remota de paso a través.

Constructor Remote_Passthru

Finalidad

Construir un objeto Remote_Passthru.

Sintaxis

```
Remote_Passthru (Remote_Connection* a_active_connection,
                 Runtime_Operation* a_runtime_passthru,
                 sqlint32*          a_rc)
```

Argumentos de entrada

Tabla 229. Argumentos de entrada para el constructor Remote_Passthru

Nombre	Tipo de datos	Descripción
a_active_connection	Remote_Connection*	Conexión para esta consulta.
a_runtime_info	Runtime_Operation*	Datos internos para la consulta.

Argumentos de salida

Tabla 230. Argumentos de salida para el constructor Remote_Passthru

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Código de retorno; 0 indica que la operación se ha realizado satisfactoriamente.

Valor de retorno

Ninguno.

Función get_connection

Dónde se define

Remote_Operation

Finalidad

Recuperar un puntero a la conexión que es propietaria de esta operación.

Sintaxis

```
Remote_Connection* get_connection ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Puntero al objeto Remote_Connection.

Función get_server

Dónde se define

Remote_Operation

Finalidad
Recuperar un puntero al servidor que es propietario de esta operación.

Sintaxis

```
Fenced_Generic_Server* get_server ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al objeto de servidor.

Función get_input_data

Dónde se define
Remote_Operation

Finalidad
Recuperar un puntero a la lista de valores de entrada para la operación.

Sintaxis

```
Runtime_Data_List* get_input_data ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la lista de valores de datos.

Función get_output_data

Dónde se define
Remote_Operation

Finalidad
Recuperar un puntero a la lista de almacenamientos intermedios de datos de salida.

Sintaxis

```
Runtime_Data_List* get_output_data ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la lista de almacenamientos intermedios de datos.

Función get_SQL_statement

Dónde se define
Remote_Operation

Finalidad
Recuperar la sentencia para la ejecución de paso de través.

Sintaxis

```
char* get_SQL_statement ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Sentencia (terminada por nulo).

Función report_eof

Dónde se define
Remote_Operation

Finalidad
Informar de una condición de fin de archivo durante una búsqueda y carga.

Sintaxis

```
sqlint32 report_eof ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Este código se devuelve de fetch().

Función prepare

Finalidad

Preparar una operación remota de paso a través.

Sintaxis

```
sqlint32 prepare (Runtime_Data_Desc_List* a_data_description_list)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 231. Argumentos de salida para la función de miembro prepare

Nombre	Tipo de datos	Descripción
a_data_description_list	Runtime_Data_Desc_List*	Una lista de Runtime_Data_Desc que describe los resultados de la operación de paso a través.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función describe

Finalidad

Describir una operación remota de paso a través.

Sintaxis

```
sqlint32 describe (Runtime_Data_Desc_List* a_data_description_list)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 232. Argumentos de salida para la función de miembro describe

Nombre	Tipo de datos	Descripción
a_data_description_list	Runtime_Data_Desc_List*	Una lista de Runtime_Data_Desc que describe los resultados de la operación de paso a través.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función execute

Finalidad

Ejecutar una operación remota de paso a través.

Sintaxis

```
sqlint32 execute ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función open

Finalidad

Abrir un cursor para una operación remota de paso a través.

Sintaxis

```
sqlint32 open ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función fetch

Finalidad

Buscar y cargar una fila en una operación remota de paso a través.

Sintaxis

```
sqlint32 fetch ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función close

Finalidad
Cerrar un cursor para una operación remota de paso a través.

Sintaxis

```
sqlint32 close ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Consulta relacionada:

- "Clases de operación para la API C++" en la página 138

Clases de petición para la API C++

La tabla siguiente describe cada clase de petición para la API C++.

Tabla 233. Clases de petición

Nombre de clase	Descripción
Request	Esta clase encapsula un fragmento de petición que el reiniciador debe analizar y procesar.
Reply	Esta clase representa una parte de una consulta que el reiniciador puede procesar. Se puede crear una subclase de esta clase si el reiniciador utiliza un modelo de coste distinto del valor por omisión.
Request_Exp	Esta clase describe un nodo de expresión de SQL. El nodo de expresión de SQL puede ser una expresión de cabecera (lista de selección) o parte de un predicado.
Request_Exp_Type	Esta clase representa un descriptor de tipo de datos para los nodos Request_Exp.

Tabla 233. Clases de petición (continuación)

Nombre de clase	Descripción
Request_Constant	Esta clase representa un valor de datos para un nodo Request_Exp con un tipo de Request_Exp::constant.
Predicate_List	Esta clase encapsula una lista de predicados y la utiliza el protocolo RRC.

Consulta relacionada:

- "Clase Request_Exp" en la página 182
- "Clase Request" en la página 157
- "Clase Request_Constant" en la página 193
- "Clase Request_Exp_Type" en la página 190
- "Clase Reply" en la página 164
- "Clase Predicate_List" en la página 198

Clase Request

Este tema describe la clase Request y proporciona detalles para cada función de miembro.

Visión general

Esta clase Request encapsula un fragmento de petición que el reiniciador debe analizar y procesar.

La clase Request es una de las clases de petición para la API C++.

Uso El reiniciador no crea nunca instancias de esta clase.

Archivo
sqlqg_request.h

Clase padre
Parsed_Query_Fragment

Miembros de datos
Ninguno.

Funciones de miembro

La tabla siguiente describe cada función de miembro de la clase Request. Después de la tabla, se describe cada función más detalladamente.

Tabla 234. Funciones de miembro para la clase Request

Función de miembro	Descripción
get_number_of_quantifiers	Recuperar el número de cuantificadores para esta consulta.
get_number_of_predicates	Recuperar el número de predicados para esta consulta.
get_number_of_head_exp	Recuperar el número de expresiones de cabecera (lista de selección) para esta consulta.
get_quantifier_handle	Recuperar el manejador para el cuantificador que se encuentra en la posición número *n* de esta consulta.
get_predicate_handle	Recuperar el manejador para el predicado que se encuentra en la posición número *n* de esta consulta.
get_head_exp_handle	Recuperar el manejador para la expresión de cabecera que se encuentra en la posición número *n* de esta consulta.
get_nickname	Recuperar el objeto de apodo que está asociado con el manejador especificado.
get_head_exp	Recuperar una expresión de cabecera que está asociada con el manejador especificado.
get_predicate	Recuperar una expresión de predicado que está asociada con el manejador especificado.
get_distinct	Recuperar el distintivo DISTINCT para una consulta.

Función get_number_of_quantifiers

Dónde se define

Parsed_Query_Fragment

Finalidad

Recuperar el número de cuantificadores (apodos y expresiones de tabla) para esta consulta.

Sintaxis

```
int get_number_of_quantifiers ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Número de cuantificadores.

Función get_number_of_predicates

Dónde se define
Parsed_Query_Fragment

Finalidad
Recuperar el número de predicados para esta consulta.

Sintaxis

```
int get_number_of_predicates ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Número de predicados.

Función get_number_of_head_exp

Dónde se define
Parsed_Query_Fragment

Finalidad
Recuperar el número de expresiones de cabecera (lista de selección) para esta consulta.

Sintaxis

```
int get_number_of_head_exp ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Número de expresiones de cabecera.

Función get_quantifier_handle

Dónde se define

Parsed_Query_Fragment

Finalidad

Recuperar el manejador para el cuantificador que se encuentra en la posición número *n* de esta consulta.

Sintaxis

```
sqlint32 get_quantifier_handle (int  a_quant_pos,
                                int* a_quant_handle)
```

Argumentos de entrada

Tabla 235. Argumentos de entrada para la función de miembro get_quantifier_handle

Nombre	Tipo de datos	Descripción
a_quant_pos	int	Posición del manejador en la lista (empezando en 1).

Argumentos de salida

Tabla 236. Argumentos de salida para la función de miembro get_quantifier_handle

Nombre	Tipo de datos	Descripción
a_quant_handle	int*	Manejador.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_predicate_handle

Dónde se define

Parsed_Query_Fragment

Finalidad

Recuperar el manejador para el predicado que se encuentra en la posición número *n* de esta consulta.

Sintaxis

```
sqlint32 get_predicate_handle (int  a_pred_pos,
                               int* a_pred_handle )
```

Argumentos de entrada

Tabla 237. Argumentos de entrada para la función de miembro get_predicate_handle

Nombre	Tipo de datos	Descripción
a_pred_pos	int	Posición del manejador en la lista (empezando en 1).

Argumentos de salida

Tabla 238. Argumentos de salida para la función de miembro get_predicate_handle

Nombre	Tipo de datos	Descripción
a_pred_handle	int*	Manejador.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_head_exp_handle

Dónde se define

Parsed_Query_Fragment

Finalidad

Recuperar el manejador para la expresión de cabecera que se encuentra en la posición número *n* de esta consulta.

Sintaxis

```
sqlint32 get_head_exp_handle (int  a_head_exp_pos,
                              int* a_head_exp_handle)
```

Argumentos de entrada

Tabla 239. Argumentos de entrada para la función de miembro get_head_exp_handle

Nombre	Tipo de datos	Descripción
a_head_exp_pos	int	Posición del manejador en la lista (empezando en 1).

Argumentos de salida

Tabla 240. Argumentos de salida para la función de miembro get_head_exp_handle

Nombre	Tipo de datos	Descripción
a_head_exp_handle	int*	Manejador.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_nickname

Dónde se define

Parsed_Query_Fragment

Finalidad

Recuperar el objeto de apodo que está asociado con el manejador especificado.

Sintaxis

```
sqlint32 get_nickname (int a_quant_handle,
                       Unfenced_Generic_Nickname** a_nickname)
```

Argumentos de entrada

Tabla 241. Argumentos de entrada para la función de miembro get_nickname

Nombre	Tipo de datos	Descripción
a_quant_handle	int	Manejador para el apodo.

Argumentos de salida

Tabla 242. Argumentos de salida para la función de miembro get_nickname

Nombre	Tipo de datos	Descripción
a_nickname	Unfenced_Generic_Nickname**	Puntero al objeto de apodo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_head_exp

Dónde se define

Parsed_Query_Fragment

Finalidad

Recuperar una expresión de cabecera que está asociada con el manejador especificado.

Sintaxis

```
sqlint32 get_head_exp (int  a_head_exp_handle,
                       Request_Exp** a_head_exp)
```

Argumentos de entrada

Tabla 243. Argumentos de entrada para la función de miembro get_head_exp

Nombre	Tipo de datos	Descripción
a_head_exp_handle	int	Manejador para la expresión.

Argumentos de salida

Tabla 244. Argumentos de salida para la función de miembro get_head_exp

Nombre	Tipo de datos	Descripción
a_head_exp	Request_Exp**	Puntero al objeto de expresión.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_predicate

Dónde se define

Parsed_Query_Fragment

Finalidad

Recuperar una expresión de predicado que está asociada con el manejador especificado.

Sintaxis

```
sqlint32 get_predicate (int  a_pred_handle,
                        Request_Exp** a_pred_exp)
```

Argumentos de entrada

Tabla 245. Argumentos de entrada para la función de miembro get_predicate

Nombre	Tipo de datos	Descripción
a_pred_handle	int	Manejador para la expresión.

Argumentos de salida

Tabla 246. Argumentos de salida para la función de miembro get_predicate

Nombre	Tipo de datos	Descripción
a_pred_exp	Request_Exp**	Puntero al objeto de expresión.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_distinct

Dónde se define
Parsed_Query_Fragment

Finalidad
Recuperar el distintivo DISTINCT para una consulta.

Sintaxis

```
int get_distinct ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Distintivo Distinct.

Consulta relacionada:

- "Clases de petición para la API C++" en la página 156

Clase Reply

Este tema describe la clase Reply y proporciona detalles para el constructor y las funciones de miembro.

Visión general

La clase Reply representa una parte de una consulta que el reiniciador puede procesar. Se puede crear una subclase de esta clase si el reiniciador utiliza un modelo de coste distinto del modelo por omisión.

La clase Reply es una de las clases de petición para la API C++.

Uso El método create_reply() de la clase Unfenced_Generic_Server puede crear una instancia de esta clase. Si el reiniciador implementa una subclase de la clase Reply, se altera temporalmente el método create_reply() en la subclase específica de reiniciador de Unfenced_Generic_Server.

Archivo
sqlqg_request.h

Clase padre
Parsed_Query_Fragment

Miembros de datos

Ninguno.

Constructores y funciones de miembro

Las tablas siguientes describen el constructor y las funciones de miembro de la clase Reply. El constructor y las funciones se describen más detalladamente después de las tablas.

Tabla 247. Constructores para la clase Reply

Constructor	Descripción
Reply	Construir un objeto de respuesta vacío.

Tabla 248. Funciones de miembro para la clase Reply

Función de miembro	Descripción
get_number_of_quantifiers	Recuperar el número de cuantificadores para esta consulta.
get_number_of_predicates	Recuperar el número de predicados para esta consulta.
get_number_of_head_exp	Recuperar el número de expresiones de cabecera (lista de selección) para esta consulta.
get_quantifier_handle	Recuperar el manejador para el cuantificador que se encuentra en la posición número *n* de esta consulta.
get_predicate_handle	Recuperar el manejador para el predicado que se encuentra en la posición número *n* de esta consulta.
get_head_exp_handle	Recuperar el manejador para la expresión de cabecera que se encuentra en la posición número *n* de esta consulta.
get_nickname	Recuperar el objeto de apodo que está asociado con el manejador especificado.
get_head_exp	Recuperar una expresión de cabecera que está asociada con el manejador especificado.
get_predicate	Recuperar una expresión de predicado que está asociada con el manejador especificado.
get_distinct	Recuperar el distintivo DISTINCT para una consulta.

Tabla 248. Funciones de miembro para la clase Reply (continuación)

Función de miembro	Descripción
set_distinct	Establecer el distintivo DISTINCT para una consulta.
add_head_exp	Añadir una expresión de cabecera a la respuesta.
add_predicate	Añadir una expresión de predicado a la respuesta.
add_quantifier	Añadir un cuantificador a la respuesta.
add_order_entry	Añadir una especificación ORDER BY a la respuesta.
get_number_of_order_entries	Recuperar el número de entradas ORDER BY para la respuesta.
get_order_entry	Recuperar una entrada ORDER BY específica.
get_exec_desc	Recuperar el descriptor de ejecución que está asociado con la respuesta.
set_exec_desc	Almacenar un descriptor de ejecución en la respuesta.
get_parameter_order	Recuperar una lista de manejadores de parámetro.
cardinality	Recuperar la cardinalidad para el fragmento de consulta.
total_cost	Recuperar el coste total para un fragmento de consulta.
re_exec_cost	Coste para ejecutar un fragmento de consulta otra vez.
first_tuple_cost	Recuperar el coste para buscar y cargar la primera tupla.
all_costs	Recuperar todos los valores de coste de una sola vez.
next	Recuperar un puntero a la siguiente respuesta de una cadena de respuestas.
set_next_reply	Enlazar una nueva respuesta a la respuesta actual.
get_server	Recuperar un puntero al servidor que es propietario de la respuesta.

Constructor Reply

Finalidad

Construir un objeto de respuesta vacío.

Sintaxis

```
Reply (Request*                 a_rq,
       Unfenced_Generic_Server* a_server,
       sqlint32*                a_rc)
```

Argumentos de entrada

Tabla 249. Argumentos de entrada para el constructor Reply

Nombre	Tipo de datos	Descripción
a_rq	Request*	Petición de la que se obtiene esta respuesta.
a_server	Unfenced_Generic_Server*	Servidor para un protocolo de respuesta o petición.

Argumentos de salida

Tabla 250. Argumentos de salida para el constructor Reply

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Código de retorno; 0 indica que la operación se ha realizado satisfactoriamente.

Valor de retorno

Ninguno.

Función get_number_of_quantifiers

Dónde se define

Parsed_Query_Fragment

Finalidad

Recuperar el número de cuantificadores (apodos y expresiones de tabla) para esta consulta.

Sintaxis

```
int get_number_of_quantifiers ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno
Número de cuantificadores.

Función get_number_of_predicates

Dónde se define
Parsed_Query_Fragment

Finalidad
Recuperar el número de predicados para esta consulta.

Sintaxis

```
int get_number_of_predicates ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Número de predicados.

Función get_number_of_head_exp

Dónde se define
Parsed_Query_Fragment

Finalidad
Recuperar el número de expresiones de cabecera (lista de selección) para esta consulta.

Sintaxis

```
int get_number_of_head_exp ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Número de expresiones de cabecera.

Función get_quantifier_handle

Dónde se define
Parsed_Query_Fragment

Finalidad

Recuperar el manejador para el cuantificador que se encuentra en la posición número *n* de esta consulta.

Sintaxis

```
sqlint32 get_quantifier_handle (int  a_quant_pos,
                                int* a_quant_handle)
```

Argumentos de entrada

Tabla 251. Argumentos de entrada para la función de miembro get_quantifier_handle

Nombre	Tipo de datos	Descripción
a_quant_pos	int	Posición del manejador en la lista (empezando en 1).

Argumentos de salida

Tabla 252. Argumentos de salida para la función de miembro get_quantifier_handle

Nombre	Tipo de datos	Descripción
a_quant_handle	int*	Manejador.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_predicate_handle

Dónde se define

Parsed_Query_Fragment

Finalidad

Recuperar el manejador para el predicado que se encuentra en la posición número *n* de esta consulta.

Sintaxis

```
sqlint32 get_predicate_handle (int  a_pred_pos,
                               int* a_pred_handle)
```

Argumentos de entrada

Tabla 253. Argumentos de entrada para la función de miembro get_predicate_handle

Nombre	Tipo de datos	Descripción
a_pred_pos	int	Posición del manejador en la lista (empezando en 1).

Argumentos de salida

Tabla 254. Argumentos de salida para la función de miembro get_predicate_handle

Nombre	Tipo de datos	Descripción
a_pred_handle	int*	Manejador.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_head_exp_handle

Dónde se define

Parsed_Query_Fragment

Finalidad

Recuperar el manejador para la expresión de cabecera que se encuentra en la posición número *n* de esta consulta.

Sintaxis

```
sqlint32 get_head_exp_handle (int  a_head_exp_pos,
                              int* a_head_exp_handle)
```

Argumentos de entrada

Tabla 255. Argumentos de entrada para la función de miembro get_head_exp_handle

Nombre	Tipo de datos	Descripción
a_head_exp_pos	int	Posición del manejador en la lista (empezando en 1).

Argumentos de salida

Tabla 256. Argumentos de salida para la función de miembro get_head_exp_handle

Nombre	Tipo de datos	Descripción
a_head_exp_handle	int*	Manejador.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_nickname

Dónde se define

Parsed_Query_Fragment

Finalidad

Recuperar el objeto de apodo que está asociado con el manejador especificado.

Sintaxis

```
sqlint32 get_nickname (int                          a_quant_handle,
                       Unfenced_Generic_Nickname** a_nickname)
```

Argumentos de entrada

Tabla 257. Argumentos de entrada para la función de miembro get_nickname

Nombre	Tipo de datos	Descripción
a_quant_handle	int	Manejador para el apodo.

Argumentos de salida

Tabla 258. Argumentos de salida para la función de miembro get_nickname

Nombre	Tipo de datos	Descripción
a_nickname	Unfenced_Generic_Nickname**	Puntero al objeto de apodo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_head_exp

Dónde se define

Parsed_Query_Fragment

Finalidad

Recuperar una expresión de cabecera que está asociada con el manejador especificado.

Sintaxis

```
sqlint32 get_head_exp (int a         a_head_exp_handle,
                       Request_Exp** a_head_exp)
```

Argumentos de entrada

Tabla 259. Argumentos de entrada para la función de miembro get_head_exp

Nombre	Tipo de datos	Descripción
a_head_exp_handle	int	Manejador para la expresión.

Argumentos de salida

Tabla 260. Argumentos de salida para la función de miembro get_head_exp

Nombre	Tipo de datos	Descripción
a_head_exp	Request_Exp**	Puntero al objeto de expresión.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_predicate

Dónde se define

Parsed_Query_Fragment

Finalidad

Recuperar una expresión de predicado que está asociada con el manejador especificado.

Sintaxis

```
sqlint32 get_predicate (int           a_pred_handle,
                        Request_Exp** a_pred_exp)
```

Argumentos de entrada

Tabla 261. Argumentos de entrada para la función de miembro get_predicate

Nombre	Tipo de datos	Descripción
a_pred_handle	int	Manejador para la expresión.

Argumentos de salida

Tabla 262. Argumentos de salida para la función de miembro get_predicate

Nombre	Tipo de datos	Descripción
a_pred_exp	Request_Exp**	Puntero al objeto de expresión.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_distinct

Dónde se define
Parsed_Query_Fragment

Finalidad
Recuperar el distintivo DISTINCT para una consulta.

Sintaxis

```
int get_distinct ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Distintivo Distinct.

Función set_distinct

Dónde se define
Parsed_Query_Fragment

Finalidad
Establecer el distintivo DISTINCT para una consulta.

Sintaxis

```
void set_distinct (int a_distinct)
```

Argumentos de entrada

Tabla 263. Argumentos de entrada para la función de miembro set_distinct

Nombre	Tipo de datos	Descripción
a_distinct	int	Distintivo Distinct (1 para DISTINCT; 0 para no DISTINCT).

Argumentos de salida
Ninguno.

Valor de retorno
Ninguno.

Función add_head_exp

Finalidad
Añadir una expresión de cabecera a la respuesta.

Sintaxis

```
sqlint32 add_head_exp (int a_head_exp_handle)
```

Argumentos de entrada

Tabla 264. Argumentos de entrada para la función de miembro add_head_exp

Nombre	Tipo de datos	Descripción
a_head_exp_handle	int	Manejador para la expresión.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función add_predicate

Finalidad

Añadir una expresión de predicado a la respuesta.

Sintaxis

```
sqlint32 add_predicate_exp (int a_pred_handle)
```

Argumentos de entrada

Tabla 265. Argumentos de entrada para la función de miembro add_predicate

Nombre	Tipo de datos	Descripción
a_pred_handle	int	Manejador para la expresión.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función add_quantifier

Finalidad

Añadir un cuantificador a la respuesta.

Sintaxis

```
sqlint32 add_quantifier (int a_quant_handle)
```

Argumentos de entrada

Tabla 266. Argumentos de entrada para la función de miembro add_quantifier

Nombre	Tipo de datos	Descripción
a_quant_handle	int	Manejador para el cuantificador.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función add_order_entry

Finalidad

Añadir una especificación ORDER BY a la respuesta.

Sintaxis

```
sqlint32 add_order_entry (int                    a_gindex,
                          Reply::order_direction a_direction)
```

Argumentos de entrada

Tabla 267. Argumentos de entrada para la función de miembro add_order_entry

Nombre	Tipo de datos	Descripción
a_gindex	int	Índice de expresión de cabecera para la ordenación (no un manejador).
a_direction	Reply_order_direction	Dirección (ascendente o descendente) para la ordenación.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_number_of_order_entries

Finalidad

Recuperar el número de entradas ORDER BY para la respuesta.

Sintaxis

```
int get_number_of_order_entries ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Número de entradas ORDER BY.

Función get_order_entry

Finalidad

Recuperar una entrada ORDER BY específica.

Sintaxis

```
get_order_entry (int                    a_pos,
                 int*                   a_gindexP,
                 Reply::order_direction* a_direction)
```

Argumentos de entrada

Tabla 268. Argumentos de entrada para la función de miembro get_order_entry

Nombre	Tipo de datos	Descripción
a_pos	int	Posición de la entrada de orden (empezando en 1).

Argumentos de salida

Tabla 269. Argumentos de salida para la función de miembro get_order_entry

Nombre	Tipo de datos	Descripción
a_gindexP	int*	Índice de expresión de cabecera para la expresión ORDER BY.
a_direction	Reply::order_direction*	Dirección para la ordenación.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_exec_desc

Finalidad

Recuperar el descriptor de ejecución que está asociado con la respuesta.

Sintaxis

```
void get_exec_desc (void** a_exec_desc,
                    int*   a_exec_desc_size)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 270. Argumentos de salida para la función de miembro get_exec_desc

Nombre	Tipo de datos	Descripción
a_exec_desc	void**	Puntero al descriptor de ejecución.
a_exec_desc_size	int*	Tamaño del descriptor de ejecución.

Valor de retorno

Ninguno.

Función set_exec_desc

Finalidad

Almacenar un descriptor de ejecución en la respuesta.

Uso El almacenamiento para el descriptor de ejecución debe asignarse con Wrapper_Utillities::allocate.

Sintaxis

```
void set_exec_desc (void** a_exec_desc,
                    int*   a_exec_desc_size)
```

Argumentos de entrada

Tabla 271. Argumentos de entrada para la función de miembro set_exec_desc

Nombre	Tipo de datos	Descripción
a_exec_desc	void**	Puntero al descriptor de ejecución.
a_exec_desc_size	int*	Tamaño del descriptor de ejecución.

Argumentos de salida

Ninguno.

Valor de retorno
Ninguno.

Función get_parameter_order

Finalidad
Recuperar una lista de manejadores de parámetro. El orden de la lista corresponde al orden de los valores de parámetro en el objeto Remote_Operation.

Sintaxis

```
sqlint32 get_parameter_order (int*  a_number_of_params,
                              int** a_param_handle_array)
```

Argumentos de entrada
Ninguno.

Argumentos de salida

Tabla 272. Argumentos de salida para la función de miembro get_parameter_order

Nombre	Tipo de datos	Descripción
a_number_of_params	int*	Número de manejadores de parámetro.
a_param_handle_array	int**	Matriz de manejadores de parámetro.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función cardinality

Finalidad
Recuperar la cardinalidad para el fragmento de consulta.

Sintaxis

```
sqlint32 cardinality (float* a_cardinality)
```

Argumentos de entrada
Ninguno.

Argumentos de salida

Tabla 273. Argumentos de salida para la función de miembro cardinality

Nombre	Tipo de datos	Descripción
a_cardinality	float*	Cardinalidad.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función total_cost

Finalidad

Recuperar el coste total para un fragmento de consulta.

Sintaxis

```
sqlint32 total_cost (float* a_total_cost)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 274. Argumentos de salida para la función de miembro total_cost

Nombre	Tipo de datos	Descripción
a_total_cost	float*	Coste.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función re_exec_cost

Finalidad

Coste para ejecutar un fragmento de consulta otra vez.

Sintaxis

```
sqlint32 re_exec_cost (float* a_re_exec_cost)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 275. Argumentos de salida para la función de miembro re_exec_cost

Nombre	Tipo de datos	Descripción
a_re_exec_cost	float*	Coste.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función first_tuple_cost

Finalidad

Recuperar el coste para buscar y cargar la primera tupla.

Sintaxis

```
sqlint32 first_tuple_cost (float* a_first_tuple_cost)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 276. Argumentos de salida para la función de miembro first_tuple_cost

Nombre	Tipo de datos	Descripción
a_first_tuple_cost	float*	Coste.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función all_costs

Finalidad

Recuperar todos los valores de coste de una sola vez.

Sintaxis

```
sqlint32 all_costs (float* a_total_cost,
                    float* a_first_tuple_cost,
                    float* a_re_exec_cost)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 277. Argumentos de salida para la función de miembro all_costs

Nombre	Tipo de datos	Descripción
a_total_cost	float*	Coste total.
a_first_tuple_cost	float*	Coste de primera tupla.
a_re_exec_cost	float*	Coste de reejecución.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función next

Finalidad
Recuperar un puntero a la siguiente respuesta de una cadena de respuestas.

Sintaxis

```
Reply* next ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero a la siguiente respuesta. El valor es nulo si está al final de la cadena.

Función set_next_reply

Finalidad
Enlazar una nueva respuesta a la respuesta actual.

Uso El reiniciador verifica que añade la respuesta al final de la cadena. De lo contrario, se pueden producir pérdidas de memoria.

Sintaxis

```
void set_next_reply (Reply* a_next_reply)
```

Argumentos de entrada

Tabla 278. Argumentos de entrada para la función de miembro set_next_reply

Nombre	Tipo de datos	Descripción
a_next_reply	Reply*	Respuesta que se debe añadir a una cadena.

Argumentos de salida
Ninguno.

Valor de retorno
Ninguno.

Función get_server

Finalidad
Recuperar un puntero al servidor que es propietario de esta respuesta.

Sintaxis

```
Unfenced_Generic_server*  get_server ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al objeto de servidor.

Consulta relacionada:

- "Clases de petición para la API C++" en la página 156

Clase Request_Exp

Este tema describe la clase Request_Exp y proporciona detalles para cada función de miembro.

Visión general

La clase Request_Exp representa un nodo en un árbol de expresión. Este nodo puede ser una referencia de columna, un valor de constante, un parámetro de sistema principal o un operador.

La clase Request_Exp es una de las clases de petición para la API C++.

Uso El reiniciador no crea nunca instancias de esta clase.

Archivo
sqlqg_request.h

Miembros de datos

Ninguno.

Tipos Request_Exp::kind

Tipo enum

Valores
badkind, column, unbound, constant, oper

Funciones de miembro

La tabla siguiente describe cada función de miembro de la clase Request_Exp. Después de la tabla, se describe cada función más detalladamente.

Tabla 279. Funciones de miembro para la clase Request_Exp

Función de miembro	Descripción
get_kind	Recuperar la clase de nodo de expresión.
get_data_type	Recuperar el tipo de datos que está asociado con la expresión.
get_parent	Recuperar un puntero al nodo de expresión padre para el nodo actual.
get_next_child	Recuperar un puntero al siguiente hijo del mismo padre.
get_handle	Recuperar el manejador para el nodo de expresión.
get_column_name	Recuperar el nombre de columna para un nodo de expresión de columna.
get_column_quantifier_handle	Recuperar el manejador para el cuantificador de columna.
get_value	Recuperar el valor de un nodo de expresión constante.
get_first_child	Recuperar el primer hijo de un nodo de expresión de operador.
get_number_of_children	Recuperar el número de hijos para un nodo de expresión de operador.
get_token	Recuperar la señal para un nodo de expresión de operador.
get_signature	Recuperar la signatura completa para un nodo de expresión de operador.

Función get_kind

Finalidad

Recuperar la clase de nodo de expresión.

Uso sqlint32 get_kind() es una sintaxis adicional para esta función de miembro. Esta sintaxis se utiliza en la expresión Unfenced_Server::get_selectivity() cuando no hay disponible una Petición ni una Respuesta. Esta sintaxis get_kind() no hace distinciones entre las referencias de columna y las referencias desenlazadas.

Sintaxis

```
sqlint32 get_kind (Parsed_Query_Fragment* a_query_fragment,
                   kind*                  a_kind )
```

Argumentos de entrada

Tabla 280. Argumentos de entrada para la función de miembro get_kind

Nombre	Tipo de datos	Descripción
a_query_fragment	Parsed_Query_Fragment*	Petición o Respuesta a la que pertenece esta expresión.

Argumentos de salida

Tabla 281. Argumentos de salida para la función de miembro get_kind

Nombre	Tipo de datos	Descripción
a_kind	kind*	Clase de nodo de expresión.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_data_type

Finalidad

Recuperar el tipo de datos que está asociado con la expresión.

Sintaxis

```
sqlint32 get_data_type (Request_Exp_Type** a_new_type)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 282. Argumentos de salida para la función de miembro get_data_type

Nombre	Tipo de datos	Descripción
a_new_type	Request_Exp_Type**	Puntero al tipo de descriptor.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_parent

Finalidad

Recuperar un puntero al nodo de expresión padre para el nodo actual.

Sintaxis

```
Request_Exp* get_parent ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Puntero al nodo padre. El valor es nulo si se trata de un nodo de nivel superior.

Función get_next_child

Finalidad

Recuperar un puntero al siguiente hijo del mismo padre (es decir, el siguiente sibling del nodo actual).

Sintaxis

```
Request_Exp* get_next_child ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Puntero al siguiente hijo. Es valor es nulo si no hay más siblings.

Función get_handle

Finalidad

Recuperar el manejador para el nodo de expresión.

Sintaxis

```
sqlint32 get_handle (int* a_handle)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 283. Argumentos de salida para la función de miembro get_handle

Nombre	Tipo de datos	Descripción
a_handle	int*	Manejador.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_column_name

Finalidad

Recuperar el nombre de columna para un nodo de expresión de columna.

Uso Este método sólo es válido cuando la clase de expresión es Request_Exp::column.

Sintaxis

```
sqlint32 get_column_name (char** a_col_name,
                          int*   a_name_length)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 284. Argumentos de salida para la función de miembro get_column_name

Nombre	Tipo de datos	Descripción
a_col_name	char**	Puntero al nombre de columna (no terminado por nulo).
a_name_length	int*	Longitud del nombre de columna.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_column_quantifier_handle

Finalidad

Recuperar el manejador para el cuantificador de columna (la tabla o la expresión de tabla).

Uso Este método sólo es válido cuando la clase de expresión es Request_Exp::column.

Sintaxis

```
sqlint32 get_column_quantifier_handle (int* a_quant_handle)
```

Argumentos de entrada
Ninguno.

Argumentos de salida

Tabla 285. Argumentos de salida para la función de miembro get_column_quantifier_handle

Nombre	Tipo de datos	Descripción
a_quant_handle	int*	Manejador de cuantificador.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_value

Finalidad
Recuperar el valor de un nodo de expresión constante.

Uso Este método sólo es válido cuando la clase de expresión es Request_Exp::constant.

Sintaxis

```
sqlint32 get_value (Request_Constant** a_constant_value)
```

Argumentos de entrada
Ninguno.

Argumentos de salida

Tabla 286. Argumentos de salida para la función de miembro get_value

Nombre	Tipo de datos	Descripción
a_constant_value	Request_Constant**	Puntero al valor constante.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_first_child

Finalidad
Recuperar el primer hijo de un nodo de expresión de operador.

Uso Este método sólo es válido cuando la clase de expresión es Request_Exp::oper.

Sintaxis

```
Request_Exp* get_first_child ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al primer hijo. Es valor es nulo si no hay hijos.

Función get_number_of_children

Finalidad
Recuperar el número de hijos para un nodo de expresión de operador.

Uso Este método sólo es válido cuando la clase de expresión es Request_Exp::oper.

Sintaxis

```
int get_number_of_children ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Número de hijos.

Función get_token

Finalidad
Recuperar la señal (el nombre de función) para un nodo de expresión de operador.

Uso Este método sólo es válido cuando la clase de expresión es Request_Exp::oper.

Sintaxis

```
sqlint32 get_token (char** a_operator_token,
                    int*   a_token_length )
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 287. Argumentos de salida para la función de miembro get_token

Nombre	Tipo de datos	Descripción
a_operator_token	char**	Puntero a la señal (no terminada por nulo).
a_token_length	int*	Longitud de la señal.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_signature

Finalidad

Recuperar la signatura completa para un nodo de expresión de operador.

Uso Este método sólo es válido cuando la clase de expresión es Request_Exp::oper.

Sintaxis

```
sqlint32 get_signature (char** a_signature)
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Tabla 288. Argumentos de salida para la función de miembro get_signature

Nombre	Tipo de datos	Descripción
a_signature	char**	Puntero a la signatura terminada por nulo.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Consulta relacionada:

• "Clases de petición para la API C++" en la página 156

Clase Request_Exp_Type

Este tema describe la clase Request_Exp_Type y proporciona detalles para cada función de miembro.

Visión general

La clase Request_Exp_Type representa un descriptor de tipo de datos para los nodos Request_Exp.

La clase Request_Exp_Type es una de las clases de petición para la API C++.

Uso El reiniciador no crea nunca instancias de esta clase.

Archivo
sqlqg_runtime_data_operation.h

Miembros de datos
Ninguno.

Funciones de miembro

La tabla siguiente describe cada función de miembro de la clase Request_Exp_Type. Después de la tabla, se describe cada función más detalladamente.

Tabla 289. Funciones de miembro para la clase Request_Exp_Type

Función de miembro	Descripción
get_for_bit_data	Recuperar el distintivo FOR BIT DATA.
get_null_indicator	Recuperar el indicador de anulable.
get_data_type	Recuperar el tipo de datos.
get_maximum_length	Recuperar la longitud máxima para el tipo.
get_precision	Obtener la precisión para tipos numéricos.
get_scale	Obtener la escala para un tipo numérico.
get_codepage	Recuperar la página de códigos para los tipos de carácter.

Función get_for_bit_data

Finalidad
Recuperar el distintivo FOR BIT DATA para los datos.

Sintaxis

```
unsigned char get_for_bit_data ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Distintivo FOR BIT DATA (Y o N).

Función get_null_indicator

Finalidad
Recuperar el indicador de anulable.

Sintaxis

```
short get_null_indicator ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Distintivo de indicador de nulo, que es SQL_NULLABLE o SQL_NO_NULLS.

Función get_data_type

Finalidad
Recuperar el tipo de datos.

Sintaxis

```
short get_data_type ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Tipo de datos (SQL_TYP_xxx).

Función get_maximum_length

Finalidad
Recuperar la longitud máxima para el tipo.

Sintaxis

```
int get_maximum_length ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Longitud máxima para el tipo.

Función get_precision

Finalidad
Obtener la precisión para tipos numéricos.

Sintaxis

```
unsigned char get_precision ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Precisión numérica.

Función get_scale

Finalidad
Obtener la escala para un tipo numérico.

Sintaxis

```
unsigned char get_scale ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Escala numérica.

Función get_codepage

Finalidad
Recuperar la página de códigos para los tipos de carácter.

Sintaxis

```
unsigned short get_codepage ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Página de códigos.

Consulta relacionada:

- "Clases de petición para la API C++" en la página 156

Clase Request_Constant

Este tema describe la clase Request_Constant y proporciona detalles para cada función de miembro.

Visión general

La clase Request_Constant representa un valor de datos para un nodo Request_Exp con un tipo de Request_Exp::constant.

La clase Request_Constant es una de las clases de petición para la API C++.

Uso El reiniciador no crea nunca instancias de esta clase.

Archivo
sqlqg_runtime_data_operation.h

Miembros de datos
Ninguno.

Funciones de miembro

La tabla siguiente describe cada función de miembro de la clase Request_Constant. Después de la tabla, se describe cada función más detalladamente.

Tabla 290. Funciones de miembro para la clase Request_Constant

Función de miembro	Descripción
get_actual_length	Recuperar la longitud real del valor de datos.
get_data	Recuperar un puntero al valor de datos.
is_data_null	Indicar si el valor de datos es nulo o no nulo.
get_for_bit_data	Recuperar el distintivo FOR BIT DATA.
get_null_indicator	Recuperar el indicador de anulable.

Tabla 290. Funciones de miembro para la clase Request_Constant (continuación)

Función de miembro	Descripción
get_data_type	Recuperar el tipo de datos.
get_maximum_length	Recuperar la longitud máxima para el tipo.
get_precision	Obtener la precisión para tipos numéricos.
get_scale	Obtener la escala para un tipo numérico.
get_codepage	Recuperar la página de códigos para los tipos de carácter.

Función get_actual_length

Finalidad

Recuperar la longitud real del valor de datos.

Sintaxis

```
int get_actual_length ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Longitud de datos real.

Función get_data

Finalidad

Recuperar un puntero al valor de datos.

Sintaxis

```
unsigned char* get_data ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Puntero al valor de datos.

Función is_data_null

Finalidad

Indicar si el valor de datos es nulo o no nulo.

Sintaxis

```
sqlint32 is_data_null ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Indicación de nulo (TRUE o FALSE).

Función get_for_bit_data

Finalidad

Recuperar el distintivo FOR BIT DATA para los datos.

Sintaxis

```
unsigned char get_for_bit_data ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Distintivo FOR BIT DATA (Y o N).

Función get_null_indicator

Finalidad

Recuperar el indicador de anulable.

Sintaxis

```
short get_null_indicator ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno
Distintivo de indicador de nulo, que es SQL_NULLABLE o SQL_NO_NULLS.

Función get_data_type

Finalidad
Recuperar el tipo de datos.

Sintaxis

```
short get_data_type ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Tipo de datos (SQL_TYP_xxx). Consulte el archivo de cabecera sql.h para ver los símbolos de SQL_TYP_xxx.

Función get_maximum_length

Finalidad
Recuperar la longitud máxima para el tipo.

Sintaxis

```
int get_maximum_length ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Longitud máxima para el tipo.

Función get_precision

Finalidad
Obtener la precisión para tipos numéricos.

Sintaxis

```
unsigned char get_precision ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Precisión numérica.

Función get_scale

Finalidad
Obtener la escala para un tipo numérico.

Sintaxis

```
unsigned char get_scale ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Escala numérica.

Función get_codepage

Finalidad
Recuperar la página de códigos para los tipos de carácter.

Sintaxis

```
unsigned short get_codepage ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Página de códigos.

Consulta relacionada:

- "Clases de petición para la API C++" en la página 156

Clase Predicate_List

Este tema describe la clase Predicate_List y proporciona detalles para cada función de miembro.

Visión general

La clase Predicate_List encapsula una lista de predicados y la utiliza el protocolo RRC.

La clase Predicate_List es una de las clases de petición para la API C++.

Uso El reiniciador no crea una instancia de esta clase a no ser que un código para un modelo de coste personalizado invoque el método get_selectivity() para una lista específica de predicados.

Archivo
sqlqg_request.h

Miembros de datos
Ninguno.

Funciones de miembro

La tabla siguiente describe cada función de miembro de la clase Predicate_List. Después de la tabla, se describe cada función más detalladamente.

Tabla 291. Funciones de miembro para la clase Predicate_List

Función de miembro	Descripción
create_empty_predicate_list	Crear una instancia de una lista de predicados vacía.
create_and_copy_predicate_list	Crear una lista de predicados que contenga todos los predicados para una respuesta específica.
get_number_of_predicates	Recuperar el número de predicados de la lista.
get_predicate_handle	Recuperar un manejador de predicado para una posición específica de la lista.
get_predicate	Recuperar una expresión de predicado para un manejador de expresión.
get_number_of_applied_predicates	Obtener el número de predicados que ya se han aplicado.
get_applied_predicate_handle	Recuperar un manejador de predicado aplicado para una posición especificada de la lista.

Tabla 291. Funciones de miembro para la clase Predicate_List (continuación)

Función de miembro	Descripción
get_applied_predicate	Recuperar una expresión de predicado aplicado para un manejador de expresión.
add_predicate	Añadir un predicado a la lista.
add_applied_predicate	Añadir una expresión de predicado a la lista de predicados aplicados.

Función create_empty_predicate_list

Finalidad

Crear una instancia de una lista de predicados vacía.

Sintaxis

```
static sqlint32 create_empty_predicate_list
   (Reply*           a_reply,
    Predicate_List** a_pred_list )
```

Argumentos de entrada

Tabla 292. Argumentos de entrada para la función de miembro create_empty_predicate_list

Nombre	Tipo de datos	Descripción
a_reply	Reply*	Respuesta que es propietaria de los predicados.

Argumentos de salida

Tabla 293. Argumentos de salida para la función de miembro create_empty_predicate_list

Nombre	Tipo de datos	Descripción
a_pred_list	Predicate_List**	Puntero a la nueva lista de predicados.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función create_and_copy_predicate_list

Finalidad

Crear una lista de predicados que contenga todos los predicados para una respuesta específica.

Sintaxis

```
static sqlint32 create_and_copy_predicate_list
   (Reply*           a_reply,
    Predicate_List** a_pred_list)
```

Argumentos de entrada

Tabla 294. Argumentos de entrada para la función de miembro create_and_copy_predicate_list

Nombre	Tipo de datos	Descripción
a_reply	Reply*	Respuesta que es propietaria de los predicados.

Argumentos de salida

Tabla 295. Argumentos de salida para la función de miembro create_and_copy_predicate_list

Nombre	Tipo de datos	Descripción
a_pred_list	Predicate_List**	Puntero a la nueva lista de predicados.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_number_of_predicates

Finalidad

Recuperar el número de predicados de la lista.

Sintaxis

```
int get_number_of_predicates ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Número de predicados.

Función get_predicate_handle

Finalidad

Recuperar un manejador de predicado para una posición específica de la lista.

Sintaxis

```
sqlint32 get_predicate_handle (int  a_pred_pos,
                               int* a_pred_handle)
```

Argumentos de entrada

Tabla 296. Argumentos de entrada para la función de miembro get_predicate_handle

Nombre	Tipo de datos	Descripción
a_pred_pos	int	Posición de un predicado en la lista (empezando en 1).

Argumentos de salida

Tabla 297. Argumentos de salida para la función de miembro get_predicate_handle

Nombre	Tipo de datos	Descripción
a_pred_handle	int*	Manejador de expresión de predicado.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_predicate

Finalidad

Recuperar una expresión de predicado para un manejador de expresión.

Sintaxis

```
sqlint32 get_predicate (int           a_pred_handle,
                        Request_Exp** a_pred_exp)
```

Argumentos de entrada

Tabla 298. Argumentos de entrada para la función de miembro get_predicate

Nombre	Tipo de datos	Descripción
a_pred_handle	int	Manejador de expresión de predicado.

Argumentos de salida

Tabla 299. Argumentos de salida para la función de miembro get_predicate

Nombre	Tipo de datos	Descripción
a_pred_exp	Request_Exp**	Puntero a un nodo de expresión.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_number_of_applied_predicates

Finalidad

Obtener el número de predicados que ya se han aplicado.

Sintaxis

```
int get_number_of_applied_predicates ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Número de predicados aplicados.

Función get_applied_predicate_handle

Finalidad

Recuperar un manejador de predicado aplicado para una posición especificada de la lista.

Sintaxis

```
sqlint32 get_applied_predicate_handle (int  a_applied_pred_pos,
                                       int* a_applied_pred_handle)
```

Argumentos de entrada

Tabla 300. Argumentos de entrada para la función de miembro get_applied_predicate_handle

Nombre	Tipo de datos	Descripción
a_applied_pred_pos	int	Posición del predicado en la lista (empezando en 1).

Argumentos de salida

Tabla 301. Argumentos de salida para la función de miembro get_applied_predicate_handle

Nombre	Tipo de datos	Descripción
a_applied_pred_handle	int*	Manejador de expresión de predicado aplicado.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_applied_predicate

Finalidad

Recuperar una expresión de predicado aplicado para un manejador de expresión.

Sintaxis

```
sqlint32 get_applied_predicate (int          a_applied_pred_handle,
                                Request_Exp** a_applied_pred_exp)
```

Argumentos de entrada

Tabla 302. Argumentos de entrada para la función de miembro get_applied_predicate

Nombre	Tipo de datos	Descripción
a_applied_pred_handle	int	Manejador de expresión de predicado aplicado.

Argumentos de salida

Tabla 303. Argumentos de salida para la función de miembro get_applied_predicate

Nombre	Tipo de datos	Descripción
a_applied_pred_exp	Request_Exp**	Puntero a un nodo de expresión.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función add_predicate

Finalidad

Añadir un predicado a la lista.

Sintaxis

```
sqlint32 add_predicate (int a_pred_handle)
```

Argumentos de entrada

Tabla 304. Argumentos de entrada para la función de miembro add_predicate

Nombre	Tipo de datos	Descripción
a_pred_handle	int	Manejador para la expresión de predicado.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función add_applied_predicate

Finalidad

Añadir una expresión de predicado a la lista de predicados aplicados.

Sintaxis

```
sqlint32 add_applied_predicate (int a_applied_pred_handle)
```

Argumentos de entrada

Tabla 305. Argumentos de entrada para la función de miembro add_applied_predicate

Nombre	Tipo de datos	Descripción
a_applied_pred_handle	int	Manejador para la expresión de predicado aplicado.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Consulta relacionada:

• "Clases de petición para la API C++" en la página 156

Clases de datos para la API C++

La tabla siguiente describe cada clase de datos para la API C++.

Tabla 306. Clases de datos

Nombre de clase	Descripción
Runtime_Data_Desc	Clase que encapsula la descripción de un elemento de datos.
Runtime_Data_Desc_List	Una lista de Runtime_Data_Desc.
Runtime_Data	Clase que encapsula un valor de datos que se pasa entre el reiniciador y la interfaz de pasarela de consulta. Este valor de datos puede ser datos de columna que están en un resultado o en un parámetro de consulta.
Runtime_Data_List	Una lista de punteros a objetos Runtime_Data.

Consulta relacionada:

- "Clase Runtime_Data_Desc" en la página 205
- "Clase Runtime_Data_Desc_List" en la página 210
- "Clase Runtime_Data" en la página 213
- "Clase Runtime_Data_List" en la página 222

Clase Runtime_Data_Desc

Este tema describe la clase Runtime_Data_Desc y proporciona detalles para cada función de miembro.

Visión general

La clase Runtime_Data_Desc encapsula la descripción de un elemento de datos.

La clase Runtime_Data_Desc es una de las clases de datos para la API C++.

Uso La pasarela de consulta de DB2 crea una instancia de esta clase y el reiniciador no crea nunca una subclase de esta clase. El reiniciador puede crear una instancia de esta clase durante el proceso de preparación para una operación de paso a través.

Archivo
sqlqg_runtime_data_operation.h

Miembros de datos
Ninguno.

Constructores y funciones de miembro

Las tablas siguientes describen el constructor y las funciones de miembro de la clase Runtime_Data_Desc. El constructor y las funciones se describen más detalladamente después de las tablas.

Tabla 307. Constructores para la clase Runtime_Data_Desc

Constructor	Descripción
Runtime_Data_Desc	Construir una instancia de Runtime_Data_Desc.

Tabla 308. Funciones de miembro para la clase Runtime_Data_Desc

Función de miembro	Descripción
get_for_bit_data	Recuperar el distintivo FOR BIT DATA para los datos.
get_null_indicator	Recuperar el indicador de anulable.
get_data_type	Recuperar el tipo de datos.
get_maximum_length	Recuperar la longitud máxima para el tipo.
get_precision	Obtener la precisión para tipos numéricos.
get_scale	Obtener la escala para un tipo numérico.
get_codepage	Recuperar la página de códigos para los tipos de carácter.

Constructor Runtime_Data_Desc

Finalidad

Construir una instancia de Runtime_Data_Desc. Este constructor se utiliza cuando se crea una instancia de la clase Runtime_Data_Desc en respuesta a una operación de preparación o descripción para una sesión de paso a través.

Sintaxis

```
Runtime_Data_Desc (sqlint32*      a_rc,
                   short          a_type,
                   int            a_max_length,
                   short          a_codepage,
                   short          a_null_ind,
                   unsigned char  a_data_precision=0,
                   unsigned char  a_data_scale=0,
                   sqluint8*      a_data_name=NULL,
                   short          a_name_length=0,
                   short          a_remote_type=0)
```

Argumentos de entrada

Tabla 309. Argumentos de entrada para el constructor Runtime_Data_Desc

Nombre	Tipo de datos	Descripción
a_type	short	Tipo de datos. Utilice los valores de SQL_TYP_xxx que están definidos en el archivo de cabecera sql.h.
a_max_length	int	Longitud máxima de los datos.
a_codepage	short	Página de códigos para los datos de tipo carácter.
a_null_ind	short	Indicador de nulo.
a_data_precision	unsigned char	Precisión para datos numéricos.
a_data_scale	unsigned char	Escala para un tipo de datos numérico.
a_data_name	sqluint8*	Nombre del elemento de datos.
a_name_length	short	Longitud del nombre.
a_remote_type	short	Código de tipo remoto.

Argumentos de salida

Tabla 310. Argumentos de salida para el constructor Runtime_Data_Desc

Nombre	Tipo de datos	Descripción
a_rc	sqlint32*	Puntero al valor de código de retorno; 0 indica que la operación se ha realizado satisfactoriamente.

Función get_for_bit_data

Finalidad

Recuperar el distintivo FOR BIT DATA para los datos.

Sintaxis

```
unsigned char get_for_bit_data ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Distintivo FOR BIT DATA (Y o N).

Función get_null_indicator

Finalidad
Recuperar el indicador de anulable.

Sintaxis

```
short get_null_indicator ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Distintivo de indicador de nulo, que es SQL_NULLABLE o SQL_NO_NULLS.

Función get_data_type

Finalidad
Recuperar el tipo de datos.

Sintaxis

```
short get_data_type ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Tipo de datos (SQL_TYP_xxx). Consulte el archivo de cabecera sql.h para ver los símbolos de SQL_TYP_xxx.

Función get_maximum_length

Finalidad
Recuperar la longitud máxima para el tipo.

Sintaxis

```
int get_maximum_length ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Longitud máxima para el tipo.

Función get_precision

Finalidad
Obtener la precisión para tipos numéricos.

Sintaxis

```
unsigned char get_precision ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Precisión numérica.

Función get_scale

Finalidad
Obtener la escala para un tipo numérico.

Sintaxis

```
unsigned char get_scale ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Escala numérica.

Función get_codepage

Finalidad
Recuperar la página de códigos para los tipos de carácter.

Sintaxis

```
unsigned short get_codepage ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Página de códigos.

Consulta relacionada:

- "Clases de datos para la API C++" en la página 205

Clase Runtime_Data_Desc_List

Este tema describe la clase Runtime_Data_Desc_List y proporciona detalles para cada función de miembro.

Visión general

La clase Runtime_Data_Desc_List proporciona una lista de la clase Runtime_Data_Desc.

La clase Runtime_Data_Desc_List es una de las clases de datos para la API C++.

Uso La pasarela de consulta de DB2 crea una instancia de esta clase pero el reiniciador no crea nunca una subclase de esta clase.

Archivo
sqlqg_runtime_data_operation.h

Miembros de datos
Ninguno.

Funciones de miembro

La tabla siguiente describe cada función de miembro de la clase Runtime_Data_Desc_List. Después de la tabla, se describe cada función más detalladamente.

Tabla 311. Funciones de miembro para la clase Runtime_Data_Desc_List

Función de miembro	Descripción
get_number_of_values	Recuperar el número de valores.
set_number_of_values	Establecer el número de valores de la lista.
get_ith_value	Recuperar el descriptor de datos que está en la posición número *i*.

Tabla 311. Funciones de miembro para la clase Runtime_Data_Desc_List (continuación)

Función de miembro	Descripción
set_ith_value	Almacenar un puntero de Runtime_Data_Desc en la posición número *i* de la lista.
operator[]	Recuperar la entrada que está en la posición número *i* de la lista utilizando la anotación de subíndice.

Función get_number_of_values

Finalidad

Recuperar el número de valores.

Sintaxis

```
int get_number_of_values ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Número de valores.

Función set_number_of_values

Finalidad

Establecer el número de valores de la lista. Esta función de miembro puede vaciar la lista estableciendo el número en 0. Esta función de miembro también puede alargar o acortar la lista.

Sintaxis

```
sqlint32 set_number_of_values (int a_count)
```

Argumentos de entrada

Tabla 312. Argumentos de entrada para la función de miembro set_number_of_values

Nombre	Tipo de datos	Descripción
a_count	int	Número de valores.

Argumentos de salida

Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_ith_value

Finalidad
Recuperar el descriptor de datos que está en la posición número *i*.

Sintaxis

```
Runtime_Data_Desc* get_ith_value (int a_index)
```

Argumentos de entrada

Tabla 313. Argumentos de entrada para la función de miembro get_ith_value

Nombre	Tipo de datos	Descripción
a_index	int	Indexar en la lista (empezando en 0).

Argumentos de salida
Ninguno.

Valor de retorno
Puntero al valor que está en la posición número *i* (o nulo).

Función set_ith_value

Finalidad
Almacenar un puntero Runtime_Data_Desc en la posición número *i* de la lista.

Sintaxis

```
sqlint32 set_ith_value (Runtime_Data_Desc* a_desc,
                        int                a_index)
```

Argumentos de entrada

Tabla 314. Argumentos de entrada para la función de miembro set_ith_value

Nombre	Tipo de datos	Descripción
a_desc	Runtime_Data_Desc*	Puntero de descriptor.
a_index	int	Índice (empezando en 0).

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función operator[]

Finalidad
Recuperar la entrada que está en la posición número *i* de la lista utilizando la anotación de subíndice.

Sintaxis

```
Runtime_Data_Desc* operator[] (int a_index)
```

Argumentos de entrada

Tabla 315. Argumentos de entrada para la función de miembro operator []

Nombre	Tipo de datos	Descripción
a_index	int	Indexar en la lista (empezando en 0).

Argumentos de salida
Ninguno.

Valor de retorno
La entrada que está en la posición número *i*.

Consulta relacionada:

- "Clases de datos para la API C++" en la página 205

Clase Runtime_Data

Este tema describe la clase Runtime_Data y proporciona detalles para cada función de miembro.

Visión general

La clase Runtime_Data encapsula un valor de datos que se pasa entre el reiniciador y la interfaz de pasarela de consulta. Este valor de datos pueden ser datos de columna que están en un resultado o en un parámetro de consulta.

La clase Runtime_Data es una de las clases de datos para la API C++.

Uso La pasarela de consulta de DB2 crea una instancia de esta clase y el reiniciador no crea nunca una subclase de esta clase.

Archivo
sqlqg_runtime_data_operation.h

Miembros de datos
Ninguno.

Funciones de miembro

La tabla siguiente describe cada función de miembro de la clase Runtime_Data. Después de la tabla, se describe cada función más detalladamente.

Tabla 316. Funciones de miembro para la clase Runtime_Data

Función de miembro	Descripción
get_actual_length	Recuperar la longitud real del valor de datos.
set_actual_length	Establecer la longitud real del valor de datos.
get_data	Recuperar un puntero al valor de datos.
set_data	Copiar información en el valor de datos.
is_data_null	Indicar si el valor de datos es nulo o no lo es.
set_data_null	Marcar el valor de datos como nulo.
clear_null_indicator	Restablecer el indicador de nulo para el valor de datos.
set_friendly_div_by_0	Indicar que un valor es nulo porque se ha producido un error de división por cero.
set_friendly_exception	Indicar que un valor es nulo debido a una excepción numérica.
is_data_nullable	Recuperar una indicación de que el valor de datos es anulable o no lo es.
is_semantic_null	Devolver una indicación de que el valor es semánticamente nulo.
check_friendly_div_by_0	Determinar si la razón de una indicación de nulo es un error de división por cero.
check_friendly_exception	Determinar si la razón de una indicación de nulo es una excepción numérica.
get_for_bit_data	Recuperar el distintivo FOR BIT DATA para los datos.
get_null_indicator	Recuperar el indicador de anulable.
get_data_type	Recuperar el tipo de datos.
get_maximum_length	Recuperar la longitud máxima.

Tabla 316. Funciones de miembro para la clase Runtime_Data (continuación)

Función de miembro	Descripción
get_precision	Obtener la precisión para tipos numéricos.
get_scale	Obtener la escala para un tipo numérico.
get_codepage	Recuperar la página de códigos para los tipos de carácter.

Función get_actual_length

Finalidad

Recuperar la longitud real del valor de datos.

Sintaxis

```
int get_actual_length ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Longitud de datos real.

Función set_actual_length

Finalidad

Establecer la longitud real del valor de datos.

Sintaxis

```
void set_actual_length (int a_length)
```

Argumentos de entrada

Tabla 317. Argumentos de entrada para la función de miembro set_actual_length

Nombre	Tipo de datos	Descripción
a_length	int	Longitud.

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función get_data

Finalidad

Recuperar un puntero al valor de datos.

Sintaxis

```
unsigned char* get_data ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Puntero al valor de datos.

Función set_data

Finalidad

Copiar información en el valor de datos.

Sintaxis

```
sqlint32 set_data (unsigned char* a_data_ptr,
                   int            a_copy_len)
```

Argumentos de entrada

Tabla 318. Argumentos de entrada para la función de miembro set_data

Nombre	Tipo de datos	Descripción
a_data_ptr	unsigned char*	Valor de datos.
a_copy_len	int	Longitud del valor.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función is_data_null

Finalidad

Indicar si el valor de datos es nulo o no lo es.

Sintaxis

```
sqlint32 is_data_null ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Indicación de nulo.

Función set_data_null

Finalidad
Marcar el valor de datos como nulo.

Sintaxis

```
sqlint32 set_data_null ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función clear_null_indicator

Finalidad
Restablecer el indicador de nulo para el valor de datos.

Sintaxis

```
sqlint32 clear_null_indicator ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función set_friendly_div_by_0

Finalidad

Indicar que un valor es nulo porque se ha producido un error de división por cero.

Sintaxis

```
sqlint32 set_friendly_div_by_0 ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función set_friendly_exception

Finalidad

Indicar que un valor es nulo debido a una excepción numérica.

Sintaxis

```
sqlint32 set_friendly_exception ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función is_data_nullable

Finalidad

Recuperar una indicación de que el valor de datos es anulable o no lo es.

Sintaxis

```
sqlint32 is_data_nullable ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Indicación de anulable. El valor es TRUE si es anulable.

Función is_semantic_null

Finalidad
Devolver una indicación de que el valor es semánticamente nulo.

Sintaxis

```
sqlint32 is_semantic_null ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Indicación de anulable. El valor es TRUE si es anulable.

Función check_friendly_div_by_0

Finalidad
Determinar si la razón de una indicación de nulo es un error de división por cero.

Sintaxis

```
sqlint32 check_friendly_div_by_0
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Indicación de nulo. El valor es TRUE si se ha producido una condición de división por cero.

Función check_friendly_exception

Finalidad
Determinar si la razón de una indicación de nulo es una excepción numérica.

Sintaxis

```
sqlint32 check_friendly_exception ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Indicación de nulo. El valor es TRUE si se ha producido una excepción.

Función get_for_bit_data

Finalidad
Recuperar el distintivo FOR BIT DATA para los datos.

Sintaxis

```
unsigned char get_for_bit_data ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Distintivo FOR BIT DATA (Y o N).

Función get_null_indicator

Finalidad
Recuperar el indicador de anulable.

Sintaxis

```
short get_null_indicator ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Distintivo de indicador de nulo, que es SQL_NULLABLE o SQL_NO_NULLS.

Función get_data_type

Finalidad

Recuperar el tipo de datos.

Sintaxis

```
short get_data_type ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Tipo de datos (SQL_TYP_xxx). Consulte el archivo de cabecera sql.h para ver los símbolos de SQL_TYP_xxx.

Función get_maximum_length

Finalidad

Recuperar la longitud máxima para el tipo.

Sintaxis

```
int get_maximum_length ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Longitud máxima para el tipo.

Función get_precision

Finalidad

Obtener la precisión para tipos numéricos.

Sintaxis

```
unsigned char get_precision ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno
Precisión numérica.

Función get_scale

Finalidad
Obtener la escala para un tipo numérico.

Sintaxis

```
unsigned char get_scale ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Escala numérica.

Función get_codepage

Finalidad
Recuperar la página de códigos para los tipos de carácter.

Sintaxis

```
unsigned short get_codepage ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Página de códigos.

Consulta relacionada:

- "Clases de datos para la API C++" en la página 205

Clase Runtime_Data_List

Este tema describe la clase Runtime_Data_List y proporciona detalles para cada función de miembro.

Visión general

La clase Runtime_Data_List proporciona una lista de punteros a objetos Runtime_Data.

La clase Runtime_Data_List es una de las clases de datos para la API C++.

Uso La pasarela de consulta de DB2 crea una instancia de esta clase y el reiniciador no crea nunca una subclase de esta clase.

Archivo
sqlqg_runtime_data_operation.h

Miembros de datos
Ninguno.

Funciones de miembro

La tabla siguiente describe cada función de miembro de la clase Runtime_Data_List. Después de la tabla, se describe cada función más detalladamente.

Tabla 319. Funciones de miembro para la clase Runtime_Data_List

Función de miembro	Descripción
get_number_of_values	Recuperar el número de valores.
get_ith_value	Recuperar el descriptor de datos que está en la posición número *i*.
operator[]	Recuperar la entrada que está en la posición número *i* de la lista utilizando la anotación de subíndice.

Función get_number_of_values

Finalidad
Recuperar el número de valores.

Sintaxis

```
int get_number_of_values ()
```

Argumentos de entrada
Ninguno.

Argumentos de salida
Ninguno.

Valor de retorno
Número de valores.

Función get_ith_value

Finalidad

Recuperar el valor de datos que está en la posición número *i*.

Sintaxis

```
Runtime_Data* get_ith_value (int a_index)
```

Argumentos de entrada

Tabla 320. Argumentos de entrada para la función de miembro get_ith_value

Nombre	Tipo de datos	Descripción
a_index	int	Indexar en la lista (empezando en 0).

Argumentos de salida

Ninguno.

Valor de retorno

Puntero al valor que está en la posición número *i* (o nulo).

Función operator[]

Finalidad

Recuperar la entrada que está en la posición número *i* de la lista utilizando la anotación de subíndice.

Sintaxis

```
Runtime_Data* operator[] (int a_index)
```

Argumentos de entrada

Tabla 321. Argumentos de entrada para la función de miembro operator[]

Nombre	Tipo de datos	Descripción
a_index	int	Indexar en la lista (empezando en 0).

Argumentos de salida

Ninguno.

Valor de retorno

La entrada que está en la posición número *i*.

Consulta relacionada:

• "Clases de datos para la API C++" en la página 205

Clase Wrapper_Utilities

Este tema describe la clase Wrapper_Utilities y proporciona detalles para cada función de miembro.

Visión general

La clase Wrapper_Utilities es un contenedor para varias funciones estáticas de programa de utilidad. No cree instancias ni subclases de la clase Wrapper_Utilities.

La clase Wrapper_Utilities es una clase de programas de utilidad para la API C++.

Archivo
sqlqg_utils.h

Miembros de datos
Ninguno.

Funciones de miembro

La tabla siguiente describe cada función de miembro de la clase Wrapper_Utilities. Después de la tabla, se describe cada función más detalladamente.

Tabla 322. Funciones de miembro para la clase Wrapper_Utilities

Función de miembro	Descripción
convert_to_upper	Convertir una serie de caracteres a mayúsculas utilizando la página de códigos especificada.
convert_to_lower	Convertir una serie de caracteres a minúsculas utilizando la página de códigos especificada.
report_error	Generar un error para informar al usuario.
report_warning	Generar un aviso para informar al usuario.
allocate	Asignar un bloque de memoria.
deallocate	Liberar un bloque de memoria que se ha asignado con allocate().
get_sb_DB_codepage	Devolver la página de códigos de un solo byte para la base de datos actual.
get_db_DB_codepage	Devolver la página de códigos de doble byte para la base de datos actual.
string_to_tokens	Explorar una serie de caracteres y dividir la serie en señales sucesivas.

Tabla 322. Funciones de miembro para la clase Wrapper_Utilities (continuación)

Función de miembro	Descripción
get_db2_install_path	Devolver una serie de caracteres terminada por nulo que muestre el nombre de vía de acceso absoluta del directorio de instalación de DB2 Information Integrator.
get_db2_instance_path	Devolver una serie de caracteres terminada por nulo que muestre el nombre de vía de acceso absoluta de la instancia de DB2 Information Integrator.
trace_data	Grabar un bloque de información en el recurso de rastreo de DB2.

Función convert_to_upper

Finalidad

Convertir una serie de caracteres a mayúsculas utilizando la página de códigos especificada.

Uso Utilizar la función get_sb_DB_codepage() o get_db_DB_codepage() para obtener la página de códigos de base de datos actual.

Sintaxis

```
int convert_to_upper (char*        a_string,
                      size_t       a_length,
                      unsigned int a_codepage)
```

Argumentos de entrada

Tabla 323. Argumentos de entrada para la función de miembro convert_to_upper

Nombre	Tipo de datos	Descripción
a_string	char*	Serie a convertir (no terminada por nulo).
a_length	size_t	Longitud de la serie.
a_codepage	unsigned int	Página de códigos para la conversión.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. Un valor de -1 indica que la página de códigos no está definida y que la serie que se debe convertir contiene caracteres ampliados. Se saltan estos caracteres ampliados.

Función convert_to_lower

Finalidad

Convertir una serie de caracteres a minúsculas utilizando la página de códigos especificada.

Uso Utilizar la función get_sb_DB_codepage() o get_db_DB_codepage() para obtener la página de códigos de base de datos actual.

Sintaxis

```
int convert_to_lower (char*        a_string,
                      size_t       a_length,
                      unsigned int a_codepage)
```

Argumentos de entrada

Tabla 324. Argumentos de entrada para la función de miembro convert_to_lower

Nombre	Tipo de datos	Descripción
a_string	char*	Serie a convertir (no terminada por nulo).
a_length	size_t	Longitud de la serie.
a_codepage	unsigned int	Página de códigos para la conversión.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente. Un valor de -1 indica que la página de códigos no está definida y que la serie que se debe convertir contiene caracteres ampliados. Se saltan estos caracteres ampliados.

Función report_error

Finalidad

Generar un error para informar al usuario.

Sintaxis

```
sqlint32 report_error (const char* a_function_name,
                       int         a_sql_code,
                       int         a_number_of_tokens,
                         ...)
```

Argumentos de entrada

Tabla 325. Argumentos de entrada para la función de miembro report_error

Nombre	Tipo de datos	Descripción
a_function_name	const char*	Serie de caracteres terminada por nulo que identifica la función que genera el error. La serie no puede tener más de cinco caracteres. Este valor de serie es accesible para el programa cliente a través del campo sqlerrp de SQLCA y aparece en letras mayúsculas con un prefijo de SQL.
a_sql_code	int	El código SQL predefinido para el error. Consulte el archivo sqlcodes.h.
a_number_of_tokens	int	El número de señales de sustitución para el mensaje.
...	(int, const char*)	Un par de valores para cada señal. Cada par consta de una longitud de entero y de un puntero a la serie de caracteres que no está terminada por nulo.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Esta función devuelve al llamante el código de error indicado y el llamante devuelve este código de error a DB2.

Función report_warning

Finalidad

Generar un aviso para informar al usuario.

Sintaxis

```
sqlint32 report_warning (const char*  a_function_name,
                         int          a_warning_index,
                         char         a_warning_flag,
                         int          a_number_of_tokens,
                           ...)
```

Argumentos de entrada

Tabla 326. Argumentos de entrada para la función de miembro report_warning

Nombre	Tipo de datos	Descripción
a_function_name	const char*	Serie de caracteres terminada por nulo que identifica la función que genera el aviso. La serie no puede tener más de cinco caracteres. Este valor de serie es accesible para el programa cliente a través del campo sqlerrp de SQLCA y aparece en letras mayúsculas con un prefijo de SQL.
a_warning_index	int	Valor que identifica el tipo de aviso. Utilice las constantes de SQL_WARN_xxx que están definidas en el archivo de cabecera sql.h.
a_warning_flag	char	Distintivo de aviso. Utilice la constante de SQL_WARNING que está definida en el archivo de cabecera sql.h.
a_number_of_tokens	int	Número de señales (pares de serie y longitud) que se deben proporcionar a través del campo sqlerrm de SQLCA y que es accesible para un programa cliente.
...	(int, const char*)	Un par de valores para cada señal. Cada par consta de una longitud de entero y de un puntero a la serie de caracteres que no está terminada por nulo.

Argumentos de salida

Ninguno.

Valor de retorno

Un valor de 0 indica que el aviso se ha generado satisfactoriamente. Un valor distinto de cero indica que se ha producido un problema al generarse el error. Se debe devolver un código de retorno distinto de cero a DB2.

Función allocate

Finalidad

Asignar un bloque de memoria.

Sintaxis

```
int allocate (size_t a_size,
              void** a_block)
```

Argumentos de entrada

Tabla 327. Argumentos de entrada para la función de miembro allocate

Nombre	Tipo de datos	Descripción
a_size	size_t	Tamaño de un bloque que se debe asignar.

Argumentos de salida

Tabla 328. Argumentos de salida para la función de miembro allocate

Nombre	Tipo de datos	Descripción
a_block	void**	Puntero al bloque asignado.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función deallocate

Finalidad

Liberar un bloque de memoria que se ha asignado con allocate().

Sintaxis

```
void deallocate (void* a_block)
```

Argumentos de entrada

Tabla 329. Argumentos de entrada para la función de miembro deallocate

Nombre	Tipo de datos	Descripción
a_block	void*	Bloque de memoria que se debe liberar.

Argumentos de salida

Ninguno.

Valor de retorno

Ninguno.

Función get_sb_DB_codepage

Finalidad

Devolver la página de códigos de un solo byte para la base de datos actual.

Sintaxis

```
int get_sb_DB_codepage ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Página de códigos.

Función get_db_DB_codepage

Finalidad

Devolver la página de códigos de doble byte para la base de datos actual.

Sintaxis

```
int get_db_DB_codepage ()
```

Argumentos de entrada

Ninguno.

Argumentos de salida

Ninguno.

Valor de retorno

Página de códigos.

Función string_to_tokens

Finalidad

Explorar una serie de caracteres y dividir la serie en señales sucesivas. Esta función de miembro es una alternativa segura a las funciones strtok() y strtok_r().

Sintaxis

```
char* string_to_tokens (char*       a_string,
                        const char* a_sep,
                        char**      a_last)
```

Argumentos de entrada

Tabla 330. Argumentos de entrada para la función de miembro string_to_tokens

Nombre	Tipo de datos	Descripción
a_string	char*	Serie que se debe explorar.
a_sep	const char*	Serie a utilizar como separador entre las señales.
a_last	char*	Valor que se utiliza para mantener el estado entre las llamadas a string_to_tokens.

Argumentos de salida

Ninguno.

Valor de retorno

Puntero a la siguiente señal de la serie o nulo.

Función get_db2_install_path

Finalidad

Devolver una serie de caracteres terminada por nulo que muestre el nombre de vía de acceso absoluta del directorio de instalación de DB2 Information Integrator.

Sintaxis

```
sqlint32 get_db2_install_path (char*    a_path,
                               sqlint32 a_path_size)
```

Argumentos de entrada

Tabla 331. Argumentos de entrada para la función de miembro get_db2_install_path

Nombre	Tipo de datos	Descripción
a_path	char*	Puntero al almacenamiento intermedio que contiene el nombre de vía de acceso.
a_path_size	sqlint32	Longitud del almacenamiento intermedio (incluido el espacio para un terminador nulo).

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función get_db2_instance_path

Finalidad

Devolver una serie de caracteres terminada por nulo que muestre el nombre de vía de acceso absoluta de la instancia de DB2 Information Integrator.

Sintaxis

```
sqlint32 get_db2_instance_path (char*    a_path,
                                sqlint32 a_path_size)
```

Argumentos de entrada

Tabla 332. Argumentos de entrada para la función de miembro get_db2_instance_path

Nombre	Tipo de datos	Descripción
a_path	char*	Puntero al almacenamiento intermedio que contiene el nombre de vía de acceso.
a_path_size	sqlint32	Longitud del almacenamiento intermedio (incluido el espacio para un terminador nulo).

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Función trace_data

Finalidad

Grabar un bloque de información en el recurso de rastreo de DB2.

Uso Para obtener más información sobre el recurso de rastreo de DB2, consulte la publicación *Consulta de mandatos de DB2*.

Sintaxis

```
void trace_data (int    a_probe,
                 void*  a_data,
                 int    a_data_size)
```

Argumentos de entrada

Tabla 333. Argumentos de entrada para la función de miembro trace_data

Nombre	Tipo de datos	Descripción
a_probe	int	Número que identifica el punto de rastreo.
a_data	void*	Puntero a los datos que se deben rastrear.
a_data_size	int	Longitud de los datos que se deben rastrear.

Argumentos de salida

Ninguno.

Valor de retorno

Código de retorno. Un valor de 0 indica que la operación se ha realizado satisfactoriamente.

Consulta relacionada:

- "Clases de catálogo para la API C++" en la página 1
- "Clases de datos para la API C++" en la página 205

Documentación técnica de DB2 Information Integrator

Los siguientes temas describen cómo:

- Acceder a información sobre manuales y del release, incluyendo la impresión y solicitud de manuales
- Acceder a temas utilizando el Centro de información de DB2 Information Integrator o el CD de documentación en HTML de DB2

Acceso a la información sobre manuales y del release

La información técnica de DB2 Information Integrator está disponible en los siguientes formatos:

- Manuales (PDF e impresos). En la biblioteca de DB2 Information Integrator existe una descripción de cada manual desde el IBM Publications Center en el sitio www.ibm.com/shop/publications/order.
- Un centro de información (formato HTML).
- Ayuda para las herramientas de bases de datos de DB2 (formato HTML).

Manuales de DB2 Information Integrator

El CD de documentación en PDF de DB2 Information Integrator contiene archivos PDF de los manuales en la biblioteca de DB2 Information Integrator y en la biblioteca de DB2 Universal Database. La estructura del CD de documentación en PDF de DB2 Information Integrator es:

- En sistemas operativos Windows: *x*:\doc*%L*
- En sistemas operativos UNIX: */cdrom*/doc/*%L*/

donde:

- *x* representa la letra de la unidad de CD-ROM de Windows
- *cdrom* se refiere al punto de montaje de UNIX del CD-ROM
- *%L* es el entorno nacional de la documentación que desea utilizar, por ejemplo, en_US

Idioma	Entorno nacional	Identif.
Alemán	de_DE	g
Árabe	ar_AA	w
Búlgaro	bg_BG	u
Checo	cs_CZ	x
Chino simplificado	zh_CN	c
Chino tradicional	zh_TW	t
Coreano	ko_KR	k
Croata	hr_HR	9

Danés	da_DK	d
Eslovaco	sk_SK	7
Esloveno	sl_SI	l
Español	es_ES	z
Finlandés	fi_FI	y
Francés	fr_FR	f
Griego	el_GR	a
Holandés	nl_NL	q
Húngaro	hu_HU	h
Inglés	en_US	e
Italiano	it_IT	i
Japonés	ja_JP	j
Noruego	no_NO	n
Polaco	pl_PL	p
Portugués de Brasil	pt_BR	b
Portugués	pt_PT	v
Rumano	ro_RO	8
Ruso	ru_RU	r
Sueco	sv_SE	s
Turco	tr_TR	m

El carácter en la sexta posición de un nombre de archivo PDF indica la versión de idioma de un manual (consulte la tabla siguiente). Por ejemplo, el nombre de archivo `iiyige80` identifica la versión inglesa del manual *IBM DB2 Information Integrator Guía de instalación* y el nombre de archivo `iiyigg80` identifica la versión alemana del mismo manual.

Los manuales de la tabla siguiente están disponibles para DB2 Information Integrator.

Tabla 334. Documentación de DB2 Information Integrator

Nombre	Número de documento	Categoría de instalación	Nombre de archivo PDF
IBM DB2 Information Integrator Guía de soluciones	SC10-3988	getting_started	iiyisx80
IBM DB2 Information Integrator Guía de instalación	GC10-3987	getting_started	iiyigx80
IBM DB2 Information Integrator Guía de migración	SC10-3993	getting_started	iiymgx80

Tabla 334. Documentación de DB2 Information Integrator (continuación)

Nombre	Número de documento	Categoría de instalación	Nombre de archivo PDF
IBM DB2 Information Integrator Guía de sistemas federados	SC10-3990	admin	iiyfpx80
IBM DB2 Information Integrator Guía de configuración de fuentes de datos	Sólo disponible en línea	optional	iiylsx80
IBM DB2 Information Integrator Guía del desarrollador	SC10-3991	ad	iiyfsx80

Impresión de manuales desde archivos PDF

Puede imprimir manuales de DB2 Information Integrator desde los archivos PDF del CD de documentación en PDF de DB2 Information Integrator. Puede utilizar Adobe Acrobat Reader para imprimir el manual entero, un rango de páginas o páginas específicas.

Prerrequisitos:

Asegúrese de tener Adobe Acrobat Reader. Está disponible en el sitio Web de Adobe en www.adobe.com.

Procedimiento:

Para imprimir un manual de DB2 Information Integrator desde un archivo PDF:

1. Inserte el CD de documentación en PDF de DB2 Information Integrator. En sistemas operativos UNIX, monte el CD.
2. Inicie Adobe Acrobat Reader.
3. Abra el archivo PDF desde una de las ubicaciones siguientes:
   - En sistemas operativos Windows: *x*:\doc\%*L*
   - En sistemas operativos UNIX: /*cdrom*/doc/%*L*/

   donde:
   - *x* representa la letra de la unidad de CD-ROM de Windows
   - *cdrom* se refiere al punto de montaje de UNIX del CD-ROM
   - %*L* es el entorno nacional de la documentación que desea imprimir, por ejemplo, en_US
4. Pulse **Archivo –> Imprimir**.

5. En la página Imprimir, especifique si desea imprimir todas las páginas, las páginas actuales o un rango de páginas.
6. Pulse **Aceptar**.

Solicitud de manuales impresos

Puede obtener manuales impresos solicitando el paquete de la documentación (doc pack) para el producto DB2 Information Integrator al distribuidor de IBM. Los doc packs son un subconjunto de manuales de la biblioteca de DB2 Information Integrator. Estos doc packs están pensados para ayudar al usuario a empezar a utilizar el producto DB2 Information Integrator que ha adquirido.

Los manuales en los doc packs son los mismos que los del CD de documentación en PDF de DB2 Information Integrator que se proporciona con el producto DB2 Information Integrator.

También puede utilizar uno de los métodos siguientes para solicitar manuales individuales:

- Póngase en contacto con el representante de márketing de IBM o distribuidor autorizado. Para encontrar un representante local de IBM, consulte el IBM Worldwide Directory of Contacts en la página Web www.ibm.com/planetwide.
- Llame al teléfono 1-800-879-2755 si está en los Estados Unidos o al 1-800-IBM-4YOU si está en Canadá.
- Visite el IBM Publications Center en www.ibm.com/shop/publications/order.

Notas del release y requisitos de la instalación

Las notas del release y los requisitos de la instalación proporcionan información específica para el nivel de release y de FixPak del producto. Las notas del release también proporcionan resúmenes de las actualizaciones de la documentación que se incorporan en cada release y FixPak.

Las notas del release y los requisitos de la instalación están disponibles en formato texto y HTML en el CD-ROM del producto:

- En sistemas operativos Windows: *x*:\doc\%*L*
- En sistemas operativos UNIX: /*cdrom*/doc/%*L*/

donde:

- *x* representa la letra de la unidad de CD-ROM de Windows
- *cdrom* se refiere al punto de montaje de UNIX del CD-ROM
- %*L* es el entorno nacional de la documentación que desee utilizar; por ejemplo, en_US

Tabla 335. Notas del release

Nombre	Nombre de archivo	Ubicación
DB2 Information Integrator Notas del release	ReleaseNotes	• CD-ROM del producto • Centro de información de DB2 Information Integrator • Área de ejecución de la instalación de DB2 Information Integrator
DB2 Information Integrator Requisitos de instalación	Prereqs	• CD-ROM del producto • Área de ejecución de la instalación de DB2 Information Integrator

FixPaks para la documentación de DB2 Information Integrator

IBM puede poner periódicamente a disposición del usuario FixPaks de documentación. Puede utilizar los FixPak de documentación para actualizar la información que ha instalado desde el CD de documentación en HTML de DB2 a medida que aparezca nueva información.

Los FixPak de documentación son acumulativos. Por ejemplo, si ha instalado la documentación para la Versión 8.1 y, a continuación, aplica la Versión 8.1.2, obtendrá actualizaciones de documentación para el FixPak 1 y la Versión 8.1.2.

Cuando instale los FixPak de documentación, la documentación en HTML estará más actualizada que los manuales PDF impresos y en línea para el producto.

Acceso a los temas utilizando el Centro de información de DB2 Information Integrator o el CD de documentación en HTML de DB2

El Centro de información de DB2 Information Integrator proporciona acceso a la información que necesite para obtener el máximo provecho de DB2 Information Integrator en su empresa.

Características del Centro de información de DB2 Information Integrator

El Centro de información de DB2 Information Integrator tiene las siguientes características:

Árbol de navegación integrado

Localiza cualquier tema de la biblioteca de DB2 Information Integrator desde un sólo árbol de navegación.

Búsqueda
Busca todos los temas de la estación de trabajo pulsando **Buscar** en la barra de herramientas de navegación.

Índice maestro
Accede a la información en la ayuda de temas y herramientas desde un índice maestro. El índice contiene entradas de toda la biblioteca de DB2 Information Integrator.

Glosario maestro
El glosario maestro define los términos utilizados en la biblioteca de DB2 Information Integrator.

Documentación actualizada con regularidad
Mantiene los temas actualizados bajando los temas HTML actualizados.

Búsqueda de temas en el Centro de información de DB2 Information Integrator

Los siguientes elementos principales constituyen el Centro de información de DB2 Information Integrator:

Árbol de navegación
El árbol de navegación está ubicado en el marco izquierdo de la ventana del navegador. El árbol se expande y se contrae para mostrar y ocultar los enlaces de temas, el glosario y el índice maestro del Centro de información de DB2 Information Integrator.

Barra de herramientas de navegación
La barra de herramientas de navegación está ubicada en el marco superior derecho de la ventana del navegador. Utilice los pulsadores de la barra de herramientas de navegación para buscar en el Centro de información de DB2 Information Integrator, ocultar el árbol de navegación y encontrar el tema que se visualiza actualmente en el árbol de navegación.

Marco de contenido
El marco de contenido está ubicado en el marco inferior derecho de la ventana del navegador. Cuando pulse un enlace en el árbol de navegación, un resultado de la búsqueda o siga un enlace desde otro tema o desde el índice maestro, el marco de contenido visualizará el tema apropiado.

Prerrequisitos:

Para acceder al Centro de información de DB2 Information Integrator desde un navegador, deberá utilizar uno de los navegadores siguientes:

- Microsoft Explorer versión 5 o posterior
- Netscape Navigator versión 6.1 o posterior

Restricciones:

El Centro de información de DB2 Information Integrator contiene solamente los conjuntos de temas que ha instalado desde el CD de documentación en HTML de DB2. Si el navegador Web devuelve un error `Archivo no encontrado` al seguir el enlace a un tema, instale uno o más conjuntos de temas adicionales del CD de documentación en HTML de DB2.

Procedimiento:

Para encontrar un tema buscándolo mediante palabras clave:

1. En la barra de herramientas de navegación, pulse **Buscar**.
2. En el campo de entrada de texto superior de la ventana Buscar, entre dos o más términos relacionados con el área de interés y, a continuación, pulse **Buscar**. El campo **Resultados** visualizará una lista de temas. Los temas que coincidan más con la serie de búsqueda estarán al principio de la lista.

   Entre más términos para aumentar la precisión de la consulta y reducir el número de temas devueltos de la misma.
3. En el campo **Resultados**, pulse el título del tema que desee leer. El tema se visualizará en el marco de contenido.

Para encontrar un tema en el árbol de navegación:

1. En el árbol de navegación, pulse el icono de libro junto a la categoría de temas en la que está interesado. Se visualizará una lista de subcategorías bajo el icono.
2. Siga pulsando los iconos de libro hasta que encuentre la categoría que contiene los temas en los que está interesado. Las categorías que enlazan con temas muestran el título de la categoría como un enlace al mover el cursor sobre el título de la categoría. Se utiliza un icono de página en el árbol de navegación para identificar los temas.
3. Pulse el enlace al tema. El tema se visualizará en el marco de contenido.

Para encontrar un tema utilizando el índice maestro:

1. En el árbol de navegación, pulse **Índice**. El índice se expandirá para visualizar una lista de los enlaces ordenados alfabéticamente.
2. En el árbol de navegación, pulse el primer carácter del término que está buscando. Se visualizará una lista de entradas con ese carácter inicial en el marco de contenido. Si se visualiza un icono de libro, el término tendrá varias entradas de índice.
3. Pulse el icono de libro que corresponda al término en el que esté interesado. Se visualizará una lista de temas bajo el término que ha pulsado.

4. Pulse el título del tema que satisface sus necesidades. El tema se visualizará en el marco de contenido.

Utilización de la documentación en formato HTML de DB2

Este tema describe cómo instalar, ver y copiar la documentación del CD de documentación en HTML de DB2, y cómo actualizarla después de instalarla.

Instalación de la documentación en formato HTML de DB2

El directorio de instalación para el CD de documentación en HTML de DB2 es diferente para cada categoría de información:

vía_acceso_CD_html/doc/htmlcd/*%L*/*categoría*

vía_acceso_CD_html
Directorio en el que se instala el CD de documentación en HTML de DB2.

%L Entorno nacional de la documentación que desee utilizar, por ejemplo, en_US.

categoría
Identificador de categoría, por ejemplo, getting_started para la información de instalación.

Cómo ver documentación técnica directamente desde el CD de documentación en HTML de DB2

Todos los temas HTML se pueden ver desde el CD de documentación en HTML de DB2.

Restricciones:

Deberá instalar el producto DB2 para ver la ayuda en línea.

Procedimiento:

Para ver la documentación en HTML desde el CD de documentación en HTML de DB2:

1. Inserte el CD de documentación en HTML de DB2.
2. Inicie el navegador Web y abra el siguiente archivo:
   - Para sistemas operativos Windows:
     x:\Archivos de programa\IBM\sqllib\doc\htmlcd*%L*\index.*
   - Para sistemas operativos UNIX:
     /cdrom/Archivos de programa/IBM/sqllib/doc/htmlcd/*%L*/index.htm

   donde:
   - *x* representa la letra de la unidad de CD-ROM de Windows
   - *cdrom* se refiere al punto de montaje de UNIX del CD-ROM

- *%L* es el entorno nacional de la documentación que desea utilizar, por ejemplo, en_US

Copia de archivos desde el CD de documentación en HTML de DB2 en un servidor Web

La biblioteca completa de DB2 está disponible en el CD de documentación en HTML de DB2, de manera que puede copiar la biblioteca en un servidor Web para acceder a la misma fácilmente.

Procedimiento:

Copie los archivos desde el CD de documentación en HTML de DB2 en la vía de acceso apropiada del servidor Web (se muestra la vía de acceso por omisión):

- Para sistemas operativos Windows: *x*:\Archivos de programa\IBM\sqllib\doc\htmlcd*%L**.*
- Para sistemas operativos UNIX: */cdrom*/Archivos de programa/IBM/sqllib/doc/htmlcd/*%L*

donde:

- *x* representa la letra de la unidad de CD-ROM de Windows
- *cdrom* se refiere al punto de montaje de UNIX del CD-ROM
- *%L* es el entorno nacional de la documentación que desea utilizar, por ejemplo, en_US

Actualización de la documentación en HTML en el sistema

Cuando haya actualizaciones de IBM disponibles, podrá actualizar los archivos HTML que ha instalado desde el CD de documentación en HTML de DB2 realizando una de las acciones siguientes:

- Utilizando el Centro de información (si tiene instaladas las herramientas administrativas de la interfaz gráfica de usuario de DB2)
- Bajando y aplicando el FixPak de documentación en HTML de DB2

Este procedimiento no actualiza el código de DB2.

Prerrequisitos:

Asegúrese de que el sistema tenga acceso a Internet, ya que el actualizador baja el FixPak de documentación más reciente desde el servidor de IBM, si es necesario. Para conectarse a Internet, puede que tenga que proporcionar la información del proxy.

Procedimiento:

Para utilizar el Centro de información para actualizar la documentación en HTML local:

1. Inicie el Centro de información de DB2:
   - Desde las herramientas de administración gráficas, pulse el icono **Centro de información** en la barra de herramientas.
   - En la línea de mandatos, entre db2ic.
2. Pulse **Centro de información** –> **Actualizar documentación local** para iniciar la actualización.

   Si existe una actualización de documentación disponible, se bajará y aplicará.

Para bajar y aplicar manualmente la actualización de documentación:

1. Abra la página de soporte de DB2 en el navegador Web en la siguiente dirección www.ibm.com/software/data/db2/udb/winos2unix/support.
2. Pulse **DB2 versión 8** y busque el enlace del FixPak de documentación para su sistema operativo.
3. Determine si la documentación local de DB2 está anticuada comparando el nivel del FixPak de documentación con el nivel de la documentación que tiene instalado.
4. Si se encuentra disponible una versión más reciente de la documentación, baje el FixPak para el sistema operativo. Existe un FixPak para todos los sistemas operativos Windows y un FixPak para todos los sistemas operativos UNIX.
5. Aplique el FixPak:
   - Para sistemas operativos Windows: el FixPak de documentación es un archivo zip autoextraíble. Baje el FixPak de documentación en un directorio vacío y desempaquételo. Ejecute el mandato **setup** en ese directorio para instalar el FixPak de documentación.
   - Para sistemas operativos UNIX: el FixPak de documentación es un archivo tar.Z comprimido. Descomprima y desempaquete el archivo para crear un directorio llamado `delta_install`. Ejecute el script installdocfix dentro de ese directorio para instalar el FixPak de documentación.

Búsqueda en la documentación de DB2

Para buscar en la documentación de DB2, utilice Netscape versión 6.1 (o posterior) o Microsoft Internet Explorer versión 5 (o posterior). Asegúrese de que el soporte Java del navegador esté habilitado.

Al pulsar el icono de búsqueda de la barra de herramientas de navegación del Centro de información de DB2 Information Integrator se abrirá una ventana

en el navegador. Si va a utilizar la función de búsqueda por primera vez, es posible que la ventana de búsqueda tarde aproximadamente un minuto en cargarse.

Restricciones:

Al realizar búsquedas en la documentación se aplican las siguientes restricciones:

- No se soportan las búsquedas booleanas. Los calificadores de búsqueda booleanos *and* y *or* se pasan por alto en una búsqueda. Por ejemplo, las siguientes búsquedas producen los mismos resultados:
  - servlets *and* beans
  - servlets *or* beans
- No se soportan las búsquedas con caracteres comodín. Una búsqueda de *java** buscará la serie literal *java** y no encontrará, por ejemplo, *javadoc*.

En general, obtendrá mejores resultados de las búsquedas si busca frases en lugar de palabras.

Procedimiento:

Si desea realizar búsquedas en la documentación de DB2:

1. En la barra de herramientas de navegación, pulse **Buscar**.
2. En el campo de entrada de texto superior de la ventana Buscar, entre dos o más términos relacionados con el área de interés y, a continuación, pulse **Buscar**. El campo **Resultados** visualizará una lista de los temas ordenados según la exactitud.

   Entre más términos para aumentar la precisión de la consulta y reducir el número de temas devueltos de la misma.
3. En el campo **Resultados**, pulse el título del tema que desee leer. El tema se visualiza en el marco de contenido.

Cuando realice una búsqueda, el primer resultado se cargará automáticamente en el marco del navegador. Para ver el contenido de otros resultados de la búsqueda, pulse en el resultado en la lista de resultados.

Resolución de problemas de búsqueda de documentación de DB2 con Netscape 4.x

La mayoría de los problemas de búsqueda están relacionados con el soporte Java que los navegadores Web proporcionan. Este tema proporciona posibles soluciones.

Procedimiento:

Un problema común con Netscape 4.x implica la ausencia o la colocación errónea de una clase de seguridad. Intente la siguiente solución, especialmente si ve la línea siguiente en la consola Java del navegador:

```
Cannot find class  java/security/InvalidParameterException
```

Copie los siguientes archivos desde el CD de documentación en HTML de DB2 en el directorio `java\classes\java\security\` dentro del directorio en el que el navegador Netscape está instalado. Puede que tenga que crear la estructura de subdirectorios `java\security\`.

- En sistemas operativos Windows:

  ```
  x:Archivos de programa\IBM\sqllib\doc\htmlcd\%L
  \InvalidParameterException.class
  ```

- En sistemas operativos UNIX:

  ```
  /cdrom/Archivos de programa/IBM/sqllib/doc/htmlcd/%L
  /InvalidParameterException.class
  ```

donde:

- *x* representa la letra de la unidad de CD-ROM de Windows
- *cdrom* se refiere al punto de montaje de UNIX del CD-ROM
- *%L* es el entorno nacional de la documentación que desea utilizar, por ejemplo, en_US

Si el navegador Netscape sigue sin visualizar la ventana de entrada de búsqueda, intente las siguientes acciones:

- Detenga todas las instancias de los navegadores Netscape para asegurarse de que no hay ningún código de Netscape en ejecución en la máquina. A continuación, abra una nueva instancia del navegador Netscape e inicie la búsqueda otra vez.
- Depure la antememoria del navegador.
- Pruebe una versión diferente de Netscape o un navegador diferente.

Accesibilidad

Los usuarios con discapacidades físicas, como por ejemplo mobilidad reducida o visión limitada, pueden utilizar los productos de software satisfactoriamente utilizando las características de accesibilidad. Las principales características de accesibilidad de DB2 Information Integrator Versión 8 son las siguientes:

- Es posible trabajar con todas las características utilizando el teclado en lugar del ratón.
- Es posible personalizar el tamaño y el color de los fonts.
- Es posible recibir señales de alerta visuales o sonoras.
- DB2 soporta las aplicaciones de accesibilidad que utilizan la API de accesibilidad de Java.
- La documentación de DB2 se proporciona en un formato accesible.

Entrada de teclado y navegación

Es posible utilizar las herramientas de bases de datos de DB2, como por ejemplo el Centro de control, el Centro de depósito de datos y el Centro de duplicación, utilizando solamente el teclado. Puede utilizar teclas o combinaciones de teclas en lugar del ratón para realizar la mayoría de las operaciones.

En sistemas basados en UNIX, la posición del foco del teclado está resaltada, lo que indica qué área de la ventana está activa y dónde serán efectivas las pulsaciones.

Pantalla accesible

Las herramientas de bases de datos de DB2 tienen características que amplían la interfaz de usuario y mejoran la accesibilidad para los usuarios con visión reducida. Estas mejoras de la accesibilidad incluyen soporte para propiedades de font personalizables.

Valores de font

Para las herramientas de bases de datos de DB2, puede utilizar el cuaderno Valores de herramientas para seleccionar el color, el tamaño y el font para el texto de los menús y las ventanas.

Sin dependencias de color

No es necesario distinguir los colores para utilizar cualquiera de las funciones de este producto.

Señales de alerta alternativas

Puede especificar si desea recibir las alertas a través de señales visuales o sonoras, utilizando el cuaderno Valores de herramientas.

Compatibilidad con tecnologías de asistencia

La interfaz gráfica de DB2 Information Integrator soporta la API de accesibilidad de Java que permite el uso de lectores de pantalla y de otras tecnologías de asistencia utilizadas por personas discapacitadas.

Documentación accesible

La documentación para la familia de productos DB2 está disponible en formato HTML. Puede ver la documentación de acuerdo con las preferencias de pantalla establecidas en el navegador. Puede utilizar lectores de pantalla y otras tecnologías de asistencia.

Avisos

Esta información se ha desarrollado para productos y servicios que se ofrecen en los EE.UU. Es posible que IBM no comercialice los productos, servicios o características descritos en este documento en todos los países. Consulte al representante local de IBM para obtener información sobre los productos y servicios que actualmente pueden adquirirse en su zona. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se pueda utilizar dicho producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ninguno de los derechos de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patentes en tramitación que afecten al tema tratado en este documento. La posesión de este documento no confiere ninguna licencia sobre dichas patentes. Puede realizar consultas sobre licencias escribiendo a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
EE.UU.

Para realizar consultas sobre licencias referentes a información de doble byte (DBCS), puede ponerse en contacto con el Departamento de Propiedad Intelectual de IBM de su país/región o escribir a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokio 106, Japón

El párrafo siguiente no es aplicable al Reino Unido ni a ningún país/región en donde tales disposiciones sean incompatibles con la legislación local:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN “TAL CUAL”, SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN DE DERECHOS, COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN DETERMINADO. Algunos estados no permiten la exclusión de garantías

expresas o implícitas en determinadas transacciones, por lo que es posible que esta declaración no sea aplicable en su caso.

Esta publicación puede contener inexactitudes técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; dichos cambios se incorporarán a las nuevas ediciones de la publicación. IBM puede efectuar, en cualquier momento y sin previo aviso, mejoras y cambios en los productos y programas descritos en esta publicación.

Las referencias hechas en esta publicación a sitios Web que no son de IBM se proporcionan sólo para la comodidad del usuario y no constituyen un aval de esos sitios Web. La información contenida en esos sitios Web no forma parte de la información del presente producto IBM y el usuario es responsable de la utilización de dichos sitios Web.

IBM puede utilizar o distribuir cualquier información que se le facilite de la manera que considere adecuada, sin contraer por ello ninguna obligación con el remitente.

Los propietarios de licencias 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 Corporation
J46A/G4
555 Bailey Avenue
San José, CA 95141-1003
EE.UU.

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluido en algunos casos el pago de una tarifa.

El programa bajo licencia descrito en este documento y todo el material bajo licencia asociado a él, los proporciona IBM según los términos del Acuerdo de Cliente de IBM, el Acuerdo Internacional de Programas Bajo Licencia de IBM o cualquier acuerdo equivalente entre el usuario e IBM.

Los datos de rendimiento contenidos en este documento se obtuvieron en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse realizado en sistemas experimentales y no es seguro que estas mediciones sean las mismas en los sistemas disponibles comercialmente. Además, algunas mediciones pueden haberse calculado mediante extrapolación. Los resultados reales pueden variar. Los usuarios del presente manual deben verificar los datos aplicables para su entorno específico.

La información referente a productos que no son de IBM se ha obtenido de los proveedores de esos productos, de sus anuncios publicados o de otras fuentes disponibles públicamente. IBM no ha probado esos productos y no puede confirmar la exactitud del rendimiento, la compatibilidad ni ninguna otra afirmación referente a productos que no son de IBM. Las preguntas sobre las prestaciones de productos que no son de IBM deben dirigirse a los proveedores de esos productos.

Todas las declaraciones de intenciones de IBM están sujetas a cambio o cancelación sin previo aviso, y sólo representan objetivos.

Este manual puede contener ejemplos de datos e informes que se utilizan en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con nombres y direcciones utilizados por una empresa real es totalmente fortuita.

LICENCIA DE COPYRIGHT:

Este manual puede contener programas de aplicaciones de ejemplo escritos en lenguaje fuente, que muestran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo como desee, sin pago alguno a IBM, con la intención de desarrollar, utilizar, comercializar o distribuir programas de aplicaciones de acuerdo con la interfaz de programación de aplicaciones correspondiente a la plataforma operativa para la que están escritos los programas de ejemplo. Estos ejemplos no se han probado exhaustivamente bajo todas las condiciones. Por lo tanto, IBM no puede asegurar ni implicar la fiabilidad, utilidad o función de estos programas.

Cada copia o parte de estos programas de ejemplo o cualquier trabajo derivado debe incluir una nota de copyright como la siguiente:

© (*nombre de la empresa*) (*año*). Partes de este código proceden de programas de ejemplo de IBM Corp. © Copyright IBM Corp. *_entre el o los años_*. Reservados todos los derechos.

Marcas registradas

Los siguientes términos son marcas registradas de International Business Machines Corporation en los EE.UU. y/o en otros países:

IBM
DB2

Los siguientes términos son marcas registradas de otras empresas:

Java y todas las marcas registradas y los logotipos basados en Java son marcas registradas de Sun Microsystems, Inc. en los EE.UU. y/o en otros países.

Microsoft y Windows son marcas registradas de Microsoft Corporation en los EE.UU. y/o en otros países.

UNIX es marca registrada de The Open Group en los EE.UU. y/o en otros países.

Otros nombres de empresas, productos o servicios pueden ser marcas registradas o marcas de servicio de otras empresas.

Índice

N

O

P

R

S

U

W

Cómo ponerse en contacto con IBM

Para ponerse en contacto con IBM en los Estados Unidos o en Canadá, llame a uno de los siguientes números:

- Para el servicio de atención al cliente: 1-800-IBM-SERV (1-800-426-7378)
- Para márketing y ventas de DB2: 1-800-IBM-4YOU (1-800-426-4968)

Para obtener información acerca de las opciones de servicio disponibles, llame a uno de los siguientes números:

- En los Estados Unidos: 1-888-426-4343
- En Canadá: 1-800-465-9600

Para localizar una oficina de IBM en su país o región, consulte el IBM Directory of Worldwide Contacts en el sitio Web www.ibm.com/planetwide.

Información sobre productos

La información acerca de DB2 Information Integrator está disponible por teléfono o en la Web.

Si vive en los Estados Unidos, puede llamar a uno de los siguientes números:

- Para solicitar productos o para obtener información general: 1-800-IBM-CALL (1-800-426-2255)
- Para solicitar publicaciones: 1-800-879-2755

Vaya al sitio Web www.ibm.com/software/data/integration. Este sitio contiene la información más reciente sobre la biblioteca técnica, cómo solicitar manuales, descargas de clientes, grupos de noticias, FixPaks, novedades y enlaces a recursos de la Web.

Para localizar una oficina de IBM en su país o región, consulte el IBM Directory of Worldwide Contacts en el sitio Web www.ibm.com/planetwide.

Comentarios sobre la documentación

Sus comentarios ayudan a IBM a proporcionar una información de calidad. Envíe sus comentarios acerca de este manual u otra documentación de DB2 Information Integrator. Puede utilizar cualquiera de los siguientes métodos para proporcionar sus comentarios:

- Envíe sus comentarios utilizando el formulario en línea de comentarios del lector en el sitio www.ibm.com/software/data/rcf.

- Envíe sus comentarios por correo electrónico (e-mail) a HOJACOM@es.ibm.com. Asegúrese de incluir el nombre del producto, el número de versión del mismo y el nombre y número de pieza del manual (si es aplicable). Si sus comentarios se refieren a texto específico, incluya la ubicación del texto (por ejemplo, un título, un número de tabla o un número de página).

Spine information:

IBM DB2 Information Integrator

Consulta de la API del desarrollador de reiniciadores

Versión 8