IBM DB2 Information Integrator

Suplemento de la Guía de configuración de fuentes de datos: Funciones definidas por el usuario de reiniciador de BioRS y de ciencias de la vida

Versión 8

IBM DB2 Information Integrator

Suplemento de la Guía de configuración de fuentes de datos: Funciones definidas por el usuario de reiniciador de BioRS y de ciencias de la vida

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 91.

Este manual es la traducción del original inglés *IBM DB2 Information Integrator Addendum to the Data Source Configuration Guide: BioRS Wrapper and Life Sciences User-Defined Functions, 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

Capítulo 1. Configuración del acceso a fuentes de datos de BioRS

Este capítulo explica qué es BioRS, cómo añadir fuentes de datos de BioRS al sistema federado y lista los mensajes de error asociados con el reiniciador de BioRS.

¿Qué es BioRS?

BioRS es un sistema de consulta y recuperación desarrollado por Biomax Informatics. Puede utilizar BioRS para recuperar información de varias fuentes de datos, incluidos archivos planos y bases de datos relacionales. Por lo general se bajan datos públicos, tales como SwissProt y GenBank, como archivos planos al sistema BioRS. BioRS puede integrar datos públicos y fuentes de datos de propiedad (por ejemplo, bases de datos privadas mantenidas por la propia organización) en un entorno común.

Una vez se ha integrado una fuente de datos al sistema BioRS, se hace referencia a la misma como *banco de datos*. Como referencia colectiva a los elementos contenidos en cada entrada del banco de datos se utiliza el término *esquema*. En una consulta de BioRS, tan solo se pueden utilizar los elementos de un banco de datos que estén indexados en el sistema BioRS. Se pueden establecer relaciones entre entradas de bancos de datos, de modo que se puedan enlazar bancos de datos del sistema BioRS.

Los bancos de datos de BioRS pueden tener una relación padre-hijo (los bancos de datos se pueden anidar). En esta relación, el banco de datos hijo contiene un elemento de tipo de datos Reference llamado PADRE. El elemento PADRE hace referencia al elemento _ID_ del banco de datos padre. Además de la presencia de este elemento PADRE predefinido, los bancos de datos anidados contienen los mismos datos que los bancos de datos no anidados.

BioRS proporciona una interfaz basada en la Web que permite que los usuarios realicen consultas en los datos de los bancos de datos de BioRS. El reiniciador de BioRS utiliza las mismas interfaces de programación de aplicaciones (API) que la interfaz basada en la Web de BioRS para efectuar consultas.

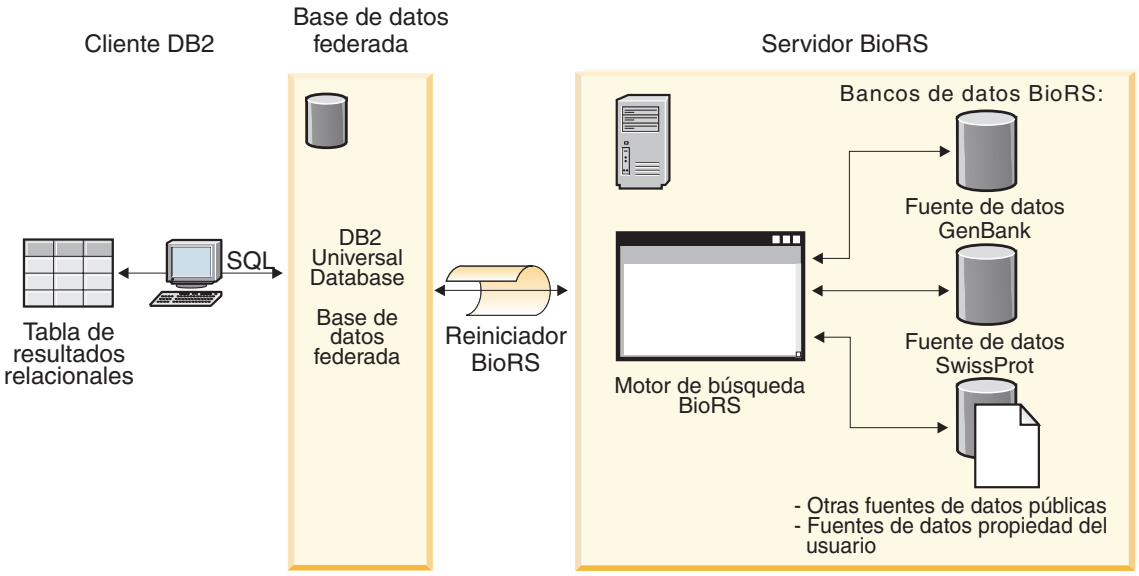


Figura 1. Cómo funciona el reiniciador de BioRS

Desde el cliente, los usuarios o las aplicaciones someten una consulta utilizando sentencias de SQL. A continuación, la consulta se envía al sistema federado en el que se ha instalado el reiniciador de BioRS. Según cómo se construya la consulta, se puede utilizar tanto DB2® Universal Database como el servidor BioRS para procesar la consulta. El servidor BioRS puede estar en un sistema distinto del sistema federado. El sistema federado debe proporcionar información de autentificación al servidor BioRS para cada consulta. Esta información puede ser una combinación de un ID de usuario y una contraseña o una indicación de no autentificado (generalmente una cuenta de huésped).

El reiniciador de BioRS funciona con BioRS Versión 5.0.14.

Para obtener información detallada sobre el producto BioRS, consulte el sitio Web de Biomax en: http://www.biomax.com.

Tareas relacionadas:

- "Adición de BioRS a un sistema federado" en la página 3

Información relacionada:

- "Reiniciador de BioRS - Consultas de ejemplo" en la página 21

Adición de BioRS a un sistema federado

Puede utilizar una fuente de datos de BioRS con el servidor federado si registra funciones personalizadas y un reiniciador de BioRS. A continuación, registre un servidor BioRS, las correlaciones de usuario y los apodos correspondientes para que el servidor federado pueda recuperar y procesar datos de BioRS.

Puede ejecutar las sentencias de SQL desde el Centro de control de DB2 o desde el procesador de línea de mandatos de DB2. Después de añadir BioRS al sistema federado, puede ejecutar consultas en una fuente de datos de BioRS.

Procedimiento:

Para añadir una fuente de datos de BioRS a un servidor federado:

1. Registre funciones personalizadas utilizando la sentencia CREATE FUNCTION.
2. Registre el reiniciador de BioRS utilizando la sentencia CREATE WRAPPER.
3. Opcional: Establezca la variable de entorno DB2_DJ_COMM para mejorar el rendimiento de consulta.
4. Registre el servidor BioRS utilizando la sentencia CREATE SERVER.
5. Opcional: Registre usuarios autorizados utilizando la sentencia CREATE USER MAPPING.
6. Registre apodos utilizando la sentencia CREATE NICKNAME.
7. Opcional: Actualice las estadísticas de cardinalidad para columnas de BioRS.

Tareas relacionadas:

- “Registro de funciones personalizadas para el reiniciador de BioRS” en la página 4
- “Registro del reiniciador de BioRS” en la página 5
- “Cómo establecer la variable de perfil DB2_DJ_COMM para el reiniciador de BioRS” en la página 6
- “Registro del servidor para una fuente de datos de BioRS” en la página 7
- “Registro de correlaciones de usuario para fuentes de datos de BioRS” en la página 7
- “Registro de apodos para las fuentes de datos de BioRS” en la página 9
- “Actualización de las estadísticas de cardinalidad de columnas de BioRS” en la página 12

Registro de funciones personalizadas para el reiniciador de BioRS

El registro de las funciones personalizadas para el reiniciador de BioRS forma parte de la tarea más amplia de añadir BioRS a un sistema federado. Una vez que haya registrado las funciones personalizadas, debe registrar el reiniciador.

Puede utilizar el archivo de ejemplo create_function_mappings.ddl para registrar funciones personalizadas. Este archivo se encuentra en el directorio sqllib/samples/lifesci/biors. El archivo create_function_mappings.ddl contiene definiciones para cada función personalizada. Puede ejecutar este archivo DDL para registrar las funciones personalizadas para cada base de datos de DB2 que tenga instalado el reiniciador de BioRS.

Requisitos previos:

- Todas las funciones personalizadas para el reiniciador de BioRS deben registrarse con el nombre de esquema BioRS.
- Debe registrar una vez cada función personalizada para cada base de datos de DB2 que tenga el reiniciador de BioRS instalado.

Procedimiento:

Para registrar funciones personalizadas, debe emitir la sentencia CREATE FUNCTION con la palabra clave TEMPLATE.

El nombre totalmente calificado de cada función es BioRS.<nombre-función>.

El siguiente ejemplo registra una versión de la función CONTAINS:

```
CREATE FUNCTION biors.contains (varchar(), varchar())
RETURNS INTEGER AS TEMPLATE;
```

La tarea siguiente de esta secuencia de tareas es el registro del reiniciador apropiado de BioRS.

Tareas relacionadas:

- "Registro del reiniciador de BioRS" en la página 5

Información relacionada:

- "Sentencia CREATE FUNCTION (Con origen o plantilla)" en la publicación *Consulta de SQL, Volumen 2*
- "Funciones personalizadas y consultas de BioRS" en la página 16
- "Reiniciador de BioRS - Consultas de ejemplo" en la página 21
- "Tabla de funciones personalizadas - Reiniciador de BioRS" en la página 32

Registro del reiniciador de BioRS

El registro del reiniciador de BioRS forma parte de la tarea más amplia de añadir BioRS a un sistema federado. Debe registrar el reiniciador para poder acceder a una fuente de datos. Los reiniciadores son mecanismos utilizados por los servidores federados a fin de comunicarse con las fuentes de datos y recuperar datos de las mismas. Los reiniciadores se instalan en el sistema como archivos de biblioteca. La Tabla 1 lista los archivos de biblioteca de BioRS por omisión y el sistema operativo soportado para cada archivo.

Tabla 1. Archivos de biblioteca de BioRS y sistemas operativos soportados

Archivo de biblioteca de BioRS	Sistema operativo
libdb2lsbiors.a	IBM AIX Versión 4.3.3 o posterior
db2lsbiors.dll	Microsoft Windows NT Versión 4 Microsoft Windows 2000 Microsoft Windows XP

Procedimiento:

Para registrar el reiniciador de BioRS, emita la sentencia CREATE WRAPPER.

Por ejemplo, para registrar un reiniciador de BioRS en AIX denominado `wrap_biors` desde el archivo de biblioteca por omisión, `libdb2lsbiors.a`, emita la sentencia siguiente:

```
CREATE WRAPPER reiniciador_biors LIBRARY 'libdb2lsbiors.a';
```

Tareas relacionadas:

- "Comprobación del reiniciador no relacional y de las bibliotecas de funciones definidas por el usuario de ciencias de la vida" en la publicación *DB2 Information Integrator Guía de instalación*
- "Cómo establecer la variable de perfil DB2_DJ_COMM para el reiniciador de BioRS" en la página 6

Información relacionada:

- "Sentencia CREATE WRAPPER" en la publicación *Consulta de SQL, Volumen 2*

Cómo establecer la variable de perfil DB2_DJ_COMM para el reiniciador de BioRS

El establecimiento de la variable de perfil DB2_DJ_COMM para el reiniciador de BioRS forma parte de la tarea más amplia de añadir BioRS a un sistema federado. Para mejorar el rendimiento cuando se acceda a fuentes de datos de BioRS, puede establecer opcionalmente la variable de perfil de DB2 DB2_DJ_COMM. Esta variable especifica si el servidor federado carga el reiniciador en la inicialización.

El uso del procesador aumenta cuando el servidor federado carga las bibliotecas del reiniciador durante el inicio de la base de datos. Para evitar un uso excesivo, especifique únicamente las bibliotecas a las que piense acceder.

Procedimiento:

Para establecer la variable de perfil de DB2 DB2_DJ_COMM, emita el mandato **db2set** con la biblioteca de reiniciador que corresponda al reiniciador especificado en la sentencia CREATE WRAPPER asociada.

Por ejemplo:

```
db2set DB2_DJ_COMM='libdb2lsbiors.a'
```

Asegúrese de que no haya espacios en ningún lado del signo de igualdad (=).

La tarea siguiente de esta secuencia de tareas es el registro del servidor para BioRS.

Conceptos relacionados:

- "Environment Variables and the Profile Registry" en la publicación *Administration Guide: Implementation*

Tareas relacionadas:

- "Registro del servidor para una fuente de datos de BioRS" en la página 7

Información relacionada:

- "db2set - Mandato Registro de perfiles de DB2" en la publicación *Consulta de mandatos*

Registro del servidor para una fuente de datos de BioRS

El registro del servidor para una fuente de datos de BioRS forma parte de la tarea más amplia de añadir BioRS a un sistema federado. Después de registrar el reiniciador, debe registrar el servidor correspondiente.

Procedimiento:

Para registrar el servidor BioRS en el sistema federado, emita una sentencia CREATE SERVER.

Por ejemplo:

```
CREATE SERVER brs_server WRAPPER wrap_biors OPTIONS(NODE 'biors_server2.com');
```

Tareas relacionadas:

- "Registro de apodos para las fuentes de datos de BioRS" en la página 9

Información relacionada:

- "Opciones de la sentencia CREATE SERVER - Reiniciador de BioRS" en la página 40

Registro de correlaciones de usuario para fuentes de datos de BioRS

El registro de correlaciones de usuario forma parte de la tarea más amplia de añadir BioRS a un sistema federado. Es posible que no sea necesario crear correlaciones de usuario, según el método de acceso de cuenta o los métodos que se utilicen en el sistema BioRS.

- Si el servidor BioRS está configurado para acceso de huésped para todas las cuentas de usuario, no es necesario crear correlaciones de usuario en DB2 Information Integrator.
- Si el servidor BioRS está configurado para autentificar cuentas de usuario con ID y contraseñas, es necesario crear correlaciones de usuario en la base de datos federada para las cuentas que deben utilizar el reiniciador de BioRS.
- Si el servidor BioRS está configurado para utilizar una mezcla de cuentas de huésped y de usuario autentificado, deben crearse correlaciones de usuario para las cuentas de usuario autentificadas en la base de datos federada para las cuentas que deben utilizar el reiniciador de BioRS.

Las correlaciones de usuario proporcionan una manera de autentificar el acceso de los usuarios o las aplicaciones que consultan una fuente de datos de BioRS con el reiniciador de BioRS. Si un usuario o una aplicación somete una consulta de SQL a un apodo registrado de BioRS y no se ha definido ninguna correlación de usuario para dicho usuario o dicha aplicación, el reiniciador de

BioRS utiliza un ID de usuario y una contraseña por omisión en un intento de recuperar los datos del servidor remoto de BioRS. Si un banco de datos que se está consultando requiere autentificación, podría devolverse un mensaje de error.

Para asegurarse de que se pasan el ID de usuario y la contraseña correctos al servidor BioRS, cree correlaciones de usuario en la base de datos federada para los usuarios autorizados para buscar en fuentes de datos de BioRS. Cuando se crea una correlación de usuario, la contraseña se almacena en formato cifrado en una tabla de catálogo del sistema de bases de datos federadas.

Procedimiento:

Para registrar correlaciones de usuarios de BioRS, utilice la sentencia CREATE USER MAPPING.

Por ejemplo, la sentencia CREATE USER MAPPING siguiente correlaciona el usuario Charlie con el usuario Charlene en el servidor Biors_Server1.

```
CREATE USER MAPPING FOR Charlie SERVER Biors_Server1
OPTIONS(REMOTE_AUTHID 'Charlene', REMOTE_PASSWORD 'Charlene_pw');
```

También puede definir su propia correlación de usuarios. En el ejemplo siguiente, USER es una palabra clave que identifica el usuario actual, no un nombre de usuario USER.

```
CREATE USER MAPPING FOR USER SERVER Biors_Server1
OPTIONS(REMOTE_AUTHID 'Yudong', REMOTE_PASSWORD 'Yudong_pw')
```

La tarea siguiente de esta secuencia de tareas es el registro de apodos para el reiniciador de BioRS.

Tareas relacionadas:

- "Registro de apodos para las fuentes de datos de BioRS" en la página 9

Información relacionada:

- "Sentencia CREATE USER MAPPING" en la publicación *Consulta de SQL, Volumen 2*
- "Opciones de la sentencia CREATE USER MAPPING - Reiniciador de BioRS" en la página 41

Registro de apodos para las fuentes de datos de BioRS

El registro de apodos para fuentes de datos de BioRS forma parte de la tarea más amplia de añadir BioRS a un sistema federado. Después de registrar un servidor, debe registrar un apodo para cada fuente de datos de BioRS a la que desee acceder. Los apodos se utilizan cuando se hace referencia a una fuente de datos de BioRS en una consulta.

Importante: Una vez se ha integrado una fuente de datos en el sistema BioRS, se hace referencia a la misma como *banco en datos* de BioRS. Los bancos de datos de BioRS equivalen a los apodos en un sistema federado.

Requisitos previos:

- Si un nombre de banco de datos de BioRS no se ajusta a la sintaxis federada de DB2, debe utilizar la opción de apodo REMOTE_OBJECT cuando registre el apodo.
- Si un nombre de elemento de BioRS no se ajusta a la sintaxis federada de DB2, debe utilizar la opción de columna ELEMENT_NAME cuando registre el apodo.

Restricciones:

No utilice el elemento AllText de BioRS como primera columna para un apodo. Puede utilizar el elemento AllText de BioRS en cualquier otra posición de columna (por ejemplo, como segunda columna o como tercera columna).

Procedimiento:

Para registrar un apodo de BioRS, utilice la sentencia CREATE NICKNAME.

Un apodo federado equivale directamente a un banco de datos de BioRS. Cuando crea un apodo federado debe definir una lista de columnas de apodo. Las columnas de apodo especificadas deben corresponder a elementos de un formato de banco de datos de BioRS. BioRS define cinco tipos de datos posibles para elementos: Texto, Número, Fecha, Autor y Consulta. Estos tipos de datos sólo se pueden correlacionar con los tipos de datos de sistema federado CLOB, CHAR o VARCHAR.

El modo más simple de registrar un apodo para un banco de datos de BioRS consiste en proporcionar al apodo el mismo nombre que el banco de datos de BioRS. Por ejemplo:

```
CREATE NICKNAME SwissProt
 (ID VARCHAR(32) OPTIONS (ELEMENT_NAME '_ID_'),
 ALLTEXT VARCHAR(128),
 ENTRYDATE VARCHAR (64))
FOR SERVER brs_server;
```

El banco de datos de BioRS implícito SwissProt es el nombre del apodo.

La utilización de esta sintaxis simple CREATE NICKNAME limita al usuario a una sola familia de apodos por esquema de DB2. Puede utilizar otros nombres especificando la opción REMOTE_OBJECT. Esta opción de apodo especifica que el nombre del tipo de objeto de BioRS debe estar asociado con el apodo. El nombre especificado en la opción REMOTE_OBJECT determina el esquema y el banco de datos de BioRS para el apodo. La opción REMOTE_OBJECT también especifica la relación del apodo con otros apodos.

En el ejemplo siguiente muestra el mismo conjunto de características de apodo que en el ejemplo anterior, pero se cambia el nombre de apodo y se utiliza la opción REMOTE_OBJECT para especificar el banco de datos BioRS para el que se está definiendo el apodo:

```
CREATE NICKNAME NewSP
 (ID VARCHAR(32) OPTIONS (ELEMENT_NAME '_ID_'),
 ALLTEXT VARCHAR(128),
 ENTRYDATE VARCHAR (64))
FOR SERVER brs_server
 OPTIONS (REMOTE_OBJECT 'SwissProt');
```

El banco de datos de BioRS implícito es SwissProt y el nombre del apodo es NewSP.

Conceptos relacionados:

- "Información estadística de BioRS" en la página 27

Tareas relacionadas:

- "Actualización de las estadísticas de cardinalidad de apodo de BioRS" en la página 29

Información relacionada:

- "Elemento AllText de BioRS" en la página 31
- "Sentencia CREATE NICKNAME - Ejemplos para reiniciador de BioRS" en la página 11
- "Sintaxis de la sentencia CREATE NICKNAME - Reiniciador de BioRS" en la página 38
- "Consideraciones para modificar apodos - Reiniciador de BioRS" en la página 32

Sentencia CREATE NICKNAME - Ejemplos para reiniciador de BioRS

Este tema proporciona ejemplos que muestran cómo utilizar la sentencia CREATE NICKNAME para registrar apodos para el reiniciador de BioRS.

Ejemplo 1:

El ejemplo siguiente muestra cómo crear un apodo para un banco de datos de BioRS remoto que no se ajusta a la sintaxis de DB2 Information Integrator:

```
CREATE NICKNAME SwissFT
 (ID VARCHAR(32) OPTIONS (ELEMENT_NAME '_ID_'),
 ALLTEXT VARCHAR (128),
 ENTRYDATE VARCHAR (64),
 FtLength VARCHAR (16),
   FOR SERVER biors1
 OPTIONS (REMOTE_OBJECT 'SwissProt.Features');
```

El nombre de este apodo es SwissFT. Las columnas de la tabla son ID, ALLTEXT, ENTRYDATE y FtLength. La opción de columna ELEMENT_NAME se especifica para la columna ID. Debe especificar la opción ELEMENT_NAME cuando el nombre de un elemento de BioRS no se ajusta a la sintaxis federada válida de DB2 para nombres de columna. En este ejemplo, elemento de BioRS _ID_ se ajusta a la sintaxis federada de DB2, pero `_ID_` es un nombre que potencialmente puede producir confusiones en los usuarios de DB2 Information Integrator. El nombre `ID` es simple y fácil de comprender. En general, utilice la opción ELEMENT_NAME en las siguientes circunstancias:

- Cuando un nombre de elemento de BioRS no se ajusta a la sintaxis federada válida de DB2
- Cuando la sensibilidad a las mayúsculas y minúsculas de un nombre de elemento de BioRS no se ajusta a los estándares de sistema federado de DB2 establecidos
- Cuando el nombre de un elemento de BioRS puede no ser obvio para los usuarios de DB2 Information Integrator

De modo adicional, la opción REMOTE_OBJECT se utiliza para especificar el nombre del banco de datos de BioRS al que equivale el apodo. Debe especificar la opción REMOTE_OBJECT cuando el nombre de un banco de datos de BioRS no se ajusta a la sintaxis federada válida de DB2. En este ejemplo, el nombre de banco de datos ″SwissProt.Features″ no se ajusta a la sintaxis federada válida de DB2. En general, utilice la opción REMOTE_OBJECT en las siguientes circunstancias:

- Cuando la sensibilidad a las mayúsculas y minúsculas de los nombres de banco de datos de BioRS no se ajusta a los estándares de sistema federado de DB2 establecidos

- Cuando el nombre de banco de datos de BioRS no se ajusta a la sintaxis federada válida de DB2
- Cuando el nombre de un banco de datos de BioRS puede no ser obvio para los usuarios de DB2 Information Integrator

Ejemplo 2:

El ejemplo siguiente muestra cómo crear un apodo para una tabla que utiliza un banco de datos de BioRS que está enlazado con otro banco de datos de BioRS:

```
CREATE NICKNAME SwissFT2
 (ID VARCHAR(32) OPTIONS (ELEMENT_NAME '_ID_'),
 ALLTEXT VARCHAR (1200),
 FtKey VARCHAR (32),
 FtLength VARCHAR (64),
   FtDescription VARCHAR (128),
   Parent VARCHAR (32) OPTIONS (REFERENCED_OBJECT 'SwissProt'))
 FOR SERVER biors1
 OPTIONS (REMOTE_OBJECT 'SwissProt.Features');
```

El nombre de este apodo es SwissFT2. Las columnas de la tabla son ID, ALLTEXT, FtKey, FtLength, FtDescription y Parent. La opción de columna ELEMENT_NAME se especifica para la columna ID. La opción REMOTE_OBJECT se utiliza para especificar el nombre del banco de datos de BioRS al cual corresponde el apodo.

Además, la columna Parent utiliza la opción REFERENCED_OBJECT. Debe especificar esta opción para las columnas que correspondan a los elementos del tipo de datos Reference de BioRS. La opción REFERENCED_OBJECT especifica el nombre del banco de datos de BioRS al cual hace referencia la columna. En este caso, el elemento Parent hace referencia al banco de datos SwissProt de BioRS.

Tareas relacionadas:

- "Registro de apodos para las fuentes de datos de BioRS" en la página 9

Información relacionada:

- "Sintaxis de la sentencia CREATE NICKNAME - Reiniciador de BioRS" en la página 38
- "Consideraciones para modificar apodos - Reiniciador de BioRS" en la página 32

Actualización de las estadísticas de cardinalidad de columnas de BioRS

Para actualizar las estadísticas de cardinalidad de columnas de BioRS en el sistema federado, debe modificar la vista de catálogo SYSSTAT.COLUMNS.

El mantenimiento de las estadísticas de cardinalidad correctas para columnas de BioRS permite que el optimizador y el reiniciador de BioRS elijan el mejor plan de acceso a datos de rendimiento durante el proceso de consulta.

Opcionalmente puede actualizar las estadísticas de cardinalidad de columnas de BioRS como parte de la tarea más amplia de añadir BioRS a un sistema federado. También puede actualizar las estadísticas de cardinalidad de columnas de BioRS si desea mejorar el rendimiento de la consulta para fuentes de datos de BioRS.

Restricciones:

No utilice este procedimiento para actualizar las estadísticas de cardinalidad para columnas que correspondan al elemento _ID_ de BioRS. Debe utilizar un procedimiento distinto para actualizar las estadísticas de cardinalidad para las columnas que correspondan al elemento _ID_ de BioRS.

Procedimiento:

Para actualizar las estadísticas de cardinalidad de columnas de BioRS, emita una sentencia UPDATE utilizando la sintaxis siguiente:

```
UPDATE sysstat.columns SET colcard=(SELECT COUNT(DISTINCT <nombre-columna>)
                                      FROM <esquema-apodo>.<nombre-apodo>)
WHERE  tabschema=<esquema-apodo>
  AND tabname=<nombre-apodo>
  AND colname=<nombre-columna>
```

- *<nombre-columna>* es el nombre de la columna cuyas estadísticas de cardinalidad desea actualizar.
- *<esquema-apodo>* es el nombre del esquema que está asociado con el apodo en el que se utiliza la columna especificada.
- *<nombre-apodo>* es el nombre del apodo en el que se utiliza la columna especificada.

La consulta puede tardar varios minutos en ejecutarse debido a que deben recuperarse todas las entradas para el banco de datos que se especifica en el apodo.

Si una columna puede contener varios valores (por ejemplo, el elemento PublicationYear del formato de base de datos de SwissProt), el cálculo pasa a ser demasiado complejo para utilizar una consulta de SQL. Para dichas columnas, debe calcular manualmente el valor de cardinalidad y, a continuación, actualizar la vista de catálogo SYSSTAT.COLUMNS. Para calcular el valor de cardinalidad, divida el número de valores diferenciados de la columna por el número medio de valores por fila. El valor de cardinalidad calculado no puede ser mayor que la cardinalidad de la tabla.

Ejemplo:

Supongamos que tiene un apodo con tres filas. Los valores de la columna PublicationYear para estas tres filas son:

- 1997 1992 1985
- 1997 1992 1982
- 1992 1991 1990 1976 1974 1971

Existen nueve valores diferenciados y el número medio de valores en una fila es cuatro. La cardinalidad para esta columna PublicationYear es 9/4 o 3 (2,25 redondeado al siguiente entero más alto). Una vez calculada la cardinalidad, puede actualizar la vista de catálogo SYSSTAT.COLUMNS utilizando la sentencia UPDATE siguiente:

```
UPDATE sysstat.columns SET colcard=3
WHERE  tabschema=<esquema-apodo>
  AND tabname=<nombre-apodo>
  AND colname=<nombre-columna>
```

- *3* es el valor de cardinalidad de columna.
- *<esquema-apodo>* es el nombre del esquema que está asociado con el apodo implícito en el que se utiliza la columna especificada.
- *<nombre-apodo>* es el nombre del apodo implícito en el que se utiliza la columna especificada.
- *<nombre-columna>* es el nombre de la columna cuyas estadísticas de cardinalidad desea actualizar.

Conceptos relacionados:

- "Información estadística de BioRS" en la página 27

Tareas relacionadas:

- "Actualización de las estadísticas de cardinalidad de apodo de BioRS" en la página 29
- "Actualización de la cardinalidad de columnas _ID_ de BioRS" en la página 30

Directrices para optimizar el rendimiento del reiniciador de BioRS

Este tema proporciona directrices sobre cómo optimizar el rendimiento de las consultas cuando se utiliza el reiniciador de BioRS.

Minimizar la cantidad de datos transferidos entre motores de búsqueda.
El entorno federado utiliza dos motores de consulta. Para el reiniciador de BioRS, estos motores de consulta son DB2® Universal Database y BioRS. El motor DB2 procesa predicados (operadores relacionales, tales como =, BETWEEN, LIKE y <>) especificados en las

columnas de apodo. El motor BioRS procesa predicados especificados utilizando cuatro funciones personalizadas para el reiniciador de BioRS.

Para minimizar la cantidad de datos transferidos entre los dos motores de búsqueda, estructure sus consultas de modo que el proceso de datos se pase al sistema BioRS siempre que sea posible.

Si necesita llevar a cabo operaciones de unión en una consulta, aproveche las relaciones padre-hijo que ya existen en los bancos de datos de BioRS y efectúe operaciones de unión de igualdad siempre que sea posible. Las operaciones de unión de igualdad se procesan en BioRS, lo que también minimiza la cantidad de datos transferidos entre los motores de consulta DB2 y BioRS.

Atención: no interrumpa las consultas de DB2 Information Integrator en BioRS (por ejemplo, utilizando **Control-D** o **Control-Z** en el procesador de línea de mandatos o deteniendo un programa de aplicación). La interrupción de una consulta deja ″muerto″ los procesos que están en ejecución en el servidor BioRS. Estos procesos ″muertos″ degradarán rápidamente el rendimiento de los sistemas BioRS y DB2 Information Integrator. Si hay bastantes procesos ″muertos″ en ejecución, se pueden producir errores inesperados durante el proceso de consulta de DB2 Information Integrator. Por ejemplo, una consulta válida puede devolver 0 filas, cuando se esperan filas. En situaciones extremas, BioRS, DB2 Information Integrator o ambos productos pueden detenerse o finalizar anormalmente.

Mantener información estadística de BioRS en el entorno federado.
En un sistema federado, la base de datos federada se basa en estadísticas de catálogos para que los objetos con apodo optimicen el proceso de las consultas. El mantenimiento de estadísticas actuales sobre las fuentes de datos de BioRS es esencial para optimizar el rendimiento del reiniciador de BioRS. Si los datos estadísticos o las características estructurales para un objeto remoto con los que está definido un apodo han cambiado, debe actualizar las estadísticas de cardinalidad de columna de apodo correspondientes en el sistema federado.

Para optimizar el rendimiento del reiniciador de BioRS, efectúe estas actualizaciones en DB2 Information Integrator en intervalos regulares.

Conceptos relacionados:

- “Tuning query processing” en la publicación *Federated Systems Guide*
- “Predicados de unión de igualdad para el reiniciador de BioRS” en la página 19
- “Información estadística de BioRS” en la página 27

Información relacionada:

- "Funciones personalizadas y consultas de BioRS" en la página 16
- "Reiniciador de BioRS - Consultas de ejemplo" en la página 21

Funciones personalizadas y consultas de BioRS

El entorno federado utiliza dos motores de consulta. Para el reiniciador de BioRS, estos motores de consulta son DB2 Universal Database y BioRS. Puede especificar qué predicados deben pasarse al motor BioRS utilizando las cuatro funciones personalizadas de BioRS, que son las siguientes:

- BIORS.CONTAINS
- BIORS.CONTAINS_LE
- BIORS.CONTAINS_GE
- BIORS.SEARCH_TERM

Estas cuatro funciones personalizadas están registradas en el esquema de BioRS. Debe utilizar el esquema de BioRS para hacer referencia a las funciones.

Las funciones personalizadas BIORS.CONTAINS, BIORS.CONTAINS_LE y BIORS.CONTAINS_GE requieren un argumento de columna de término de consulta y un argumento de texto de consulta. El ejemplo siguiente muestra una sentencia BIORS.CONTAINS:

```
BIORS.CONTAINS (<columna término búsqueda>,<término consulta>)
```

El valor del argumento de columna de término de búsqueda debe hacer referencia a una columna indexada de BioRS. La utilización de una columna no indexada genera el mensaje de error SQL30090N ("Operación no válida para el entorno de ejecución de aplicación").

El valor del argumento de término de consulta sólo puede ser un literal, una variable del lenguaje principal o una referencia de columna. No se puede utilizar concatenación aritmética o de cadenas. Además, el valor del argumento de término de consulta no puede ser NULL, aunque la columna de término de búsqueda utilizada esté definida para permitir valores nulos.

No importa si el argumento de término de consulta está en mayúsculas o minúsculas.

Los tipos de datos y formatos válidos del argumento de término de consulta dependen del tipo de datos de BioRS de la columna de término de búsqueda que se utiliza. BioRS define cinco tipos de datos posibles: Text, Author, Date, Number y Reference. Los tipos de datos de BioRS y los términos de consulta de función válidos para cada tipo de datos se listan en la Tabla 2 en la página 17.

Tabla 2. Tipos de datos de BioRS y términos de consulta de función personalizada válidos

Tipo de datos de columna de término de búsqueda	Término de consulta válido	Formato
Texto	VARCHAR() o CHAR()	Término de texto de BioRS, incluidos caracteres comodín.
Autor	VARCHAR() o CHAR()	Referencia de autor de BioRS en formato "<apellido>, <inic>". "<apellido>" es el apellido del autor. "<inic>" son las iniciales del autor, sin puntos. Se acepta un espacio en blanco entre la coma y las iniciales. De modo alternativo, <apellido> se puede especificar solo, sin la coma ni las iniciales.
Fecha	VARCHAR(), CHAR(), DATE o TIMESTAMP	S es una cadena de caracteres, fecha con formato de DB2, aaaa/mm/dd.
Número	VARCHAR() o CHAR(), INTEGER, SMALLINT, BIGINT REAL, DOUBLE, DECIMAL	Números con formato de DB2.
Consulta	VARCHAR() o CHAR()	Término de texto de BioRS.

Las demás combinaciones de columnas de término de búsqueda de tipo de datos y argumentos de términos de consulta de BioRS generan el mensaje de error SQL30090N ("Operación no válida para el entorno de ejecución de aplicación"). Únicamente puede utilizar las combinaciones que se muestran en la Tabla 2.

El argumento de término de consulta para columnas de término de búsqueda del tipo de datos Text, Author y Reference deben coincidir con un patrón de lenguaje de consulta de BioRS. En BioRS, los argumentos de término de consulta pueden consistir en cadenas alfanuméricas y caracteres comodín. La función BIORS.CONTAINS soporta dos caracteres comodín: **?** (signo de interrogación) y * (asterisco).

El carácter comodín **?** coincide con un único carácter. Por ejemplo, el predicado `BioRS.CONTAINS (descripción, 'bacteri?')=1` coincide con el término bacteria pero no con el término bacteriano.

El carácter comodín * coincide con cero o más caracteres. Por ejemplo, el predicado `BioRS.CONTAINS (descripción, 'bacteri*')=1` coincide con los términos bacteri, bacteria y bacteriano.

Para obtener información detallada sobre patrones de lenguaje de consulta de BioRS, consulte la documentación de BioRS.

La función BIORS.CONTAINS pude especificarse para todos los tipos de columna de BioRS.

Las funciones personalizadas BIORS.CONTAINS_GE y BIORS.CONTAINS_LE sólo se pueden especificar para columnas cuyo tipo de datos implícito de BioRS sea Number o Date. La función BIORS.CONTAINS_GE selecciona filas en las que la columna contiene un valor mayor o igual que el valor representado por el argumento de término de consulta. La función BIORS.CONTAINS_LE selecciona filas en las que la columna contiene un valor menor o igual que el valor representado por el argumento de término de consulta.

Las funciones BIORS.CONTAINS, BIORS.CONTAINS_GE y BIORS.CONTAINS_LE devuelven un resultado entero. Cuando se utiliza una de estas tres funciones CONTAINS en un predicado, el valor de retorno debe compararse con el valor 1 utilizando los operadores = o <>. Por ejemplo:

```
SELECT * FROM s.MySP WHERE BIORS.CONTAINS (s.AllText, 'muscus') = 1;
```

Una expresión con el formato `NOT (BioRS.Contains (col,value) = 1)` equivale a `BioRS.CONTAINS (col,value) <> 1`.

Puede ejecutar consultas que, de otra manera, no serían posibles, emitiendo la función BIORS.SEARCH_TERM. Puede utilizar esta función para especificar un término de búsqueda utilizando el formato de BioRS. La función BIORS.SEARCH_TERM requiere dos argumentos. El primer argumento es una referencia a la columna _ID_ del apodo al cual debe aplicarse el término. El segundo argumento es una cadena de caracteres que contiene el término sin ningún nombre de banco de datos.

En el ejemplo siguiente se seleccionan todas las columnas para entradas en el banco de datos MyEMBL donde el elemento SeqLength contiene un valor mayor o igual que 100.

```
SELECT * FROM MyEMBL s WHERE
 BIORS.SEARCH_TERM (s.ID, '[SeqLength GREATER number:100;]') = 1;
```

En el ejemplo siguiente se selecciona la columna MolWeight del apodo Swiss donde el valor del elemento MolWeight es mayor o igual que 100368.

```
SELECT s.molweight FROM Swiss s WHERE
   BIORS.SEARCH_TERM (s.ID, '[MolWeight GREATER number:100368;]') = 1;
```

Si especifica la función BIORS.SEARCH_TERM, no puede utilizar ninguna otra función personalizada en una consulta. Sin embargo, puede utilizar cualquier combinación de las funciones BIORS.CONTAINS, BIORS.CONTAINS_GE y BIORS.CONTAINS_LE en la misma consulta.

Conceptos relacionados:

- "Pushdown analysis" en la publicación *Federated Systems Guide*
- "Directrices para optimizar el rendimiento del reiniciador de BioRS" en la página 14
- "Predicados de unión de igualdad para el reiniciador de BioRS" en la página 19

Tareas relacionadas:

- "Registro de funciones personalizadas para el reiniciador de BioRS" en la página 4

Información relacionada:

- "Reiniciador de BioRS - Consultas de ejemplo" en la página 21
- "Tabla de funciones personalizadas - Reiniciador de BioRS" en la página 32

Predicados de unión de igualdad para el reiniciador de BioRS

Puede especificar predicados para el motor BioRS utilizando las cuatro funciones personalizadas de BioRS, con una excepción. La excepción se produce cuando se efectúan operaciones de unión de igualdad durante una consulta. Una operación de *unión* implica la recuperación de datos de dos o más tablas basada en valores de columna coincidentes. Una *unión de igualdad* es una operación de igualdad en la que la condición de unión tiene el formato expresión = expresión. Para consultas de BioRS, los términos de la unión de igualdad deben contener el elemento _ID_ de un banco de datos y un elemento de tipo Consulta de otro banco.

Ejemplo:

Este ejemplo muestra las definiciones de apodo de ejemplo y una consulta de unión de igualdad en la que se utilizan los apodos de ejemplo.

Supongamos que desea consultar dos bancos de datos de BioRS, SwissProt y SwissProt.features. El banco de datos SwissProt.features es hijo del banco de datos SwissProt y contiene un elemento llamado Parent. El elemento Parent contiene referencias a entradas identificadas por el elemento _ID_ de SwissProt. Se registrarán dos definiciones de apodo para los dos bancos de datos.

Definición de apodo 1:

```
CREATE NICKNAME tc600sprot (
   ID             VARCHAR (32) OPTIONS (ELEMENT_NAME '_ID_'),
   AllText        VARCHAR (128),
   EntryDate      VARCHAR (128),
   Update         VARCHAR (128),
   Description    VARCHAR (1200),
   Crossreference VARCHAR (32),
   Authors        VARCHAR (256),
   Journal        VARCHAR (256),
   JournalIssue   VARCHAR (64) OPTIONS (IS_INDEXED 'N'),
   PublicationYear VARCHAR (1024),
   Gene            VARCHAR (20) OPTIONS (IS_INDEXED 'Y'),
   Remarks         VARCHAR (1200),
   RemarkType      CHAR (20),
   CatalyticActivity VARCHAR (20),
   CoFactor        VARCHAR (64),
   Disease         VARCHAR (128),
   Function        VARCHAR (128),
   Pathway         VARCHAR (128),
   Similarity      VARCHAR (128),
   Complex         VARCHAR (64),
   FtKey           VARCHAR (32),
   FtDescription   VARCHAR (128),
   FtLength        VARCHAR (256),
   MolWeight       VARCHAR (64),
   ProteinLen      VARCHAR (32) OPTIONS (ELEMENT_NAME 'Protein_length'),
   Sequence        CLOB,
   AccNumber       VARCHAR (32),
   Taxonomy        VARCHAR (128),
   Organelle       VARCHAR (128),
   Organism        VARCHAR (128),
   Keywords        VARCHAR (1200),
   Localization    VARCHAR (128),
   FtKey_count     VARCHAR (32)) FOR SERVER biors_server_600
      OPTIONS (REMOTE_OBJECT 'SwissProt');
```

Definición de apodo 2:

```
CREATE NICKNAME tc600feat (
  ID     VARCHAR (32) OPTIONS (ELEMENT_NAME '_ID_'),
  AllText VARCHAR (1200),
  FtKey   VARCHAR (32),
  FtLength VARCHAR (64),
  FtDescription VARCHAR (128),
  Parent        VARCHAR (32) OPTIONS (REFERENCED_OBJECT 'SwissProt'))
    FOR SERVER biors_server_600 OPTIONS (REMOTE_OBJECT 'SwissProt.features');
```

La siguiente consulta hace referencia a ambos apodos de una unión de igualdad:

```
SELECT s.ID, f.ID, f.FtKey FROM tc600sprot s, tc600feat f
   WHERE BioRS.CONTAINS (s.AllText, 'anopheles') = 1
    AND BioRS.CONTAINS (s.PublicationYear, 1997) = 1
 AND BioRS.CONTAINS (f.FtKey, 'signal') = 1
 AND f.Parent = s.ID;
```

En la consulta previa, se aplican dos predicados al apodo tc600sprot (banco de datos SwissProt). Estos dos predicados filtran las filas que contienen el término ″anopheles″ y tienen el año de publicación 1997. Un predicado se aplica al apodo tc600feat (banco de datos SwissProt.features), que filtra las filas cuyo elemento FtKey contiene el término ″signal″. Los dos apodos se unen utilizando el término f.Parent = s.ID.

El conjunto resultante final sólo contiene las filas que cumplen este criterio y cuyas entradas de ″features″ hacen referencia a una entrada coincidente de un banco de datos SwissProt.

Conceptos relacionados:

- “Directrices para optimizar el rendimiento del reiniciador de BioRS” en la página 14

Información relacionada:

- “Funciones personalizadas y consultas de BioRS” en la página 16
- “Reiniciador de BioRS - Consultas de ejemplo” en la página 21

Reiniciador de BioRS - Consultas de ejemplo

Este tema proporciona varias consultas de ejemplo que utilizan los apodos swiss y swissft.

El apodo swiss se ha registrado con la siguiente sentencia CREATE NICKNAME:

```
CREATE NICKNAME swiss
  (
  ID                 CHAR (30) OPTIONS (ELEMENT_NAME '_ID_'),
  EntryDate          VARCHAR (15),
  Update             CLOB (15),
  Description        CLOB (15),
  Crossreference     CLOB (15),
  Authors            CLOB (15),
  Journal            VARCHAR (15),
  JournalIssue       VARCHAR (15),
  PublicationYear    CLOB (15),
  PublicationTitle   CLOB (15),
  Gene               CLOB (15),
  Remarks            CLOB (15),
  RemarkType         VARCHAR (15),
  CatalyticActivity  VARCHAR (15),
  CoFactor           VARCHAR (15),
```

```
  Disease            VARCHAR (15),
  Function           CLOB (15),
  Pathway            VARCHAR (15),
  Similarity         CLOB (15),
  Complex            VARCHAR (15),
  FtKey              VARCHAR (15),
  FtDescription      CLOB (15),
  FtLength           VARCHAR (15),
  MolWeight          CHAR (15),
  Protein_Length     VARCHAR (15),
  Sequence           CLOB (15),
  AccNumber          VARCHAR (15),
  Taxonomy           CLOB (15),
  Organelle          VARCHAR (15),
  Organism           VARCHAR (15),
  Keywords           VARCHAR (15),
  Localization       VARCHAR (15),
  FtKey_count        VARCHAR (15),
  AllText            CLOB (15)
  )
    FOR SERVER biors_server
      OPTIONS (REMOTE_OBJECT 'swissprot');
```

El apodo swissft se ha registrado con la siguiente sentencia CREATE NICKNAME:

```
CREATE NICKNAME swissft
  (
  ID                 VARCHAR (30) OPTIONS (ELEMENT_NAME '_ID_'),
  FtKey              VARCHAR (15),
  FtLength           VARCHAR (15),
  FtDescription      VARCHAR (15),
  Parent             VARCHAR (30) OPTIONS (REFERENCED_OBJECT 'swissprot'),
  AllText            CLOB (15)
  )
    FOR SERVER biors_server
      OPTIONS (REMOTE_OBJECT 'swissprot.features');
```

Las consultas y resultados de la Tabla 3 ilustran cómo puede estructurar las consultas para optimizar la carga de trabajo entre el sistema federado y el servidor BioRS.

Tabla 3. Ejemplos de consultas diferentes que generan resultados idénticos

Consulta	Resultado
select s.id from Swiss s where biors.CONTAINS(s.id, '100K_RAT') = 1 fetch first 3 rows only	ID---------- 100K_RAT 1 registro seleccionado.
select s.id from Swiss s where s.id LIKE '%100K_RAT%' fetch first 3 rows only	ID---------- 100K_RAT 1 registro seleccionado.

Ambas consultas de la Tabla 3 en la página 22 generan los mismos resultados. Sin embargo, la primera consulta se ejecutará con más rapidez que la segunda consulta. La primera consulta utiliza la función BIORS.CONTAINS para especificar el predicado de entrada. Como resultado, BioRS selecciona los datos del banco de datos swissprot y, a continuación, pasa los datos seleccionados a DB2 Information Integrator. En la segunda consulta, el predicado de entrada LIKE se especifica directamente en el apodo Swiss. Como resultado, BioRS transfiere todo el banco de datos swissprot a DB2 Information Integrator. Después de transferir el contenido del banco de datos, DB2 Information Integrator selecciona los datos.

Las consultas y los resultados de la Tabla 4 muestran la utilización de caracteres comodín en la función BIORS.CONTAINS. Todos los resultados de consultas de la Tabla 4 son idénticos, aunque se utilizan caracteres comodín diferentes.

Tabla 4. Consultas de ejemplo que utilizan caracteres comodín en la función BIORS.CONTAINS

Consulta	Resultado
select s.crossreference from Swiss s where biors.CONTAINS(s.crossreference, 'MEDLINE') = 1 fetch first 3 rows only	CROSSREFERENCE --------------- NCBI_TaxID=1011 NCBI_TaxID=5875 NCBI_TaxID=4081 3 registros seleccionados.
select s.crossreference from Swiss s where biors.CONTAINS(s.crossreference, '?ED?IN?') = 1 fetch first 3 rows only	CROSSREFERENCE --------------- NCBI_TaxID=1011 NCBI_TaxID=5875 NCBI_TaxID=4081 3 registros seleccionados.
select s.crossreference from Swiss s where biors.CONTAINS(s.crossreference, '*D*N*') = 1 fetch first 3 rows only	CROSSREFERENCE --------------- NCBI_TaxID=1011 NCBI_TaxID=5875 NCBI_TaxID=4081 3 registros seleccionados.

Las consultas y resultados de la Tabla 5 en la página 24 muestran cómo acceder a la información de elementos del tipo de datos Author de BioRS con la función BIORS.CONTAINS.

La sintaxis de todas las consultas de la Tabla 5 es casi idéntica. La única diferencia es la presencia o la ausencia de la primera inicial en el término de consulta y la cantidad de espacio entre el nombre de pila y la última inicial.

Tabla 5. Consultas de ejemplo que acceden a columnas del tipo de datos Author de BioRS

Consulta	Resultado
select s.authors from Swiss s where biors.CONTAINS(s.authors, 'Mueller') = 1 fetch first 3 rows only	AUTHORS --------------- Mueller D. Rehb Mayer K.F.X. Sc Zemmour J. Litt 3 registros seleccionados.
select s.authors from Swiss s where biors.CONTAINS(s.authors, 'Mueller,D') = 1 fetch first 3 rows only	AUTHORS --------------- 0 registros seleccionados.
select s.authors from Swiss s where biors.CONTAINS(s.authors, 'Mueller ,D') = 1 fetch first 3 rows only	AUTHORS --------------- 0 registros seleccionados.
select s.authors from Swiss s where biors.CONTAINS(s.authors, 'Mueller, D') = 1 fetch first 3 rows only	AUTHORS --------------- Mueller D. Rehb Zou P.J. Borovo Davies J.D. Mue 3 registros seleccionados.

Las consultas y resultados de la Tabla 6 en la página 25 ilustran cómo acceder a la información de los elementos del tipo Date de BioRS Date con la función BIORS.CONTAINS.

Cuando un campo del tipo Date de BioRS contiene una secuencia de fechas, los resultados pueden contener información adicional, tal como se muestra en el segundo ejemplo de la Tabla 6 en la página 25. Los elementos del tipo de datos Numeric (Date y Number) de BioRS pueden contener varios valores. Por lo tanto, los resultados de las consultas ejecutadas en elementos Date o Number de BioRS también pueden contener varios valores. Los múltiples valores siempre se separan mediante espacios.

Tabla 6. Consultas de ejemplo que acceden a columnas del tipo de datos Date de BioRS

Consulta	Resultado
select e.entrydate from embl e where biors.CONTAINS(e.entrydate, date('11/01/1997')) = 1 fetch first 3 rows only	ENTRYDATE --------------- 01-NOV-1997 01-NOV-1997 01-NOV-1997 3 registros seleccionados.
select g.update from gen g where biors.CONTAINS(g.update, date('11/01/1997')) = 1 fetch first 3 rows only	UPDATE --------------- 01-NOV-1997 11- 01-NOV-1997 12- 01-NOV-1997 06- 3 registros seleccionados.

Las consultas y resultados de la Tabla 7 muestran cómo utilizar las funciones BIORS.CONTAINS_LE y BIORS.CONTAINS_GE.

Tabla 7. Consultas de ejemplo que utilizan las funciones BIORS.CONTAINS_LE y BIORS.CONTAINS_GE

Consulta	Resultado
select s.molweight from Swiss s where biors.CONTAINS_LE(s.molweight, 100368) = 1 fetch first 3 rows only	MOLWEIGHT --------------- 100368 10576 8523 3 registros seleccionados.
select s.molweight from Swiss s where biors.CONTAINS_GE(s.molweight, 100368) = 1 fetch first 3 rows only	MOLWEIGHT --------------- 100368 103625 132801 3 registros seleccionados.
select s.journalissue from Swiss s where biors.CONTAINS_GE(s.journalissue, 172) = 1 fetch first 3 rows only	JOURNALISSUE --------------- 172 21 242 196 3 registros seleccionados.

Las consultas y resultados de la Tabla 8 muestran cómo utilizar la función BIORS.SEARCH_TERM para especificar un término de búsqueda utilizando el formato de BioRS.

Tabla 8. Consultas de ejemplo que utilizan la función BIORS.SEARCH_TERM

Consulta	Resultado
select s.publicationyear from Swiss s where biors.SEARCH_TERM (s.id, '[PublicationYear EQ number:1997;]')=1 fetch first 10 rows only	PUBLICATIONYEAR --------------- 1997 1997 2000 1988 1991 1997 1994 1997 1997 1998 1994 1995 1997 1997 1999 1997 1994 1994 1995 1993 1992 1997 10 registros seleccionados.
select s.molweight from Swiss s where biors.SEARCH_TERM (s.id, '[MolWeight EQ number:100368;]') = 1 fetch first 10 rows only	MOLWEIGHT --------------- 100368 100368 2 registros seleccionados.
select s.molweight from Swiss s where biors.SEARCH_TERM (s.id, '[MolWeight GREATER number:100368;]') = 1 fetch first 10 rows only	MOLWEIGHT --------------- 100368 103625 132801 194328 130277 287022 289130 135502 112715 112599 10 registros seleccionados.

La consulta siguiente muestra cómo utilizar predicados relacionales para formar una unión de igualdad entre dos bancos de datos que tienen una relación padre-hijo:

```
select s.id, f.id, f.parent from Swiss s, Swissft f
where (f.parent = s.id) fetch first 10 rows only
```

Los resultados de la consulta son los siguientes:

```
ID                   ID                 PARENT
-------------------- ------------------ ------------------------------
100K_RAT             100K_RAT.1         swissprot:100K_RAT
100K_RAT             100K_RAT.2         swissprot:100K_RAT
100K_RAT             100K_RAT.3         swissprot:100K_RAT
100K_RAT             100K_RAT.4         swissprot:100K_RAT
100K_RAT             100K_RAT.5         swissprot:100K_RAT
100K_RAT             100K_RAT.6         swissprot:100K_RAT
100K_RAT             100K_RAT.7         swissprot:100K_RAT
100K_RAT             100K_RAT.8         swissprot:100K_RAT
100K_RAT             100K_RAT.9         swissprot:100K_RAT
104K_THEPA           104K_THEPA.1       swissprot:104K_THEPA

10 registros seleccionados.
```

En los resultados de la consulta previa, el registro 100K_RAT es padre de nueve registros hijo (del 100K_RAT.1 al 100K_RAT.9).

Conceptos relacionados:

- "Directrices para optimizar el rendimiento del reiniciador de BioRS" en la página 14
- "Predicados de unión de igualdad para el reiniciador de BioRS" en la página 19

Información relacionada:

- "Funciones personalizadas y consultas de BioRS" en la página 16
- "Sentencia CREATE NICKNAME - Ejemplos para reiniciador de BioRS" en la página 11
- "Sintaxis de la sentencia CREATE NICKNAME - Reiniciador de BioRS" en la página 38

Información estadística de BioRS

En un sistema federado, la base de datos federada se basa en estadísticas de catálogos para que los objetos con apodos optimicen el proceso de las consultas. Estas estadísticas se recuperan de fuentes de datos de BioRS al crear un apodo utilizando la sentencia CREATE NICKNAME. La base de datos federada verifica la presencia del objeto en la fuente de datos y luego intenta recopilar datos estadísticos existentes de la fuente de datos. La información se lee de los catálogos de la fuente de datos y se coloca en el catálogo del sistema de bases de datos federadas de DB2® en el servidor federado.

Para fuentes de datos de BioRS, la información estadística crítica incluye:

- La cardinalidad de un apodo. Para fuentes de datos de BioRS, la cardinalidad de apodo equivale al número de entradas del banco de datos de BioRS correspondiente.

- La cardinalidad de la columna que corresponde al elemento BioRS _ID_. La cardinalidad de esta columna debe coincidir con la cardinalidad del apodo en el que se hace referencia a la columna.
- La cardinalidad de todas las columnas que el reiniciador de BioRS puede que necesite utilizar.

Debe mantener las estadísticas actuales sobre las fuentes de datos de BioRS para optimizar el rendimiento del reiniciador de BioRS. Si los datos estadísticos o las características estructurales para un objeto remoto con los que está definido un apodo cambian, debe actualizar las estadísticas de cardinalidad correspondientes en el sistema federado. Las estadísticas de cardinalidad se guardan en la vista de catálogo SYSSTAT.TABLES y en la vista de catálogo SYSSTAT.COLUMNS.

Realice las tareas siguientes para mantener las estadísticas de cardinalidad de BioRS en el sistema federado:

1. Determine las estadísticas de cardinalidad del apodo requerido, si es necesario.
2. Actualice las estadísticas de cardinalidad apropiadas en la vista de catálogo o vistas de catálogo requeridas.

Conceptos relacionados:

- "Tuning query processing" en la publicación *Federated Systems Guide*

Tareas relacionadas:

- "Determinación de las estadísticas de cardinalidad de banco de datos de BioRS" en la página 28
- "Actualización de las estadísticas de cardinalidad de apodo de BioRS" en la página 29
- "Actualización de las estadísticas de cardinalidad de columnas de BioRS" en la página 12
- "Actualización de la cardinalidad de columnas _ID_ de BioRS" en la página 30

Determinación de las estadísticas de cardinalidad de banco de datos de BioRS

Para poder actualizar las estadísticas de apodo o actualizar la cardinalidad de la columna que corresponde al elemento _ID_ de BioRS, primero debe determinar las estadísticas de cardinalidad de banco de datos de BioRS .

Procedimiento:

Para determinar las estadísticas de cardinalidad para un banco de datos específico de BioRS, utilice el programa de utilidad admin_find o

www_find.cgi de BioRS. Especifique la opción `-c` (cardinalidad). Para obtener más información sobre estos dos programas de utilidad de BioRS, consulte la documentación de BioRS.

Conceptos relacionados:

- "Información estadística de BioRS" en la página 27

Tareas relacionadas:

- "Actualización de las estadísticas de cardinalidad de apodo de BioRS" en la página 29
- "Actualización de las estadísticas de cardinalidad de columnas de BioRS" en la página 12
- "Actualización de la cardinalidad de columnas _ID_ de BioRS" en la página 30

Actualización de las estadísticas de cardinalidad de apodo de BioRS

Debe actualizar las estadísticas de cardinalidad de apodo de BioRS cuando el contenido de un banco de datos de BioRS para el que crea un apodo cambia de modo significativo. El mantenimiento de las estadísticas de cardinalidad correctas para los apodos permite que el optimizador y el reiniciador de BioRS elijan el mejor plan de acceso a datos de rendimiento.

Para actualizar las estadísticas de cardinalidad de apodo de BioRS, debe modificar la vista de catálogo SYSSTAT.TABLES con el número de cardinalidad correcto.

Requisitos previos:

Debe determinar el número de cardinalidad del banco de datos de BioRS que corresponde al apodo cuyas estadísticas desea actualizar.

Procedimiento:

Emita una sentencia UPDATE utilizando la sintaxis siguiente:

```
UPDATE sysstat.tables SET card=<cardinalidad>
WHERE tabschema=<esquema-apodo>
AND tabname=<nombre-apodo>
```

- <*cardinalidad*> es el número de cardinalidad de banco de datos de BioRS que corresponde al apodo cuyas estadísticas desea actualizar.
- <*esquema-apodo*> es el nombre del esquema que está asociado con el apodo cuyas estadísticas desea actualizar.
- <*nombre-apodo*> es el nombre del apodo cuyas estadísticas desea actualizar.

Conceptos relacionados:

- "Información estadística de BioRS" en la página 27

Tareas relacionadas:

- "Determinación de las estadísticas de cardinalidad de banco de datos de BioRS" en la página 28
- "Actualización de las estadísticas de cardinalidad de columnas de BioRS" en la página 12
- "Actualización de la cardinalidad de columnas _ID_ de BioRS" en la página 30

Actualización de la cardinalidad de columnas _ID_ de BioRS

El mantenimiento de las estadísticas de cardinalidad correcta para la columna que se correlaciona con el elemento _ID_ de BioRS permite que el optimizador y el reiniciador de BioRS elijan el mejor plan de acceso a datos de rendimiento.

Para actualizar el número de cardinalidad de la columna que se correlaciona con el elemento _ID_ de BioRS, debe modificar la vista de catálogo SYSSTAT.COLUMNS.

Requisitos previos:

Debe determinar el número de cardinalidad del banco de datos de BioRS que corresponde al apodo en el que se hace referencia a la columna. El número de cardinalidad de la columna que corresponde al elemento _ID_ de BioRS debe coincidir con la cardinalidad del apodo en el que se hace referencia a la columna.

Procedimiento:

Para actualizar las estadísticas de cardinalidad de columnas _ID_ de BioRS, emita una sentencia UPDATE utilizando la sintaxis siguiente:

```
UPDATE sysstat.columns SET colcard=<<cardinalidad>)
WHERE  tabschema=<esquema-apodo>
  AND tabname=<nombre-apodo>
  AND colname IN (SELECT colname FROM syscat.coloptions
WHERE                   tabschema=<nombre-apodo>
                  AND tabname=<nombre-apodo>
                  AND option='ELEMENT_NAME';
                  AND setting='_ID_')
```

- <*cardinalidad*> es el número de cardinalidad del banco de datos de BioRS que corresponde al apodo de la columna.

- *<esquema-apodo>* es el nombre del esquema que está asociado con el apodo de la columna.
- *<nombre-apodo>* es el nombre del apodo en el que se utiliza la columna.

Conceptos relacionados:

- "Información estadística de BioRS" en la página 27

Tareas relacionadas:

- "Determinación de las estadísticas de cardinalidad de banco de datos de BioRS" en la página 28
- "Actualización de las estadísticas de cardinalidad de apodo de BioRS" en la página 29
- "Actualización de las estadísticas de cardinalidad de columnas de BioRS" en la página 12

Elemento AllText de BioRS

Cada banco de datos del sistema BioRS contiene un elemento denominado AllText. BioRS crea automáticamente este elemento indexado para todos los bancos de datos.

El elemento AllText le permite buscar en todo el texto de una entrada, no sólo en elementos indexados específicos. Por ejemplo, la búsqueda del término `muscus` puede devolver entradas en la que la palabra `muscus` aparezca en el título, resumen, descripción u organismo.

Para utilizar el elemento AllText en una consulta de DB2 Information Integrator, debe correlacionar el elemento AllText con una columna de apodo. Tras correlacionar correctamente el elemento AllText con una columna de apodo, puede utilizar esta columna de apodo en una invocación de función personalizada CONTAINS.

Si se hace referencia a una columna que correlaciona el elemento AllText en la lista SELECT de una consulta, siempre se devolverá un valor NULL.

Tareas relacionadas:

- "Registro de apodos para las fuentes de datos de BioRS" en la página 9

Información relacionada:

- "Reiniciador de BioRS - Consultas de ejemplo" en la página 21

Consideraciones para modificar apodos - Reiniciador de BioRS

Puede modificar los apodos previamente registrados de BioRS utilizando la sentencia ALTER NICKNAME. Con la sentencia ALTER NICKNAME puede:

- Cambiar el nombre de una columna
- Cambiar el tipo de datos de una columna
- Añadir, cambiar o suprimir opciones para una columna

Restricciones:

No puede cambiar el nombre del banco de datos de BioRS al que hace referencia un apodo o utilizado por un apodo. Si el nombre de un banco de datos de BioRS que se utiliza en un apodo cambia, deberá descartar todo el apodo y volverlo a crear.

Si ha especificado la opción REMOTE_OBJECT, no puede cambiar ni suprimir el valor de la opción.

Si cambia el tipo de datos de una columna, el tipo de datos nuevo tiene que ser compatible con el tipo de datos del elemento de BioRS correspondiente.

Si cambia el nombre de elemento de una columna utilizando la opción ELEMENT_NAME, no se comprueba si el nombre nuevo es correcto. Una opción incorrecta puede dar como resultado errores cuando se hace referencia a la columna en una consulta.

Si efectúa cambios en la opción de columna IS_INDEXED, no se verifican los cambios con el servidor BioRS. Una opción incorrecta puede dar como resultado errores cuando se hace referencia a la columna en una consulta.

Información relacionada:

- "Sentencia ALTER NICKNAME" en la publicación *Consulta de SQL, Volumen 2*

Tabla de funciones personalizadas - Reiniciador de BioRS

La Tabla 9 en la página 33 proporciona ejemplos sobre cómo registrar cada una de las cuatro funciones personalizadas de BioRS.

Para ayudarle a registrar funciones personalizadas, se proporciona un archivo de ejemplo, create_function_mappings.ddl, en el directorio sqllib/samples/lifesci/biors. El archivo create_function_mappings.ddl contiene definiciones para cada función personalizada. Puede ejecutar este archivo DDL para registrar las funciones personalizadas para cada base de

datos de DB2 que tenga instalado el reiniciador de BioRS.

Tabla 9. Funciones personalizadas para el reiniciador de BioRS

Nombre de función	Descripción
CONTAINS (col VARCHAR(), term VARCHAR()), CONTAINS (col VARCHAR(), term CHAR()) CONTAINS (col VARCHAR(), term DATE) CONTAINS (col VARCHAR(), term TIMESTAMP)	Busca en una columna indexada utilizando la expresión proporcionada. **col** Columna indexada. **term** Término de búsqueda.
CONTAINS_LE (col VARCHAR(), term VARCHAR()), CONTAINS_LE (col VARCHAR(), term SMALLINT) CONTAINS_LE (col VARCHAR(), term BIGINT) CONTAINS_LE (col VARCHAR(), term DECIMAL) CONTAINS_LE (col VARCHAR(), term DOUBLE) CONTAINS_LE (col VARCHAR(), term REAL)	Busca en una columna indexada utilizando la expresión proporcionada. **col** Columna indexada. **term** Término de búsqueda.
CONTAINS_GE (col CHAR(), term CHAR()) CONTAINS_GE (col CHAR(), term DATE) CONTAINS_GE (col CHAR(), term TIMESTAMP) CONTAINS_GE (col CHAR(), term INTEGER) CONTAINS_GE (col CHAR(), term SMALLINT) CONTAINS_GE (col CLOB(), term DATE)	Busca en una columna indexada utilizando la expresión proporcionada. **col** Columna indexada. **term** Término de búsqueda.
SEARCH_TERM (col VARCHAR(), term VARCHAR()) SEARCH_TERM (col VARCHAR(), term CHAR()) SEARCH_TERM (col CHAR(), term VARCHAR()) SEARCH_TERM (col CHAR(), term CHAR())	Pasa un término de búsqueda de BioRS al motor de búsqueda BioRS. **col** Columna indexada. **term** Término de búsqueda.

Tareas relacionadas:

- "Registro de funciones personalizadas para el reiniciador de BioRS" en la página 4

Mensajes para el reiniciador de BioRS

Este tema explica los mensajes que puede recibir al trabajar con el reiniciador para BioRS. Si desea obtener más información acerca de los mensajes, consulte la publicación *Consulta de mensajes de DB2*.

Tabla 10. Mensajes emitidos por el reiniciador para BioRS

Código de error	Mensaje	Explicación
SQL0604N	El atributo de longitud, precisión o escala para una columna, tipo diferenciado, tipo estructurado, atributo de un tipo estructurado, función o correlación de tipo <elemento-datos> no es válido.	El tipo de datos para una columna de apodo no es compatible con el tipo del elemento de banco de datos implícito de BioRS. Compruebe el tipo de datos de la columna en la sentencia CREATE NICKNAME.
SQL0901N	La sentencia de SQL ha fallado debido a un error del sistema que no es grave. Las sentencias de SQL posteriores pueden procesarse. (Razón ″Error al crear el objeto del reiniciador.″)	Se ha producido un error al crear un nuevo objeto del reiniciador. Póngase en contacto con el Soporte de Software de IBM.
SQL0901N	La sentencia de SQL ha fallado debido a un error del sistema que no es grave. Las sentencias de SQL posteriores pueden procesarse. (Razón ″BioRS <punto-rastreo>/<código>.″)	Es un error interno. Póngase en contacto con el Soporte de Software de IBM.
SQL0901N	La sentencia de SQL ha fallado debido a un error del sistema que no es grave. Las sentencias de SQL posteriores pueden procesarse. (Razón ″Ha fallado la asignación de memoria: <punto-rastreo>.″)	Se ha producido un error al asignar la memoria. Asegúrese de que hay disponible suficiente memoria para el sistema principal del servidor federado y vuelva a someter la consulta. Si el problema se repite, póngase en contacto con el Soporte de Software de IBM.
SQL0901N	La sentencia de SQL ha fallado debido a un error del sistema que no es grave. Las sentencias de SQL posteriores pueden procesarse. (Razón ″sqlno_crule_save_plans[100]:rc(-214272209) Lista de planes vacía.″)	El programa optimizador y el reiniciador de BioRS no se han puesto de acuerdo sobre ningún plan para ejecutar la consulta. Simplifique la consulta y vuélvala a ejecutar.

Tabla 10. Mensajes emitidos por el reiniciador para BioRS (continuación)

Código de error	Mensaje	Explicación
SQL401N	Los tipos de datos de los operando para la operación ″=″ no son compatibles.	La consulta no es válida debido a que la expresión del lado derecho de un predicado de función personalizada tiene que ser un valor entero.
SQL1822N	Se ha recibido el código de error inesperado ″″ de la fuente de datos ″Reiniciador de BioRS.″ El texto y los símbolos asociados son ″El Banco de datos no se ha encontrado.″	El banco de datos de BioRS al que se hace referencia en una sentencia CREATE NICKNAME no se ha encontrado en el servidor BioRS. Compruebe la sentencia CREATE NICKNAME y asegúrese de que el nombre del banco de datos al que se hace referencia sea correcto.
SQL1822N	Se ha recibido el código de error inesperado ″″ de la fuente de datos ″Reiniciador de BioRS.″ El texto asociado y los símbolos asociados son ″La conexión ha excedido el tiempo de espera.″	El servidor BioRS no ha respondido a una petición de comunicaciones dentro del período especificado por la opción TIMEOUT.
SQL1822N	Se ha recibido el código de error inesperado ″<punto_rastreo>″ de la fuente de datos ″Reiniciador de BioRS.″ El texto y los símbolos asociados son ″Error al leer desde el servidor.″	Se ha producido un error de comunicaciones al leer datos desde el servidor BioRS. El valor del código de error <punto_rastreo> puede proporcionar más información sobre el error.
SQL1822N	Se ha recibido el código de error inesperado ″<punto_rastreo>″ de la fuente de datos ″Reiniciador de BioRS.″ El texto y los símbolos asociados son ″El sistema principal no se ha encontrado.″	No se ha encontrado el sistema principal del servidor BioRS identificado en la opción de servidor HOST. Compruebe la sentencia CREATE SERVER y asegúrese de que el valor de la opción de servidor HOST sea correcta.
SQL1822N	Se ha recibido el código de error inesperado ″<punto_rastreo>″ de la fuente de datos ″Reiniciador de BioRS.″ El texto asociado y los símbolos son ″No puede conectarse al servidor.″	El reiniciador no se ha podido conectar al servidor identificado en la opción de servidor HOST. El valor del código de error <punto_rastreo> puede proporcionar más información sobre el error.

Tabla 10. Mensajes emitidos por el reiniciador para BioRS (continuación)

Código de error	Mensaje	Explicación
SQL1822N	Se ha recibido el código de error inesperado ″<punto_rastreo>″ de la fuente de datos ″Reiniciador de BioRS.″ El texto y los símbolos asociados son ″No puede crearse el socket TCPIP.″	El reiniciador no ha podido crear un socket TCPIP. El valor del código de error <punto_rastreo> puede proporcionar más información sobre el código de error.
SQL1822N	Se ha recibido el código de error inesperado ″<punto_rastreo>″ de la fuente de datos ″Reiniciador de BioRS.″ El texto y los símbolos asociados son ″Error al enviar al servidor.″	El reiniciador no ha podido enviar una petición al servidor BioRS. El valor del código de error <punto_rastreo> puede proporcionar más información sobre el error.
SQL30090N	La operación no es válida para el entorno de ejecución de la aplicación. Código de razón = ″No se puede cambiar la sensibilidad a las mayúsculas-minúsculas del servidor.″	No se puede cambiar el valor de la opción de servidor CASE_SENSITIVE con sentencias de SQL. Para cambiar el valor de esta opción, debe descartar el servidor. A continuación, debe volver a crear el servidor utilizando la sentencia CREATE SERVER y especificar el valor correcto para la opción CASE_SENSITIVE.
SQL30090N	La operación no es válida para el entorno de ejecución de la aplicación. Código de razón = ″Múltiples uniones entre dos apodos.″	La consulta no es válida debido a que sólo se permite un predicado de unión entre dos apodos.
SQL30090N	La operación no es válida para el entorno de ejecución de la aplicación. Código de razón = ″El lado derecho del predicado de la función tiene que ser una constante.″	La consulta no es válida debido a que la expresión del lado derecho de un predicado de función personalizada tiene que ser una constante.
SQL30090N	La operación no es válida para el entorno de ejecución de la aplicación. Código de razón = ″Arg 1 de la función personalizada no es una columna.″	La consulta no es válida debido a que el primer argumento de una función personalizada tiene que hacer referencia a una columna de un apodo de BioRS.

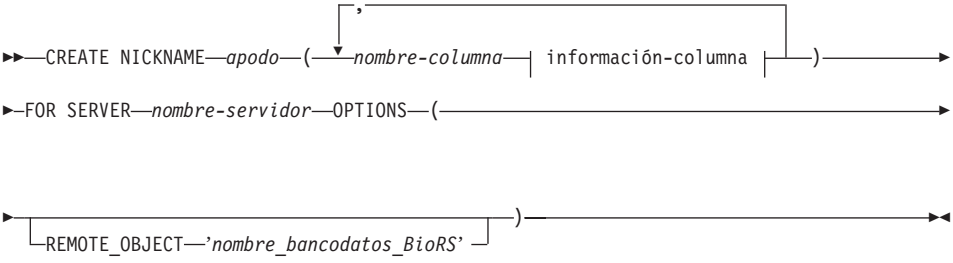
Tabla 10. Mensajes emitidos por el reiniciador para BioRS (continuación)

Código de error	Mensaje	Explicación
SQL30090N	La operación no es válida para el entorno de ejecución de la aplicación. Código de razón = "Arg 1 de la función CONTAINS no está indexado."	La consulta no es válida. La columna a la que se hace referencia en el primer argumento de una función BIORS.CONTAINS, BIORS.CONTAINS_LE o BIORS.CONTAINS_GE tiene que ser una columna indexada.
SQL30090N	La operación no es válida para el entorno de ejecución de la aplicación. Código de razón = "Tipo incorrecto para el arg1 de la función <nombre-función>."	La consulta no es válida. La columna a la que se hace referencia en el primer argumento de una función BIORS.CONTAINS, BIORS.CONTAINS_LE o BIORS.CONTAINS_GE no es del tipo de datos correcto.
SQL30090N	La operación no es válida para el entorno de ejecución de la aplicación. Código de razón = "Arg 1 de SEARCH_TERM no es la columna _ID_."	La consulta no es válida. La columna a la que se hace referencia en el primer argumento de una función SEARCH_TERM no está correlacionada con un elemento _ID_ de BioRS.
SQL30090N	La operación no es válida para el entorno de ejecución de la aplicación. Código de razón = "El parámetro de vinculación no puede ser NULL."	Un valor de columna o de variable de lenguaje principal al que se hacía referencia en el segundo argumento de una función BIORS.CONTAINS era NULL. El reiniciador de BioRS no puede procesar valores nulos.
SQL30090N	La operación no es válida para el entorno de ejecución de la aplicación. Código de razón = "No se puede convertir el valor en un literal de BioRS."	Se ha sometido un valor al reiniciador en un literal, columna o variable del lenguaje principal, que no se ha podido convertir en un literal de BioRS válido.
SQL30090N	La operación no es válida para el entorno de ejecución de la aplicación. Código de razón = "No se puede cambiar la versión del servidor."	No se puede cambiar la versión del servidor con la sentencia ALTER SERVER. Para cambiar la versión del servidor, debe descartar el servidor. A continuación, debe crear de nuevo el servidor con la versión correcta utilizando la sentencia CREATE SERVER.

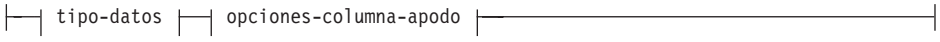
Tabla 10. Mensajes emitidos por el reiniciador para BioRS (continuación)

Código de error	Mensaje	Explicación
SQL30090N	La operación no es válida para el entorno de ejecución de la aplicación. Código de razón = "Tipo incorrecto de arg2 de la función <nombre-función>."	La consulta no es válida. La columna a la que se hace referencia en el segundo argumento de una función BIORS.CONTAINS, BIORS.CONTAINS_LE o BIORS.CONTAINS_GE no es del tipo de datos correcto.
SQL30090N	La operación no es válida para el entorno de ejecución de la aplicación. Código de razón = "El apodo no tiene columnas."	No se ha especificado ninguna declaración de columna en la sentencia CREATE NICKNAME. Las declaraciones de columna son necesarias para crear apodos.

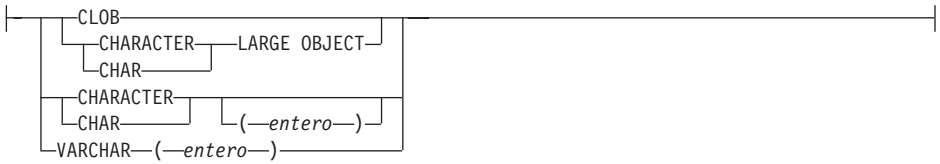
Sintaxis de la sentencia CREATE NICKNAME - Reiniciador de BioRS



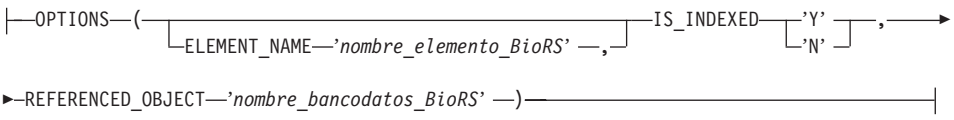
información-columna:



tipo-datos:



opciones-columna-apodo:



Opciones de columna de apodo

Los valores de opción de columna de apodo deben estar limitados por comillas simples.

ELEMENT_NAME
Especifica el nombre de elemento de BioRS. La sensibilidad a las mayúsculas y minúsculas de este nombre depende de la sensibilidad a las mayúsculas y minúsculas del servidor BioRS y del valor de la opción de servidor CASE_SENSITIVE. Sólo es necesario especificar el nombre de elemento de BioRS si es diferente del nombre de columna.

IS_INDEXED
Indica si la columna correspondiente está indexada (si se puede hacer referencia a la columna en un predicado). Los valores válidos son 'Y' y 'N'. El valor 'Y' sólo se puede especificar para columnas cuyo elemento correspondiente esté indexado por el servidor BioRS.

Cuando se crea un apodo, esta opción se añade automáticamente con el valor 'Y' a cualquier columna que corresponda a un elemento indexado de BioRS.

REFERENCED_OBJECT
Esta opción sólo es válida para columnas cuyo tipo de datos de BioRS sea Reference. Esta opción especifica el nombre del banco de datos de BioRS al que hace referencia la columna actual. La sensibilidad a las mayúsculas y minúsculas de este nombre depende de la sensibilidad a las mayúsculas y minúsculas del servidor BioRS y del valor de la opción de servidor CASE_SENSITIVE.

Opciones de apodo

Los valores de opción de apodo deben estar encerrados entre apóstrofos.

REMOTE_OBJECT
Especifica el nombre del banco de datos de BioRS que está asociado con el apodo. Este nombre determina el esquema y el banco de datos de BioRS para el apodo. Este nombre también especifica la relación del apodo con otros apodos. La sensibilidad a las mayúsculas y minúsculas de este nombre depende de la sensibilidad a las mayúsculas y minúsculas del servidor BioRS y del valor de la opción de servidor CASE_SENSITIVE.

Importante: no puede cambiar este nombre con la sentencia ALTER NICKNAME. Si el nombre del banco de datos de BioRS que se utiliza en esta opción cambia, deberá suprimir todo el apodo y volverlo a crear.

Tareas relacionadas:

- "Registro de apodos para las fuentes de datos de BioRS" en la página 9

Información relacionada:

- "Sentencia CREATE NICKNAME" en la publicación *Consulta de SQL, Volumen 2*
- "Sentencia CREATE NICKNAME - Ejemplos para reiniciador de BioRS" en la página 11

Opciones de la sentencia CREATE SERVER - Reiniciador de BioRS

Las opciones para la sentencia CREATE SERVER para BioRS son:

TYPE Especifica el tipo de servidor. El valor por omisión es `BioRS`. El valor por omisión es el único valor soportado para el reiniciador de BioRS. No es necesario especificar esta opción.

VERSION
Especifica la versión del servidor. El valor por omisión es `1.0`. El valor por omisión es el único valor soportado para el reiniciador de BioRS. No es necesario especificar esta opción.

NODE
Especifica el nombre del sistema principal en el que está disponible la herramienta de consulta BioRS. El valor por omisión es *localhost*. Es necesario especificar esta opción.

PORT Especifica el número de puerto que debe utilizarse para conectarse al servidor BioRS. El valor por omisión es 5014. Es necesario especificar esta opción.

TIMEOUT
Especifica la hora, en minutos, que el reiniciador de BioRS debe esperar una respuesta desde el servidor BioRS. El valor por omisión es 10. Es necesario especificar esta opción.

CASE_SENSITIVE
Especifica si el servidor BioRS trata los nombres con sensibilidad a las mayúsculas y minúsculas. Los valores válidos son 'Y' o 'N'. El valor por omisión es 'Y'.

En el producto BioRS, un parámetro de configuración controla la sensibilidad a las mayúsculas y minúsculas de los datos que se

guardan en la máquina del servidor BioRS. La opción CASE_SENSITIVE es la contraparte de DB2 Information Integrator del parámetro de configuración del sistema BioRS. Debe sincronizar los valores de configuración de la sensibilidad a las mayúsculas y minúsculas del servidor BioRS en el sistema BioRS y en DB2 Information Integrator. Si no mantiene sincronizados los valores de configuración de la sensibilidad a las mayúsculas y minúsculas entre BioRS y DB2 Information Integrator, se producirán errores cuando intente acceder a datos de BioRS mediante DB2 Information Integrator.

Importante: no puede cambiar ni suprimir la opción CASE_SENSITIVE después de crear un nuevo servidor BioRS en DB2 Information Integrator. Si necesita cambiar la opción CASE_SENSITIVE, debe descartar todo el servidor y volverlo a crear. Si descarta el servidor BioRS, también debe volver a crear todos los apodos correspondientes de BioRS. DB2 Information Integrator descarta automáticamente todos los apodos que corresponden a un servidor descartado.

Tareas relacionadas:

- "Registro del servidor para una fuente de datos de BioRS" en la página 7
- "Registro de apodos para las fuentes de datos de BioRS" en la página 9

Información relacionada:

- "Sentencia CREATE SERVER" en la publicación *Consulta de SQL, Volumen 2*
- "Sintaxis de la sentencia CREATE NICKNAME - Reiniciador de BioRS" en la página 38

Opciones de la sentencia CREATE USER MAPPING - Reiniciador de BioRS

GUEST

Especifica si deben realizarse operaciones bajo el mecanismo de autentificación de huésped de BioRS en el servidor BioRS. Los valores válidos son 'Y' o 'N'. El valor por omisión es 'Y'.

Si esta opción está establecida en 'Y', se utiliza autentificación de huésped para acceder al servidor BioRS para este usuario de DB2 Information Integrator.

Si esta opción está establecida en 'N', debe proporcionarse un ID y una contraseña de autorización de BioRS para acceder al servidor BioRS para este usuario de DB2 Information Integrator.

Si no se ha creado ninguna correlación de usuario, o si se ha creado una correlación de usuario sin especificar ninguna opción, se utiliza

autentificación de huésped para acceder al servidor BioRS para el usuario DB2 Information Integrator.

REMOTE_AUTHID
Especifica un ID de usuario que permite a este usuario de DB2 acceder a las fuentes de datos de BioRS. Este ID remoto debe tener el formato esperado por la aplicación de BioRS. Esta opción es necesaria si la opción GUEST está establecida en 'N'.

REMOTE_PASSWORD
Especifica la contraseña para este ID remoto. Esta opción es necesaria si la opción GUEST está establecida en 'N'.

Ejemplo:

La siguiente sentencia CREATE USER MAPPING correlaciona el usuario Charlie con el usuario Charlene en el servidor Biors_Server1.

```
CREATE USER MAPPING FOR Charlie SERVER Biors_Server1
OPTIONS(GUEST 'N' REMOTE_AUTHID 'Charlene', REMOTE_PASSWORD 'Charlene_pw');
```

Tareas relacionadas:

- "Registro de correlaciones de usuario para fuentes de datos de BioRS" en la página 7

Información relacionada:

- "Sentencia CREATE USER MAPPING" en la publicación *Consulta de SQL, Volumen 2*

Capítulo 2. Funciones definidas por el usuario de ciencias de la vida

Este capítulo explica qué son las funciones definidas por el usuario de ciencias de la vida, cómo añadirlas al sistema federado y cómo utilizarlas en las consultas.

Funciones definidas por el usuario de ciencias de la vida - visión general

Las funciones definidas por el usuario de ciencias de la vida proporcionan a los investigadores los algoritmos que utilizan con frecuencia para analizar datos. Las funciones están disponibles en las plataformas de 32–bits Windows® NT, AIX y Linux, excepto la función GeneWise que está disponible en las plataformas AIX® y Linux.

Las funciones definidas por el usuario de ciencias de la vida utilizan los códigos de una sola letra estándar y los códigos de ambigüedad de IUPAC-IUB para representar a los aminoácidos y los nucleótidos.

Para poder utilizar las funciones definidas por el usuario de ciencias de la vida, primero debe registrarlas. Una vez registradas, puede eliminarlas si es necesario.

Tareas relacionadas:

- "Registro de funciones definidas por el usuario de ciencias de la vida" en la página 45
- "Registro de funciones definidas por el usuario de ciencias de la vida" en la página 46

Información relacionada:

- "Funciones definidas por el usuario de ciencias de la vida por categoría funcional" en la página 44

Funciones definidas por el usuario de ciencias de la vida por categoría funcional

La Tabla 11 lista las funciones definidas por el usuario de ciencias de la vida por categoría funcional. También proporciona una breve descripción de cada categoría.

Tabla 11. Funciones definidas por el usuario de ciencias de la vida

Categoría funcional	Funciones definidas por el usuario	Descripción
Conversión inversa	LSPep2AmbNuc, LSPep2ProbNuc	Convierte una secuencia de aminoácidos en una secuencia de nucleótidos.
Análisis de línea de definición	LSDeflineParse	Analiza los elementos de una línea de definición, tales como los que devuelve el reiniciador de BLAST o los que están presentes en un archivo de datos en formato FASTA.
Coincidencia de patrones generalizada	LSPatternMatch, LSPrositePattern	Identifica las áreas de interés de una cadena determinada, tales como una secuencia de nucleótidos o de péptidos.
GeneWise	LSGeneWise	Alinea una secuencia de proteínas con una secuencia genómica.
Motivos	LSMultiMatch, LSMultiMatch3, LSBarCode	Busca coincidencias con patrones de secuencias de nucleótidos o de aminoácidos.
Inversión	LSRevNuc, LSRevPep, LSRevComp	Invierte una secuencia de nucleótidos o de aminoácidos.
Conversión	LSNuc2Pep, LSTransAllFrames	Convierte una secuencia de nucleótidos en una secuencia de péptidos.

Conceptos relacionados:

- "Funciones definidas por el usuario de ciencias de la vida - visión general" en la página 43

Tareas relacionadas:

- "Registro de funciones definidas por el usuario de ciencias de la vida" en la página 45

• "Registro de funciones definidas por el usuario de ciencias de la vida" en la página 46

Registro de funciones definidas por el usuario de ciencias de la vida

Para poder utilizar las funciones definidas por el usuario de las ciencias de la vida, primero debe registrarlas.

Procedimiento:

Para registrar las funciones definidas por el usuario de ciencias de la vida, utilice el mandato enable_LSFunctions en Windows NT y AIX en el directorio `sqllib/bin`.

La sintaxis del mandato enable_LSFunctions es:

```
enable_LSFunctions -n Nombredb -u IDusuario -p contraseña [-force]
```

Nombredb
El nombre de la base de datos en la que va a registrar las funciones.

IDusuario
Un ID de usuario válido para la base de datos.

contraseña
Una contraseña válida para el ID de usuario.

force Un distintivo que puede utilizar para eliminar las funciones y volverlas a registrar. Puede utilizar este distintivo para registrar las funciones de nuevo si se dañan o se eliminan accidentalmente.

El mandato crea las funciones con el nombre de esquema DB2LS.

El ejemplo siguiente muestra una salida de ejemplo para el mandato enable_LSFunctions:

```
C:> enable_LSFunctions -n test -u db2admin -p db2admin

(0) Life Sciences Functions were found
    -- Create Life Sciences Functions ...
    Create Life Sciences Functions Successfully.

*** Please allow a few seconds to clean up the system ......
```

El ejemplo siguiente muestra la salida para el mandato enable_LSFunctions cuando se utiliza el distintivo force y las funciones ya están registradas:

```
C:> enable_LSFunctions -n test -u db2admin -p db2admin -force

(21) Life Sciences Functions were found
```

```
Life Sciences functions already exist ...
Reinstall Life Sciences functions ...
    -- Drop Life Sciences Functions ...
    Drop Life Sciences Functions Successfully.
    -- Create Life Sciences Functions ...
    Create Life Sciences Functions Successfully.


*** Please allow a few seconds to clean up the system ......
```

Registro de funciones definidas por el usuario de ciencias de la vida

Si no desea tener más en el sistema funciones definidas por el usuario de ciencias de la vida, puede eliminarlas.

Procedimiento:

Para eliminar las funciones definidas por el usuario de ciencias de la vida, utilice el mandato disable_LSFunctions disponible en Windows NT y AIX en el directorio `sqllib/bin`.

La sintaxis del mandato disable_LSFunctions es:

```
disable_LSFunctions -n Nombredb -u IDusuario -p contraseña
```

Nombredb
El nombre de la base de datos de la que desea eliminar las funciones.

IDusuario
Un ID de usuario válido para la base de datos.

contraseña
Una contraseña válida para el ID de usuario.

El ejemplo siguiente muestra la salida del mandato disable_LSFunctions:

```
C:>disable_LSFunctions -n test -u db2admin -p db2admin
a
(21) Life Sciences Functions were found
    -- Drop Life Sciences Functions ...
    Drop Life Sciences Functions Successfully.


*** Please allow a few seconds to clean up the system ......
```

Funciones definidas por el usuario de conversión inversa

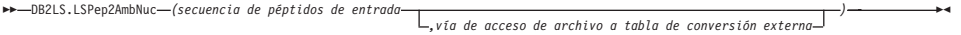
Utilice las funciones definidas por el usuario de conversión inversa para convertir una secuencia de péptidos en una secuencia de nucleótidos. La conversión inversa es una conversión hacia atrás.

Dado que la correlación de codones de tripletos de aminoácidos con nucleótidos es de uno a muchos, una conversión inversa produce dos resultados:

más ambiguo
Simple conversión de texto y apariencia. Utilice la función definida por el usuario LSPep2AmbNuc para realizar una conversión más ambigua.

más probable
Requiere información adicional de una tabla de frecuencia de codón. Utilice la función definida por el usuario LSPep2ProbNuc para realizar una conversión más probable.

Función definida por el usuario LSPep2AmbNuc



secuencia de péptidos de entrada
Una representación de cadena de caracteres válida que describe una secuencia de péptidos. Una representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 10890 bytes. Los datos de entrada utilizan los códigos de ambigüedad y símbolos de aminoácidos estándares.

vía de acceso a tabla de conversión externa
Si utiliza una tabla de conversión personalizada, incluya la información de vía de acceso de archivo para encontrar la tabla de conversión. El valor de cadena de la vía de acceso no debe tener más de 255 caracteres.

El nombre de esquema es DB2LS.

Utilice la función LSPep2AmbNuc para producir la secuencia de nucleótidos más ambigua, de acuerdo con una tabla de conversión, a partir de una secuencia de péptidos.

El resultado de la función es una cadena de caracteres con el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes. El resultado representa la secuencia de nucleótidos más ambigua, de acuerdo con una tabla de conversión, incorporada o especificada por el usuario.

Si no especifica ninguna tabla de conversión, por omisión la función utiliza la Tabla 12 en la página 48.

Tabla 12. Tabla de conversión por omisión

Símbolo de aminoácido	Abreviatura	Codón
A	Ala	GCX
B	Asx	RAY
C	Cys	TGY
D	Asp	GAY
E	Glu	GAR
F	Phe	TTY
G	Gly	GGX
H	His	CAY
I	Ile	ATH
K	Lys	AAR
L	Leu	YTX
M	Met	ATG
N	Asn	AAY
P	Pro	CCX
Q	Gln	CAR
R	Arg	MGX
S	Ser	WSX
T	Thr	ACX
V	Val	GTX
W	Trp	TGG
X	Xxx	XXX
Y	Tyr	TAY
Z	Glx	SAR
*	End	TRR

Información relacionada:

- "Función definida por el usuario LSPep2AmbNuc - mensajes de error" en la página 50
- "Función definida por el usuario LSPep2ProbNuc" en la página 51
- "Función definida por el usuario LSPep2AmbNuc - ejemplo" en la página 49

Función definida por el usuario LSPep2AmbNuc - ejemplo

Puede invocar la función con una sentencia values. La única entrada es una secuencia de péptidos, como en el ejemplo siguiente:

```
values db2ls.LSPep2AmbNuc('HR');
```

El ejemplo anterior convierte un péptido en un nucleótido utilizando las conversiones ambiguas y la tabla de conversión incorporada. El resultado de la sentencia anterior es una secuencia de nucleótidos creada a partir de los símbolos de aminoácidos estándares:

```
CAYMGX
```

El ejemplo siguiente convierte un péptido en un nucleótido utilizando las conversiones ambiguas y la tabla incorporada:

```
values db2ls.LSPep2AmbNuc('SRGFGFITYSHSSMIDEAQKSRPHKIDGRVVEPKRA');
```

El resultado de esta sentencia values es la siguiente secuencia de nucleótidos. (La secuencia se ha dividido para que quepa en la página.)

```
WSXMGXGGXTTYGGXTTYATHACXTAYWSXCAYWSXWSXATGATHGAYGARGCXCARA
ARWSXMGXCCXCAYAARATHGAYGGXMGXGTXGTXGARCCXAARMGXGCX
```

El ejemplo siguiente muestra la función aplicada a un conjunto de valores extraídos de una tabla o apodo:

```
SELECT DB2LS.LsPep2AmbNuc(peptide_seq) FROM table protein_table;
```

Los datos de la columna peptide_seq de la tabla protein_table son similares a los siguientes:

Tabla 13. Datos de la columna peptide_seq

peptide_seq
GIKEDTEEHHLRDYFE
QKYHTVNGHNCEVRKA
.....

El resultado de la sentencia select es:

```
GGXATHAARGARGAYACXGARGARCAYCAYYTXMGXGAYTAYTTYGAR
CARAARTAYCAYACXGTXAAYGGXCAYAAYTGYGARGTXMGXAARGCX
...
```

El ejemplo siguiente convierte un péptido en un nucleótido utilizando las conversiones ambiguas y una tabla definida por el usuario. Por lo general, las diferencias entre tablas de conversión son pequeñas. Es posible que tan solo

existan uno o dos símbolos que sean exclusivos. Esto puede deberse a que algunas especies tienen más codones o algunas tienen menos. Por ejemplo, el codón AGG está ausente en Drosófila.

```
values db2ls.LSPep2AmbNuc('RGNMGGGNYGNQNGGGNWNNG',
                          '\data\transl_table_06.txt')
```

Si suponemos que la tabla de conversión de entrada es para Drosófila, el resultado de la sentencia values se muestra en el ejemplo siguiente:

```
MGRGGXAAYATGGGXGGXGGXAAYTAYGGXAAYTARAAYGGXGGXGGXAAYTGGAAYAAYGGX
```

Información relacionada:

- "Función definida por el usuario LSPep2AmbNuc" en la página 47
- "Función definida por el usuario LSNuc2Pep – ejemplo" en la página 82

Función definida por el usuario LSPep2AmbNuc - mensajes de error

Tabla 14. Mensajes emitidos por la función definida por el usuario LSPep2AmbNuc

Código de error	Mensaje	Explicación
SQL0443N	La rutina ″DB2LS.LSPEP2AMBNUC″ (nombre específico ″LSPEP2AMBNUC″) ha devuelto un error SQLSTATE con el texto de diagnóstico ″Secuencia no válida″. SQLSTATE=38608	La secuencia proporcionada no es válida.
SQL0443N	La rutina ″DB2LS.LSPEP2AMBNUC″ (nombre específico ″LSPEP2AMBNUCUT″) ha devuelto un error SQLSTATE con el texto de diagnóstico ″No se ha encontrado ninguna conversión″. SQLSTATE=38610	El archivo de tabla de conversión está vacío.
SQL0443N	La rutina ″LSPEP2AMBNUC″ (nombre específico ″LSPEP2AMBNUCUT″) ha devuelto un error SQLSTATE con el texto de diagnóstico ″No se puede abrir el archivo de tabla de conversión″. SQLSTATE=38612	El archivo de tabla de conversión especificado no existe.
SQL0443N	La rutina ″DB2LS.LSPEP2AMBNUC″ (nombre específico ″LSPEP2AMBNUCUT″) ha devuelto un error SQLSTATE con el texto de diagnóstico ″Se está leyendo una línea demasiado larga del archivo″. SQLSTATE=38614	El archivo contenía una línea más larga de lo permitido.

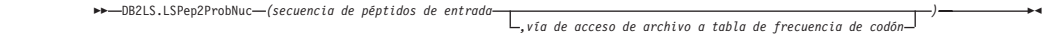
Tabla 14. Mensajes emitidos por la función definida por el usuario LSPep2AmbNuc (continuación)

Código de error	Mensaje	Explicación
SQL0443N	La rutina ″DB2LS.LSPEP2AMBNUC″ (nombre específico ″LSPEP2AMBNUCUT″) ha devuelto un error SQLSTATE con el texto de diagnóstico ″Archivo de datos no válido″. SQLSTATE=38615	El formato del archivo no es válido.
SQL0443N	La rutina ″LSPEP2AMBNUC″ (nombre específico ″LSPEP2AMBNUCUT″) ha devuelto un error SQLSTATE con el texto de diagnóstico ″No se puede construir la tabla de conversión″. SQLSTATE=38611	Se han encontrado símbolos no válidos en el archivo.

Información relacionada:

- “Función definida por el usuario LSPep2AmbNuc” en la página 47

Función definida por el usuario LSPep2ProbNuc



secuencia de péptidos de entrada
Una representación de cadena de caracteres válida que describe una secuencia de péptidos. La representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 10890 bytes. Los datos de entrada utilizan los símbolos de aminoácidos estándares.

vía de acceso de archivo a tabla de frecuencia de codón
Es la tabla de frecuencia de codón. Incluya la vía de acceso de archivo para encontrar la tabla de frecuencia. El valor de cadena de la vía de acceso no debe tener más de 255 caracteres.

El nombre de esquema es DB2LS.

Utilice la función LSPep2ProbNuc para generar la secuencia de nucleótidos más probable, a partir de una secuencia de péptidos, basada en la tabla de frecuencia de codón especificada en el segundo argumento.

El resultado de la función es una cadena de caracteres con el tipo de datos VARCHAR y una longitud real no superior a 32672 que representa la secuencia de nucleótidos más probable utilizando la tabla de frecuencia de codón.

Información relacionada:

- "Función definida por el usuario LSPep2AmbNuc" en la página 47
- "Función definida por el usuario LSPep2ProbNuc - mensajes de error" en la página 53
- "Función definida por el usuario LSPep2ProbNuc - ejemplo" en la página 52

Función definida por el usuario LSPep2ProbNuc - ejemplo

El ejemplo siguiente muestra cómo transformar una secuencia de péptidos en una secuencia de nucleótidos utilizando las conversiones más probables definidas en la tabla de frecuencia yeast_high.cod.

```
values db2ls.LSPep2ProbNuc('RDNNDDDN', '\data\yeast_high.cod')
```

El resultado de la sentencia de valores anterior es:

```
AGAGACAATAACGACGATGATAAC
```

Una segunda ejecución de la misma sentencia produce la cadena siguiente:

AGA**GAT**AATAACGACGAT**GAC**AAC

Una tercera ejecución de la misma sentencia produce la siguiente cadena con valores aleatorios:

AGAGAT**AAC**AACGAC**GACGATAAT**

Los codones que aparecen en negrita resaltan las diferencias entre la transformación actual y la anterior.

Los resultados de la única sentencia values muestran que la función LSPep2ProbNuc elige uno de los símbolos posibles basándose en estadísticas anteriores. Esto es diferente de la función LSPep2AmbNuc ya que ésta utiliza símbolos ambiguos en los que existen más conversiones posibles.

La función LSPep2ProbNuc elige las conversiones más probables para cada símbolo y, a continuación, sustituye cada símbolo por una conversión aleatoria del conjunto previamente elegido. Supongamos que tenemos los datos siguientes en una tabla de frecuencia:

Tabla 15. Datos de la tabla de frecuencia de ejemplo

Aminoácido	Codón	Frecuencia
Ala	GCG	0,17
Ala	GCA	0,13
Ala	GCT	0,17
Ala	GCC	0,53

Supongamos que la secuencia de péptidos contiene cuatro símbolos "A" (Ala). La función convierte dos veces A en GCC; una vez en GCG y otra vez en GCT. Sin embargo, el orden con el que la función produce las conversiones es aleatorio. La consulta podría convertir la primera A en cada una de las conversiones a partir del conjunto {GCC, GCC, GCG, GCT}. El resultado siempre son dos ocurrencias de GCC, una ocurrencia de GCG y una ocurrencia de GCT en la secuencia de ADN de salida. Varias ejecuciones de la función en la misma secuencia pueden devolver secuencias de ADN con los valores intercambiados.

Información relacionada:

- "Función definida por el usuario LSPep2ProbNuc" en la página 51
- "Función definida por el usuario LSPep2ProbNuc - mensajes de error" en la página 53
- "Función definida por el usuario LSPep2AmbNuc - ejemplo" en la página 49

Función definida por el usuario LSPep2ProbNuc - mensajes de error

Tabla 16. Mensajes emitidos por la función definida por el usuario LSPep2ProbNuc

Código de error	Mensaje	Explicación
SQL0443N	La rutina ″DB2LS.LSPEP2PROBNUC″ (nombre específico ″LSPEP2PROBNUC″) ha devuelto un error SQLSTATE con el texto de diagnóstico ″Secuencia no válida″. SQLSTATE=38608	La secuencia de entrada no es válida.
SQL0443N	La rutina ″DB2LS.LSPEP2PROBNUC″ (nombre específico ″LSPEP2PROBNUC″) ha devuelto un error SQLSTATE con el texto de diagnóstico ″No se ha encontrado ninguna conversión″. SQLSTATE=38610	El archivo de tabla de frecuencia de codón está vacío.
SQL0443N	La rutina ″LSPEP2PROBNUC″ (nombre específico ″LSPEP2PROBNUC″) ha devuelto un error SQLSTATE con el texto de diagnóstico ″No se puede abrir el archivo de tabla de conversión″. SQLSTATE=38612	El archivo no existe.

Tabla 16. Mensajes emitidos por la función definida por el usuario LSPep2ProbNuc (continuación)

Código de error	Mensaje	Explicación
SQL0443N	La rutina ″DB2LS.LSPEP2PROBNUC″ (nombre específico ″LSPEP2PROBNUC″) ha devuelto un error SQLSTATE con el texto de diagnóstico ″Se está leyendo una línea demasiado larga del archivo″. SQLSTATE=38614	El archivo contiene líneas más largas de lo permitido.
SQL0443N	La rutina ″DB2LS.LSPEP2PROBNUC″ (nombre específico ″LSPEP2PROBNUC″) ha devuelto un error SQLSTATE con el texto de diagnóstico ″Archivo de datos no válido″. SQLSTATE=38615	El formato del archivo no es válido.
SQL0443N	La rutina ″LSPEP2PROBNUC″ (nombre específico ″LSPEP2PROBNUC″) ha devuelto un error SQLSTATE con el texto de diagnóstico ″No se puede construir la tabla de conversión″. SQLSTATE=38611	El archivo contiene símbolos no válidos.

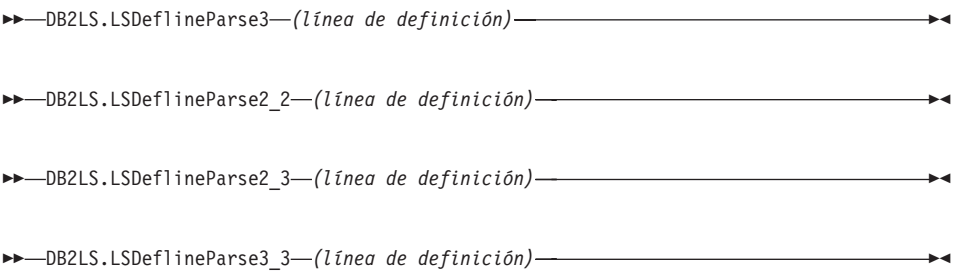
Información relacionada:

- “Función definida por el usuario LSPep2ProbNuc” en la página 51
- “Función definida por el usuario LSPep2ProbNuc - ejemplo” en la página 52

Funciones definidas por el usuario de análisis de línea de definición

Las funciones definidas por el usuario de análisis de línea de definición analizan los elementos de una línea de definición, por ejemplo, para permitir uniones con otras fuentes de datos en identificadores de secuencia analizados fuera de la línea de definición o para evaluar los predicados de partes de la línea de definición, tales como `'species = "humano"'`. Las funciones de línea de definición cubren la mayoría de formatos de línea de definición comunes. Entre los ejemplos se incluyen elementos de línea de definición que devuelve el reiniciador de BLAST o que se presentan en un archivo de datos con formato FASTA.

Funciones definidas por el usuario LSDeflineParse



línea de definición
Una representación de cadena válida de una línea de definición en formato FASTA. La cadena debe ser del tipo de datos VARCHAR y tener una longitud real no superior a 1024 bytes.

El nombre de esquema es DB2LS.

Cada función LSDeflineParse analiza los campos del identificador de secuencia FASTA estándar NCBI (NSID) y la descripción en columnas de una tabla. Las líneas de definición que son definiciones compuestas se dividen en varias filas y cada fila contiene una única definición de componente.

LSDeflineParse2 analiza una línea de definición con un NSID de dos campos. El resultado de la función es una tabla con cuatro columnas:

Tabla 17. Descripciones de las columnas de la tabla de resultados de la función definida por el usuario LSDeflineParse2

Nombre de columna	Descripción
ROWID	Entero que enumera las filas devueltas desde la función.
TAG	VARCHAR con un máximo de tres caracteres que representa el identificador NSID.
IDENTIFIER	VARCHAR con un máximo de 20 caracteres que representa el segundo campo de identificador del NSID.
DESCRIPTION	VARCHAR con un máximo de 1019 caracteres.

LSDeflineParse3 analiza una línea de definición con un NSID de tres campos. El resultado de la función es una tabla con cinco columnas:

Tabla 18. Descripciones de las columnas de la tabla de resultados de la función definida por el usuario LSDeflineParse3

Nombre de columna	Descripción
ROWID	Entero que enumera las filas devueltas desde la función.
TAG	VARCHAR con un máximo de tres caracteres que representa el identificador NSID.
ACCESSION	VARCHAR con un máximo de 20 caracteres que representa el segundo campo de identificador del NSID.
LOCUS	VARCHAR con un máximo de 20 caracteres que representa el tercer campo de identificador del NSID.
DESCRIPTION	VARCHAR con un máximo de 1017 caracteres.

LSDeflineParse2_2 analiza una línea de definición que tiene un identificador compuesto que consta de un par de NSID de dos campos concatenados. El resultado de la función es una tabla con seis columnas:

Tabla 19. Descripciones de las columnas de la tabla de resultados de la función definida por el usuario LSDeflineParse2_2

Nombre de columna	Descripción
ROWID	Entero que enumera las filas devueltas desde la función.
TAG1	VARCHAR con un máximo de tres caracteres que representa el identificador NSID del primer identificador.
IDENTIFIER1	VARCHAR con un máximo de 20 caracteres que representa el campo de segundo identificador del primer NSID.
TAG2	VARCHAR con un máximo de tres caracteres que representa el identificador NSID del primer identificador.
IDENTIFIER2	VARCHAR con un máximo de 20 caracteres que representa el campo de segundo identificador del segundo NSID.
DESCRIPTION	VARCHAR con un máximo de 1015 caracteres.

LSDeflineParse2_3 analiza una línea de definición con un identificador compuesto que consta de un NSID de dos campos concatenado con un NSID de tres campos. El orden de concatenación en la línea de definición de entrada (tanto si el NSID de dos campos va antes del NSID de tres campos, o

viceversa) no es importante. El resultado de la función es una tabla con siete columnas:

Tabla 20. Descripciones de las columnas de la tabla de resultados de la función definida por el usuario LSDeflineParse2_3

Nombre de columna	Descripción
ROWID	Entero que enumera las filas devueltas desde la función.
TAG1	VARCHAR con un máximo de tres caracteres que representa el identificador NSID del identificador de dos campos.
IDENTIFIER	VARCHAR con un máximo de 20 caracteres que representa el campo de segundo identificador del NSID de dos campos.
TAG2	VARCHAR con un máximo de tres caracteres que representa el identificador NSID del identificador de tres campos.
ACCESSION	VARCHAR con un máximo de 20 caracteres que representa el campo de segundo identificador del NSID de tres campos.
LOCUS	VARCHAR con un máximo de 20 caracteres que representa el campo de tercer identificador del NSID de tres campos.
DESCRIPTION	VARCHAR con un máximo de 1013 caracteres.

LSDeflineParse3_3 analiza una línea de definición con un identificador compuesto que consta de un par de NSID de tres campos. El resultado de la función es una tabla con ocho columnas:

Tabla 21. Descripciones de las columnas de la tabla de resultados de la función definida por el usuario LSDeflineParse3_3

Nombre de columna	Descripción
ROWID	Entero que enumera las filas devueltas desde la función.
TAG1	VARCHAR con un máximo de tres caracteres que representa el identificador NSID del primer identificador.
ACCESSION1	VARCHAR con un máximo de 20 caracteres que representa el campo de segundo identificador del primer NSID.
LOCUS1	VARCHAR con un máximo de 20 caracteres que representa el campo de tercer identificador del primer NSID.

Tabla 21. Descripciones de las columnas de la tabla de resultados de la función definida por el usuario LSDeflineParse3_3 (continuación)

Nombre de columna	Descripción
TAG2	VARCHAR con un máximo de tres caracteres que representa el identificador NSID del primer identificador.
ACCESSION2	VARCHAR con un máximo de 20 caracteres que representa el campo de segundo identificador del segundo NSID.
LOCUS2	VARCHAR con un máximo de 20 caracteres que representa el campo de tercer identificador del segundo NSID.
DESCRIPTION	VARCHAR con un máximo de 1014 caracteres.

Información relacionada:

- "Función definida por el usuario LSDeflineParse — ejemplos" en la página 58

Función definida por el usuario LSDeflineParse — ejemplos

Este tema contiene siete ejemplos que muestran cómo las funciones definidas por el usuario LSDeflineParse analizan líneas de definición en tablas de resultados.

La consulta y la tabla de resultados de ejemplo siguientes muestran cómo la función definida por el usuario LSDeflineParse2 analiza una línea de definición que contiene un NSID de dos campos:

```
select *
from table(DB2LS.LSDeflineParse2(
          '>gi|12346 hypothetical protein 185 –wheat chloroplast')) as t
```

La tabla de resultados contiene los datos siguientes:

Tabla 22. Datos resultantes de la función definida por el usuario LSDeflineParse2

Nombre de columna	Datos
ROWID	1
TAG	gi
IDENTIFIER	12346
DESCRIPTION	hypothetical protein 185 – wheat chloroplast

La consulta y la tabla de resultados de ejemplo siguientes muestran cómo la función definida por el usuario LSDeflineParse3 analiza una línea de definición que contiene un NSID de tres campos:

```
select *
from table(DB2LS.LSDeflineParse3('
          >gb|U37104|APU37104 Aethia pusilla cytochrome b gene')) as t
```

La tabla de resultados contiene los datos siguientes:

Tabla 23. Datos resultantes de la función definida por el usuario LSDeflineParse3

Nombre de columna	Datos
ROWID	1
TAG	gb
ACCESSION	U37104
LOCUS	APU37104
DESCRIPTION	Aethia pusilla cytochrome b gene

La consulta y la tabla de resultados de ejemplo siguientes muestran cómo la función definida por el usuario LSDeflineParse2_2 analiza una línea de definición que contiene un identificador compuesto que consta de un par de NSID de 2 campos:

```
select *
from table(DB2LS.LSDeflineParse2_2(
          '>gb|U37104|gim|73401A Aethia pusilla cytochrome b gene')) as t
```

La tabla de resultados contiene los datos siguientes:

Tabla 24. Datos resultantes de la función definida por el usuario LSDeflineParse2_2

Nombre de columna	Datos
ROWID	1
TAG1	gb
IDENTIFIER1	U37104
TAG2	gim
IDENTIFIER2	73401A
DESCRIPTION	Aethia pusilla cytochrome b gene

La siguiente consulta de ejemplo contiene una línea de definición con un identificador compuesto que consta de un NSID de 2 campos concatenado con un NSID de 3 campos. El ejemplo muestra cómo la función LSDeflineParse2_3 analiza la línea de definición.

```
select *
from table(DB2LS.LSDeflineParse2_3('
          >gi|12346|gp|CAA44030.1|CHTAHSRA_4
          hypothetical protein 185 – wheat chloroplast')) as t
```

La tabla de resultados contiene los datos siguientes:

Tabla 25. Datos resultantes de la función definida por el usuario LSDeflineParse2_3

Nombre de columna	Datos
ROWID	1
TAG1	gi
IDENTIFIER	12346
TAG2	gp
ACCESSION	CAA44030.1
LOCUS	CHTAHSRA_4
DESCRIPTION	hypothetical protein 185 – wheat chloroplast

La siguiente consulta de ejemplo contiene una línea de definición con un identificador compuesto que consta de un NSID de 3 campos concatenado con un NSID de 2 campos. El ejemplo muestra cómo la función LSDeflineParse2_3 analiza la línea de definición.

```
select *
from table(DB2LS.LSDeflineParse2_3('
          >gp|CAA44030.1|CHTAHSRA_4|gi|12346
          hypothetical protein 185 - wheat chloroplast')) as t
```

La tabla de resultados contiene los datos siguientes:

Tabla 26. Datos resultantes de la función definida por el usuario LSDeflineParse2_3

Nombre de columna	Datos
ROWID	1
TAG1	gi
IDENTIFIER	12346
TAG2	gp
ACCESSION	CAA44030.1
LOCUS	CHTAHSRA_4
DESCRIPTION	hypothetical protein 185 – wheat chloroplast

La consulta y la tabla de resultados de ejemplo siguientes muestran cómo la función definida por el usuario LSDeflineParse3_3 analiza una línea de definición que contiene un identificador compuesto con un par de NSID de 3 campos:

```
select * from table(DB2LS.LSDeflineParse3_3('
                  >dbj|AAD55586.1|AF055084_1|gp|CAA44030.1|CHTAHSRA_4
                  hypothetical protein 185 – wheat chloroplast')) as t
```

La tabla de resultados contiene los datos siguientes:

Tabla 27. Datos resultantes de la función definida por el usuario LSDeflineParse3_3

Nombre de columna	Datos
ROWID	1
TAG1	dbj
ACCESSION1	AAD55586.1
LOCUS1	AF055084_1
TAG2	gp
ACCESSION2	CAA44030.1
LOCUS2	CHTAHSRA_4
DESCRIPTION	hypothetical protein 185 – wheat chloroplast

Puede utilizar cualquiera de las funciones definidas por el usuario de línea de definición para analizar una línea de definición compuesta. La siguiente consulta de ejemplo contiene una línea de definición compuesta con varias definiciones separadas por un carácter Control-A. Encontrará este tipo de línea de definición en la base de datos nr de proteínas no redundantes de NCBI. El ejemplo muestra cómo la función LSDeflineParse2_3 analiza la línea de definición.

```
select *
from table(DB2LS.LSDeflineParse2_3('
           >gi|12346|gp|CAA44030.1|CHTAHSRA_4
           hypothetical protein 185 - wheat chloroplast
     ^Agp|CAA44030.1|CHTAHSRA_4|gi|12346
           hypothetical protein 185 - wheat chloroplast')) as t
```

La tabla de resultados contiene los datos siguientes:

Tabla 28. Datos resultantes de la función definida por el usuario LSDeflineParse2_3

Nombre de columna	Datos	Datos
ROWID	1	2
TAG1	gi	gi
IDENTIFIER	12346	12346
TAG2	gp	gp
ACCESSION	CAA44030.1	CAA44030.1
LOCUS	CHTAHSRA_4	CHTAHSRA_4
DESCRIPTION	hypothetical protein 185 – wheat chloroplast	hypothetical protein 185 – wheat chloroplast

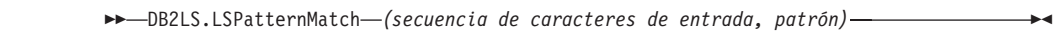
Información relacionada:

- "Funciones definidas por el usuario LSDeflineParse" en la página 54

Funciones definidas por el usuario de coincidencia de patrones generalizada

Las funciones definidas por el usuario de coincidencia de patrones generalizas identifican áreas de interés de una cadena determinada como, por ejemplo, una secuencia de nucleótidos o de péptidos.

Función definida por el usuario LSPatternMatch



secuencia de caracteres de entrada
La representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes.

patrón El patrón tal como se especifica en cualquier expresión regular de Perl válida. La representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes.

El nombre de esquema es DB2LS.

Puede utilizar la función definida por el usuario LSPatternMatch para buscar el patrón especificado en la secuencia de nucleótidos o de péptidos de entrada.

El resultado de la función es un entero que representa la posición de la primera coincidencia del patrón en la secuencia. La función devuelve un valor de cero si no existe ninguna coincidencia.

Si tiene patrones escritos con la sintaxis de PROSITE, puede convertirlos a la sintaxis de Perl con la función definida por el usuario LSPrositePattern. A continuación, puede utilizar la sintaxis convertida con la función definida por el usuario LSPatternMatch.

Información relacionada:

- "Función definida por el usuario LSPatternMatch – ejemplo" en la página 62
- "Función definida por el usuario LSPrositePattern" en la página 65

Función definida por el usuario LSPatternMatch – ejemplo

En el ejemplo siguiente, busque la posición inicial de la cadena que coincide con "coward", "cowage", "cowboy" o "cowl".

```
values DB2LS.LSPatternMatch('joe the cowboy is next', 'cow(ard|age|boy|l)')
```

La función busca por caracteres y, en este ejemplo, devuelve un valor de nueve. La cadena “cowboy” empieza en la posición nuevo, suponiendo que la primera posición es uno.

En el ejemplo siguiente, busque la posición inicial de la cadena que coincide con “not ” o “non ”:

```
values DB2LS.LSPatternMatch('match not and non but
                             no match for no or none', 'no[tn] ')
```

La función busca por caracteres y, en este ejemplo, devuelve un valor de siete. La cadena “not ” empieza en la posición siete, suponiendo que la primera posición es uno.

LSPatternMatch es útil para seleccionar sentencias para filtrar los resultados utilizando la sintaxis de PERL, que es una sintaxis más eficaz que la sentencia SQL LIKE. En el ejemplo siguiente, utilice LSPatternMatch en una salida de Blast para filtrar los genes que coinciden con un patrón determinado:

```
SELECT BlastOutput.*
 FROM BlastOutput
 WHERE db2ls.LSPatternMatch(HSP_H_Seq, 'F[GSTV]PRL') > 0;
```

Si está más familiarizado con la sintaxis de PROSITE, puede utilizar la función LSPrositePattern en la consulta anterior. Efectúe los siguientes cambios en la consulta:

```
SELECT BlastOutput.*
  FROM BlastOutput
  WHERE db2ls.LSPatternMatch(HSP_H_Seq,
          db2ls.LSPrositePattern('F-[GSTV]-P-R-L.') ) > 0;
```

Las funciones de coincidencia de patrones son útiles para buscar en otros tipos de texto, así como en las secuencias de nucleótidos o de péptidos. Considere la utilización de la sentencia SQL LIKE si el rendimiento puede ser un problema.

El ejemplo siguiente muestra una consulta que filtra alineaciones hsp de BLAST basadas en motivos de proteínas que se encuentran en la línea de tema o de destino de la alineación. El ejemplo se ha adaptado a partir de Zhang,Z., Schaffer,A.A., Miller,W., Madden,T.L., Lipman,D.J., Koonin,E.V. and Altschul,S.F. (1998) Protein sequence similarity searches using patterns as seeds. *Nucl. Acids Res.*, **26**, 3896-3990.

La consulta siguiente sólo devuelve las alineaciones en las que la secuencia de tema incluye el dominio P-loop ATPase [GA]xxxxGK[ST]. La consulta utiliza CED4, el regulador *Caenorhabditis elegans* de muerte celular, como secuencia de consulta en la base de datos de secuencia de proteínas no redundantes de NCBI. La base de datos recupera la secuencia de consultas de blast desde la conversión de la característica CDS de la entrada de GenBank X69016.

```
SELECT HSP_Q_Seq, HSP_Midline, HSP_H_Seq
FROM BlastP b, GBseq gs, gbfeat gf, gbqual gq
WHERE gs.PRIMARYACCESSION = 'X69016' and
                  gs.sequencekey = gf.sequencekey and
                  gf.featurejoinkey = gq.featurejoinkey and
                  gf.FeatureKey = 'CDS' and
                  gq.QualifierName = 'translation' and
                  gq.QualifierValue = b.BlastSeq and
                  db2ls.LSPatternMatch(HSP_H_Seq,
                    db2ls.LSPrositePattern('[GA]-x(4)-G-K-[ST].') ) > 0;
```

Puede utilizar la consulta de ejemplo siguiente para buscar HSP en una secuencia genómica que contiene poliformismos de nucleótidos simples (SNP) putativos en relación con una secuencia de consulta canónica. Se ha adaptado a partir de Extending traditional query-based integration approaches for functional characterization of post-genomic data. (2001) Barbara A Eckman, Anthony S Kosky, and Leonardo A Laroco Jr. *Bioinformatics* 17(7), 587-601.

La consulta utiliza coincidencias con patrones en la línea media hsp de blast para buscar un patrón de ≥20 coincidencias perfectas seguida de una única no coincidencia seguida de ≥20 coincidencias perfectas. Es decir, 20 caracteres ″|″, un único espacio y, a continuación, 20 caracteres ″|″ en la línea media de la alineación.

Este ejemplo también muestra la utilización de la función definida por el usuario LSPatternMatch en cadenas que no son secuencias de nucleótidos ni de péptidos.

```
SELECT   HSP_Info, HSP_Midline, HSP_H_Seq
FROM BlastOutput
WHERE db2ls.LSPatternMatch(HSP_Midline, '\|{20} \|{20}') > 0;
```

Puede volver a escribir la consulta anterior como:

```
SELECT HSP_Info, HSP_Midline, HSP_H_Seq, func.Position, func.Match
FROM BlastOutput,
     table( select * as c from table(
            LSMultiMatch(HSP_Midline, '\|{20} \|{20}') )
     as f) as func
```

Esta segunda consulta devolverá las filas de blast con una coincidencia además de la cadena coincidente y su posición en la secuencia.

BlastOutput es una vista de un apodo BlastN.

Información relacionada:

- “Función definida por el usuario LSPrositePattern - ejemplo” en la página 65
- “Función definida por el usuario LSPatternMatch” en la página 62
- “Función definida por el usuario LSPrositePattern” en la página 65

Función definida por el usuario LSPrositePattern

patrón La sintaxis de coincidencia de patrones especificada por la sintaxis de Prosite. La representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes.

El nombre de esquema es DB2LS.

Utilice la función definida por el usuario LSPrositePattern para convertir de la sintaxis de PROSITE a la sintaxis de PERL. A continuación, puede utilizar la sintaxis convertida con las funciones definidas por el usuario LSPatternMatch, LSMultiMatch y LSMultiMatch3.

El resultado de la función es una cadena de caracteres que representa una expresión regular en la sintaxis de Perl. La representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes.

Información relacionada:

- "Función definida por el usuario LSPrositePattern - ejemplo" en la página 65
- "Función definida por el usuario LSPatternMatch" en la página 62

Función definida por el usuario LSPrositePattern - ejemplo

En el ejemplo siguiente, convierta un patrón de la sintaxis de PROSITE a la sintaxis de PERL.

```
values db2ls.LSPrositePattern('[AC]-x-V-x(4)-{ED}.');
```

La función convierte el patrón de entrada de sintaxis de PROSITE en un patrón equivalente en la sintaxis de Perl, tal como se muestra en el ejemplo siguiente:

```
[AC].V.{4}[^ED]
```

El siguiente ejemplo convierte otro patrón de la sintaxis de PROSITE a la sintaxis de PERL:

```
values db2ls.LSPrositePattern('<A-x-[ST](2)-x(0,1)-V.');
```

La función convierte la cadena desde la sintaxis de PROSITE basándose en el patrón de entrada y devuelve lo siguiente:

```
\AA.[ST]{2}.{0,1}V
```

El ejemplo siguiente convierte el patrón correspondiente a la entrada de base de datos de PROSITE con el número de ID PS01205 al patrón de PERL que se utiliza como entrada en las funciones de coincidencia de patrones.

```
values db2ls.LSPrositePattern('R-P-L-[IV]-x-[NS]-F-G-S-[CA]-T-C-P-x-F.')
```

El resultado de esta consulta es:

```
RPL[IV].[NS]FGS[CA]TCP.F
```

El ejemplo siguiente muestra cómo utilizar la función en una consulta. La consulta sólo imprime las secuencias que coinciden con el patrón de PROSITE especificado.

```
SELECT H_Accession, HSP_Info, HSP_H_Seq
FROM BlastOutput
WHERE db2ls.LSPatternMatch( HSP_H_Seq,
  db2ls.LSPrositePattern('R-P-L-[IV]-x-[NS]-F-G-S-[CA]-T-C-P-x-F.') ) > 0;
```

El ejemplo siguiente convierte el patrón correspondiente a la entrada de PROSITE cuyo ID es PS00261:

```
values db2ls.LSPrositePattern('C-[STAGM]-G-[HFYL]-C-x-[ST].')
```

El resultado de esta consulta es:

```
C[STAGM]G[HFYL]C.[ST]
```

Información relacionada:

- "Función definida por el usuario LSPatternMatch – ejemplo" en la página 62
- "Función definida por el usuario LSPrositePattern" en la página 65

Soporte de expresión regular

El soporte de expresión regular se proporciona en el paquete de bibliotecas PCRE, con software de código abierto, escrito por Philip Hazel y cuyo copyright pertenece a la University of Cambridge, England.

Puede encontrar la fuente en ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/.

Funciones definidas por el usuario de GeneWise

La función definida por el usuario de GeneWise alinea una secuencia de proteínas con una secuencia genómica.

GeneWise es un componente utilizado con frecuencia que alinea una secuencia de proteínas con una secuencia genómica de ADN, que permite errores de marco de desplazamiento e intrones.

Cómo enlazar con GeneWise

Este tema describe el procedimiento para enlazar con la biblioteca de GeneWise.

Procedimiento:

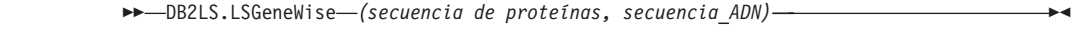
1. Baje el paquete Wise2 versión 2.1.20c desde http://www.ebi.ac.uk/Wise2.
2. Expanda el archivador en la carpeta que prefiera.
3. Compile el paquete con el soporte de pthread. Para obtener más información sobre este paso, consulte la documentación de Wise2.
4. Ejecute **make api** en este directorio raíz.
5. Establezca la variable de entorno WISE2_HOME para que apunte al directorio raíz del paquete Wise2.
6. Establezca la variable WISECONFIGDIR en el archivo sqllib/cfg/db2dj.ini para que apunte al subdirectorio wisecfg.

   Por ejemplo, si el paquete Wise2 está instalado en /usr/wise2.1.20c/, añada WISECONFIGDIR=/usr/wise2.1.20c/wisecfg/ al archivo db2dj.ini.
7. Ejecute **djxlinkLSGeneWise** desde el directorio sqllib/bin y compruebe su salida.
8. Compruebe djxlinkLSGeneWise.out en el directorio sqllib/lib.
9. Si no se informa de ningún error, esto indica que la biblioteca se ha creado satisfactoriamente.

Información relacionada:

• "Función definida por el usuario LSGeneWise" en la página 67

Función definida por el usuario LSGeneWise



secuencia de proteínas
Una representación de cadena de caracteres válida que describe una secuencia de péptidos. La representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes.

secuencia_ADN
Una representación de cadena de caracteres válida que describe una secuencia de nucleótidos. La representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes.

La Tabla 29 en la página 68 muestra la tabla de salida de una fila que devuelve la función LSGeneWise.

Tabla 29. Nombres de columna, tipos y descripciones para la tabla de salida de la función LSGeneWise

Nombre de columna	Tipo	Descripción
PROTEIN_OFFSET	INTEGER	Representa el desplazamiento inicial de la secuencia de proteínas de entrada en el que se ha encontrado una alineación.
DNA_OFFSET	INTEGER	Representa el desplazamiento inicial de la secuencia de ADN de entrada en el que se ha encontrado una alineación.
PROTEIN	VARCHAR(32672)	Fragmento de la secuencia de entrada que representa la secuencia alineada.
SIMILARITY	VARCHAR(32672)	Muestra la coincidencia entre la secuencia de proteínas y la secuencia de adn. Las coincidencias perfectas se marcan con la correspondiente letra de símbolo. Las coincidencias no perfectas con un resultado positivo se indican con el signo ″+″ y las no coincidencias se indican con un espacio.
TRANSLATED _DNA	VARCHAR(32672)	La secuencia de ADN convertida. La secuencia puede contener guiones y símbolos especiales tales como supresiones e intrones.
DNA	VARCHAR(32672)	La secuencia de ADN con marcadores especiales tales como desplazamiento de marco e intrones.

La correspondencia entre la salida del programa GeneWise y la salida de la UDF LSGeneWise es la siguiente:

- los desplazamientos de proteínas y de adn impresos por el programa GeneWise coinciden con la columnas PROTEIN_OFFSET y DNA_OFFSET.
- la secuencia de proteínas impresa en la primera línea de GeneWise coincide con la columna PROTEIN.
- la línea de similitud, la segunda línea de la salida de GeneWise coincide con la columna SIMILARITY.
- la tercera línea de la salida de GeneWise coincide con la columna TRANSLATED_DNA.
- las líneas cuarta, quinta y sexta de la salida de GeneWise se combinan, leídas verticalmente, en la columna DNA.

Utilice la función definida por el usuario LSGeneWise para alinear una secuencia de proteínas con una secuencia genómica de ADN, permitiendo errores de desplazamiento de marco e intrones.

Para obtener más información sobre la función definida por el usuario LSGeneWise, vea http://www.ebi.ac.uk/Wise2.

Tareas relacionadas:

- "Cómo enlazar con GeneWise" en la página 67

Información relacionada:

- "Función definida por el usuario LSGeneWise – ejemplo" en la página 69

Función definida por el usuario LSGeneWise – ejemplo

El ejemplo siguiente muestra una consulta en la que se utiliza la función definida por el usuario LSGeneWise y los datos resultantes.

```
select protein_offset, dna_offset, protein, similarity, translated_dna, dna
from table( db2ls.LSGeneWise( '
             VEPKRAVPRQDIDSPNAGATVKKLFVGALKDDHDEQSIRDYFQHFGNIVDINIVIDKETGK
             KRGFAFVEFDDYDPVDKVVLQKQHQLNGKMVDVKKALPKQNDQQGGGGGRGGPGGRAGGNR
             GNMGGGNYGNQNGGGNWNNGGNNWGNNR',
            'CACTTAACTGTGAAAGATATTTGTTGGTGGCATTAAAGAAGACACTGAAGAACATCACCTAAG
             AGATTATTTTGAACAGTATGGAAAAATTGAAGTGATTGAAATCATGACTGACCGAGGCAGTGG
             CAAGAAAAGGGGCTTTGCCTTRGTAACCTTTGACGACCATGACTCCGTGGATAAGATTGTCAT
             TCAGAAATACCATACTGTGAATGGCCACAACTGTGAAGTTAGAAAAGCCCTGTCAAAGCAAGA
             GATGGCTAGTGCTTCATCCAGCCAAAGAGGTCGAAGTGGTTCTGGAAACTTTGGTGGTGGTCG
             TGGAGGTGGTTTCGGTGGGAATGACAACTTCGGTCGTGGAGGAAACTTCAGTGGTCGTGGTYG
             CTTTGGTGGCAGCCGTGGTGGTGGTGGATATGGTGGC' ) ) as f;
```

Tabla 30. Tabla de resultados

Columna	Datos
PROTEIN_OFFSET	23
DNA_OFFSET	14
PROTEIN	KLFVGALKDDHDEQSIRDYFQHFGNIVDINIVIDKET GKKRGFAFVEFDDYDPVDKVVLQKQHQLNGKMVD VKKALPKQNDQQGGGGGRGGPGGRAGGNRGNMGG GNYGNQNGGGNWNNGGN
SIMILARITY	K+FVG +K+D +E +RDYF+ +G I I I+ D+ +GKKRGFA+V FDD+D VDK+V+QK H +NG +V+KAL KQ RG G GN+GGG G G N+ GGN
TRANSLATED_DNA	KIFVGGIKEDTEEHHLRDYFEQYGKIEVIEIMTDRGSGK KRGFAxVTFDDHDSVDKIVIQKYHTVNGHNCEVRKAL SKQEMASASSSQRGRSGS------ GNFGGGRGGGFGGNDNFGRGGN
DNA	aagatatttgttggtggcattaaagaagacactgaagaacatcacctaagagat...

Tareas relacionadas:

- "Cómo enlazar con GeneWise" en la página 67

Información relacionada:

- "Función definida por el usuario LSGeneWise" en la página 67

Funciones definidas por el usuario de motivos

Las funciones definidas por el usuario de motivos utilizan coincidencia de patrones en secuencias de nucleótidos o de aminoácidos.

Función definida por el usuario LSBarCode

secuencia de cadena de entrada
Una representación de cadena de caracteres válida que representa una alineación HSP entre dos fragmentos de la secuencia. La representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes.

El nombre de esquema es DB2LS.

Utilice la función definida por el usuario LSBarCode para utilizar una secuencia como entrada y generar otra secuencia sustituyendo cada carácter salvo los espacios y los signos más por el símbolo de barra vertical (|).

El resultado de la función es una secuencia de caracteres variables que representa una secuencia del código de barras.

Información relacionada:

- "Función definida por el usuario LSBarCode — ejemplo" en la página 70
- "Función definida por el usuario LSMultiMatch" en la página 72
- "Función definida por el usuario LSMultiMatch3" en la página 73

Función definida por el usuario LSBarCode — ejemplo

Este ejemplo crea un código de barras a partir de una secuencia de cadena:

```
values db2ls.LSBarCode(
   'MDY +G++L  GN  ++ +PASLTK+MT YVV +A+ + +I   D+VTVG+DAWA  NP ')
```

El resultado de la sentencia de valores anterior es:

```
||| +|++|  ||  ++ +||||||+|| ||| +|+ + +|   |+||||+||||  ||
```

El siguiente ejemplo muestra una utilización más realista de esta función. Supongamos que un investigador que ejecuta una búsqueda de BLAST desea que sólo se devuelvan las alineaciones de HSP que contengan menos del 24% de prolinas entre sus coincidencias perfectas. Este ejemplo utiliza la función

para calcular el porcentaje de prolinas (símbolo 'P') entre las coincidencias perfectas en una alineación devuelta por BLAST. Tenga en cuenta que este ejemplo también invoca la función definida por el usuario LSMultiMatch3. La consulta utiliza la función de coincidencia para buscar coincidencias perfectas. Se utiliza junto con la función LSBarCode en esta consulta debido a que Blast no siempre devuelve una secuencia de barras (" |") en una alineación. Esto se ilustra en el ejemplo siguiente:

```
Consulta:    MDYTTGQILTAGNEHQQRNPASLTKLMTGYVVDRAIDSHRITPDDIVTVGRDAWAKDNPV
Alineación:  MDY +G++L  GN  ++ +PASLTK+MT YVV +A+ + +I   D+VTVG+DAWA  NP
Destino:     MDYASGKVLAEGNADEKLDPASLTKIMTSYVVGQALKADKIKLTDMVTVGKDAWATGNPA
```

Para asegurarse de que la salida está alineada con la secuencia correcta de barras, utilice la función LSBarCode. La función sustituye todos los caracteres excepto los espacios y los signos más por una barra vertical.

```
SELECT BlastOutput.* , float( p )/ float( m ) AS percent_prolines
  FROM
  BlastOutput b,
  table(SELECT COUNT(*) AS p FROM table(
          db2ls.LSMultiMatch3(
            b.HSP_Q_Seq, 'P',
            db2ls.LSBarCode(b.HSP_Midline), '\|',
            b.HSP_H_Seq, 'P')
          ) AS f
        ) AS y,
  table(SELECT COUNT(*) AS m FROM table(
          db2ls.LSMultiMatch3(
            b.HSP_Q_Seq, '.',
            db2ls.LSBarCode(b.HSP_Midline), '\|',
            b.HSP_H_Seq, '.')
          ) AS f
        ) AS z
WHERE float(p) / float(m) < 0.25;
```

En esta consulta, BlastOutput en realidad es una vista sobre un apodo de Blast. La consulta utiliza la función LSMultiMatch3 para devolver coincidencias perfectas en la alineación. La primera utilización devuelve las coincidencias perfectas para el símbolo "P", la segunda devuelve todas las coincidencias perfectas. En la Tabla 31 se muestra una fila de la tabla de resultados.

Tabla 31. Fila de resultados de ejemplo

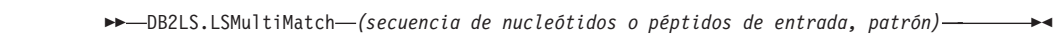
HSP_Q_SEQ	HSP_H_SEQ	HSP_INFO	PERCENT_ PROLINES
NIWDFMQGN...	NIWDFMQGN...	Identities = 80/80 (100%), Positives = 80/80 (100%), Gaps = 0/80 (0%)	+2.500000000000 00E-002

La consulta anterior se ha adaptado a partir de Extending traditional query-based integration approaches for functional characterization of post-genomic data. (2001) Barbara A Eckman, Anthony S Kosky and Leonardo A Laroco Jr. *Bioinformatics* 17(7), 587-601.

Información relacionada:

- "Función definida por el usuario LSMultiMatch3 – ejemplo" en la página 74
- "Función definida por el usuario LSBarCode" en la página 70

Función definida por el usuario LSMultiMatch



secuencia de nucleótidos o péptidos de entrada
Una representación de cadena de caracteres válida que describe una secuencia de nucleótidos o péptidos. La representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes.

patrón La gramática de coincidencia de patrones especificada por el lenguaje Perl. La representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes.

El nombre de esquema es DB2LS.

Utilice la función definida por el usuario LSMultiMatch para devolver una tabla para cada coincidencia que no solape la secuencia de entrada. Cada tabla consta de una posición de inicio y del fragmento de secuencia coincidente.

El resultado de la función es una tabla con dos columnas. La primera columna es un entero que representa la posición inicial de una coincidencia de patrón en la secuencia. La segunda columna es el fragmento de secuencia coincidente.

Información relacionada:

- "Función definida por el usuario LSMultiMatch - ejemplo" en la página 72
- "Función definida por el usuario LSBarCode" en la página 70
- "Función definida por el usuario LSMultiMatch3" en la página 73

Función definida por el usuario LSMultiMatch - ejemplo

Este ejemplo busca la posición y los fragmentos coincidentes para todas las coincidencias no solapadas de la entrada.

```
SELECT position, match FROM table
  (LSMultiMatch('match not and non but no match for no or none',
      'no[tn] ')) as f
```

La consulta devuelve una tabla que se basa en esta sentencia select que muestra los resultados de las coincidencias:

Tabla 32. Resultado de LSMultiMatch que devuelve varias filas

POSITION	MATCH
7	not
15	non

LSMultiMatch devuelve la posición y la cadena coincidente para todas las coincidencias. El ejemplo siguiente busca en Entrez Nucleotide entradas de secuencia que contengan un motivo determinado. La consulta imprime los identificadores de la secuencia y las secuencias coincidentes. Los subpatrones ".{0,9}" al principio y al final deben coincidir con nueve caracteres antes y después de la secuencia. La consulta también imprime estos caracteres.

```
select SequenceKey, Position, Match from GBSeq,
  table(db2ls.LSMultiMatch(Sequence, '.{0,9}(ATG|CGC)ACGGGC.{0,9}') )
  as fmatch
  WHERE entrez.contains(KeywordList,
        'Na/K/2Cl cotransporter AND nkcc1 gene') = 1;
```

El resultado de esta consulta es el siguiente:

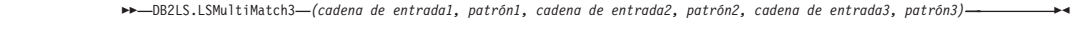
Tabla 33. Datos de Search Entrez

SEQUENCEKEY	POSITION	MATCH
N02B59AE0.04DD4E84	1	TGCTTGGTGATGACGGGCTACCCCAAC
N02B59AE0.04DD4E84	91	GGCCATGTTCGCACGGGCTCCAGAAGG
N02B59AE0.04DC5EF4	1	TGCTTGGTGATGACGGGCTACCCCAAC
N02B59AE0.04DC5EF4	91	GGCCATGTTCGCACGGGCTCCAGAAGG

Información relacionada:

- "Función definida por el usuario LSMultiMatch" en la página 72
- "Función definida por el usuario LSBarCode" en la página 70
- "Función definida por el usuario LSMultiMatch3" en la página 73

Función definida por el usuario LSMultiMatch3



cadenas de entrada

Una representación de cadena de caracteres válida que describe una secuencia de nucleótidos o péptidos o una cadena HSP_Midline de

una alineación de blast. La representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes.

patrón La gramática de coincidencia de patrones especificada por el lenguaje Perl. La representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes.

El nombre de esquema es DB2LS.

Utilice la función definida por el usuario LSMultiMatch3 para entrar tres patrones y tres cadenas y devolver cualquier posición en que las tres cadenas coincidan con sus respectivos patrones. Puede utilizar esta función definida por el usuario para llevar a cabo una coincidencia de patrones en una alineación.

El resultado de la función es una tabla con cuatro columnas. La primera columna es un entero que representa la posición inicial de una coincidencia del patrón en todas las secuencias. La función ancla todas las cadenas juntas en la primera posición. Las columnas segunda, tercera y cuarta son los fragmentos de secuencia coincidente.

Información relacionada:

- "Función definida por el usuario LSMultiMatch3 – ejemplo" en la página 74
- "Función definida por el usuario LSMultiMatch" en la página 72
- "Función definida por el usuario LSBarCode" en la página 70

Función definida por el usuario LSMultiMatch3 – ejemplo

El ejemplo siguiente utiliza la función para calcular el porcentaje de un símbolo de aminoácido determinado entre las coincidencias perfectas devueltas por Blast. Tenga en cuenta que este ejemplo también invoca la función definida por el usuario LSBarCode. Es necesario para la consulta puesto que Blast no siempre devuelve una secuencia de barras ("|") en una alineación. Esto se ilustra en el ejemplo siguiente:

```
Consulta:    MDYTTGQILTAGNEHQQRNPASLTKLMTGYVVDRAIDSHRITPDDIVTVGRDAWAKDNPV
Alineación:  MDY +G++L  GN  ++ +PASLTK+MT YVV +A+ + +I   D+VTVG+DAWA  NP
Destino:     MDYASGKVLAEGNADEKLDPASLTKIMTSYVVGQALKADKIKLTDMVTVGKDAWATGNPA
```

Para asegurarse de que la salida está alineada con la secuencia correcta de barras, utilice la función LSBarCode para convertir la secuencia. La función sustituye todos los caracteres que no sean ni un espacio ni "+" por una barra vertical.

```
SELECT BlastOutput.* , float( p )/ float( m ) AS percent_prolines
  FROM
  BlastOutput b,
  table(SELECT COUNT(*) AS p FROM table(
```

```
          db2ls.LSMultiMatch3(
            b.HSP_Q_Seq, 'P',
            db2ls.LSBarCode(b.HSP_Midline), '\|',
            b.HSP_H_Seq, 'P')
          ) AS f
        ) AS y,
  table(SELECT COUNT(*) AS m FROM table(
          db2ls.LSMultiMatch3(
            b.HSP_Q_Seq, '.',
            db2ls.LSBarCode(b.HSP_Midline), '\|',
            b.HSP_H_Seq, '.')
          ) AS f
        ) AS z
WHERE float(p) / float(m) < 0.25;
```

En esta consulta, BlastOutput es una vista de un apodo de Blast. La consulta utiliza la función LSMultiMatch3 para devolver coincidencias perfectas en la alineación. La primera utilización devuelve las coincidencias perfectas para el símbolo "P", la segunda devuelve todas las coincidencias perfectas. En la Tabla 34 se muestra una fila de la tabla de resultados.

Tabla 34. Fila de resultados de ejemplo

HSP_Q_SEQ	HSP_H_SEQ	HSP_INFO	PERCENT_PROLINES
NIWDFMQG...	NIWDFMQG...	Identities = 80/80 (100%), Positives = 80/80 (100%), Gaps = 0/80 (0%)	+2.50000000000000E-002

La consulta anterior se ha adaptado a partir de `Extending traditional query-based integration approaches for functional characterization of post-genomic data. (2001) Barbara A Eckman, Anthony S Kosky and Leonardo A Laroco Jr. *Bioinformatics* 17(7), 587-601.`

El ejemplo siguiente busca tres patrones separados en tres fragmentos de la cadena separados:

```
SELECT position, match_1, match_2, match_3
  FROM table(db2ls.LSMultiMatch3('zaza', 'a', 'abab',
             'b', 'bcbc', 'c')) as f
```

Devuelve las posiciones y las cadenas coincidentes para todas las coincidencias, tal como se muestra en la tabla siguiente:

Tabla 35. Resultado de varias coincidencias utilizando tres entradas

POSITION	MATCH_1	MATCH_2	MATCH_3
2	a	b	c
4	a	b	c

El ejemplo siguiente busca tres patrones separados dentro de tres fragmentos separados de la cadena:

```
SELECT position, match_1, match_2, match_3
  FROM table
  (LSMultiMatch3('cbccbbcccbbbccccbbbbccccc','c{1,3}b{1,3}c{1,3}',
   'abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz',
   '.','01234567890123456789012345678901234567890123456789','\d')) as f
```

Los resultados se muestran en la tabla siguiente:

Tabla 36. Resultado de varias coincidencias utilizando tres entradas

POSITION	MATCH_1	MATCH_2	MATCH_3
1	cbcc	a	0
7	cccbbbccc	g	6

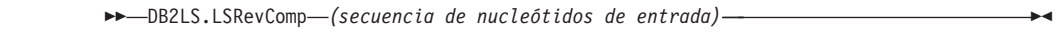
Información relacionada:

- "Función definida por el usuario LSBarCode — ejemplo" en la página 70
- "Función definida por el usuario LSBarCode" en la página 70
- "Función definida por el usuario LSMultiMatch3" en la página 73

Funciones definidas por el usuario de inversión

Las funciones definidas por el usuario de inversión invierten una secuencia de nucleótidos o de aminoácidos.

Función definida por el usuario LSRevComp



secuencia de nucleótidos de entrada
Una representación de cadena de caracteres válida que describe una secuencia de nucleótidos. La secuencia puede contener códigos de ambigüedad de la IUPAC. Una representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes.

El nombre de esquema es DB2LS.

El resultado de la función es una cadena de caracteres con un tipo de datos VARCHAR y una longitud real no superior a 32672 bytes que representa el complemento inverso de la secuencia de nucleótidos.

Información relacionada:

- "Función definida por el usuario LSRevComp—ejemplo" en la página 77

Función definida por el usuario LSRevComp—ejemplo

Puede utilizar la función LSRevComp en una sentencia de SQL siempre que desee utilizar una función incorporada que acepte una secuencia de nucleótidos. Por ejemplo:

```
SELECT DB2LS.LSRevComp(:NucSeq) FROM SYSDUMMY1;
```

Este ejemplo utiliza la función para devolver el complemento inverso de la secuencia de entrada que procede de una variable del lenguaje principal.

Si utiliza una cadena no válida o un tipo de datos no válido, obtendrá el siguiente mensaje de error:

```
SQL0443N La rutina "DB2LS.LSREVCOMP" (nombre específico "LSREVCOMP")
ha devuelto un error SQLSTATE con el texto de diagnóstico "Secuencia
no válida". SQLSTATE=38608
```

Se produce una excepción si el alfabeto de entrada no es correcto.

El ejemplo siguiente muestra cómo funciona la función definida por el usuario LSRevComp en una consulta:

```
SELECT HSP_H_Seq, db2ls.LSRevComp(HSP_H_Seq) as REV_HSP_H_Seq
FROM BlastN
WHERE BlastSeq='ccgctagtattggtcaatcttttgatatccaccgaa'
```

A continuación se muestran los resultados de la consulta:

```
HSP_H_SEQ                      REV_HSP_H_SEQ
------------------------------ ----------------------------

AGTATTGGTCAATCTTTTGAT          ATCAAAAGATTGACCAATACT

TGGTCAATCTTTTGATA              TATCAAAAGATTGACCA

TTGGCCAATCTTTTGATATCC          GGATATCAAAAGATTGGCCAA

TCAATCTTTTGATATCC              GGATATCAAAAGATTGA

GGATATCAAAAGATTGA              TCAATCTTTTGATATCC


  5 registro(s) seleccionado(s).
```

Puede utilizar la función de inversión junto con otras funciones definidas por el usuario de ciencias de la vida para convertir el complemento inverso de una secuencia de nucleótidos, tal como se muestra en el ejemplo siguiente:

```
values db2ls.LSNuc2Pep(
       db2ls.LSRevComp('TTTTTCTTATTGTCTTCCTCATCGTATTTCTTATGTTGCTGATGT'))
```

La consulta devuelve lo siguiente:

```
TSAT*EIR*GRQ*EK
```

Información relacionada:

- "Función definida por el usuario LSRevComp" en la página 76

Función definida por el usuario LSRevNuc

secuencia de nucleótidos de entrada
Una representación de cadena de caracteres válida que describe una secuencia de nucleótidos. Una representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes. La secuencia de nucleótidos debe formar parte del alfabeto de ADN o ser completamente el alfabeto de ADN.

El nombre de esquema es DB2LS.

El resultado de la función es una cadena de caracteres con un tipo de datos VARCHAR y una longitud real no superior a 32672 bytes que representa el orden inverso de la secuencia de nucleótidos.

Información relacionada:

- "Función definida por el usuario LSRevNuc - ejemplo" en la página 78
- "Función definida por el usuario LSRevComp" en la página 76
- "Función definida por el usuario LSRevPep" en la página 79

Función definida por el usuario LSRevNuc - ejemplo

Puede utilizar la función LSRevNuc en una sentencia de SQL siempre que desee utilizar una función incorporada que acepte una secuencia de nucleótidos. Por ejemplo:

```
SELECT DB2LS.LSRevNuc(:NucSeq) FROM SYSDUMMY1;
```

Este ejemplo utiliza la función para invertir los datos de entrada que proceden de una variable del lenguaje principal.

Si utiliza una cadena no válida o un tipo de datos no válido, obtendrá el siguiente mensaje de error:

```
SQL0443N La rutina "DB2LS.LSREVNUC" (nombre específico "LSREVNUC") ha
devuelto un error SQLSTATE con el texto de diagnóstico "Secuencia no válida".
SQLSTATE=38608
```

El ejemplo siguiente muestra la utilización de la función definida por el usuario LSRevNuc en una consulta.

```
SELECT HSP_H_Seq, db2ls.LSRevNuc(HSP_H_Seq) as REV_HSP_H_Seq
FROM BlastN
WHERE BlastSeq='gtaatacgtaggggggctagcgcgggcaaactgaagataaagc'
```

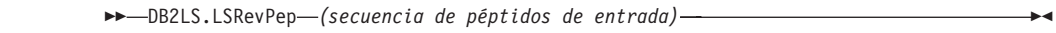
La tabla de resultados siguiente muestra las secuencias de nucleótidos invertidas que devuelve la consulta:

```
HSP_H_SEQ                                          REV_HSP_H_SEQ

-------------------------------------------------- -------------------------
CGCGGGCAAACTGAAGATAAAGC                            CGAAATAGAAGTCAAACGGGCGC

GCGCTAGCCCCCTACGTATTAC                             CATTATGCATCCCCCGATCGCG

GTAATACGTAGGGGGCTAGCG                              GCGATCGGGGGATGCATAATG

GTAATACGTAGGGGGCTAGCG                              GCGATCGGGGGATGCATAATG

GTAATACGTAGGGGGCTAGCG                              GCGATCGGGGGATGCATAATG


  5 registro(s) seleccionado(s).
```

Información relacionada:

- "Función definida por el usuario LSRevNuc" en la página 78

Función definida por el usuario LSRevPep



secuencia de péptidos de entrada
Una representación de cadena de caracteres válida que describe una secuencia de péptidos. Una representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes. La secuencia de entrada debe formar parte del alfabeto de proteína o ser completamente el alfabeto de proteína.

El nombre de esquema es DB2LS.

El resultado de la función es una cadena de caracteres con un tipo de datos VARCHAR y una longitud real no superior a 32672 que representa el orden inverso de la secuencia de péptidos.

Información relacionada:

- "Función definida por el usuario LSRevPep - ejemplo" en la página 80
- "Función definida por el usuario LSRevComp" en la página 76

- "Función definida por el usuario LSRevNuc" en la página 78

Función definida por el usuario LSRevPep - ejemplo

Puede utilizar la función LSRevPep en una sentencia de SQL siempre que desee utilizar una función incorporada que acepte una secuencia de péptidos. Por ejemplo:

```
SELECT DB2LS.LSRevPep(:NucSeq) FROM SYSDUMMY1;
```

Este ejemplo utiliza la función para invertir los datos de entrada que proceden de una variable del lenguaje principal.

Si utiliza una cadena no válida o un tipo de datos no válido, obtendrá el siguiente mensaje de error:

```
SQL0443N La rutina "DB2LS.LSREVPEP" (nombre específico "LSREVPEP") ha devuelto
un error SQLSTATE con el texto de diagnóstico "Secuencia no válida".
SQLSTATE=38608
```

El ejemplo siguiente muestra cómo se utiliza la función definida por el usuario LSRevPep en una consulta:

```
SELECT  HSP_H_Seq, db2ls.LSRevPep(HSP_H_Seq) as REV_HSP_H_Seq
FROM BlastP
WHERE BlastSeq='MLCEIECRALSTAHTRLIHDFEPRDALTYLEGKNIFTEDH'
```

La tabla siguiente muestra las secuencias de péptidos invertidas que devuelve la consulta.

```
HSP_H_SEQ                          REV_HSP_H_SEQ
---------------------------------- ----------------------------------

MLCEIECRALSTAHTRLIHDFEPRDALTYL...  HDETFINKGELYTLADRPEFDHILRTHATS...

RVVSTEHTRLVTDAYPEFSISFTATKN        NKTATFSISFEPYADTVLRTHETSVVR

STAHIRVLRDMVPGDEITCFYGSEFF         FFESGYFCTIEDGPVMDRLVRIHATS

AHTRRCPDHEPRGVITYL                 LYTIVGRPEHDPCRRTHA


  4 registro(s) seleccionado(s).
```

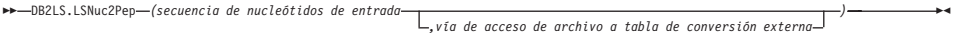
Información relacionada:

- "Función definida por el usuario LSRevPep" en la página 79

Conversión

Las funciones definidas por el usuario de conversión convierten una secuencia de nucleótidos en una secuencia de péptidos.

Función definida por el usuario LSNuc2Pep



secuencia de nucleótidos de entrada
Una representación de cadena de caracteres válida que describe una secuencia de nucleótidos. Una representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes.

vía de acceso a tabla de conversión externa
Si utiliza una tabla de conversión personalizada, incluya la información de vía de acceso de archivo para encontrar la tabla de conversión. El valor de cadena de la vía de acceso no debe tener más de 255 caracteres.

El nombre de esquema es DB2LS.

El resultado de la función es una cadena de caracteres con un tipo de datos VARCHAR y una longitud real no superior a 10890 bytes que representa la secuencia de péptidos.

La entrada es una secuencia de nucleótidos que utiliza el juego de caracteres de IUB. Las funciones suponen que el primer codón empieza en el primer carácter de la secuencia de nucleótidos. Si el primer codón no empieza en el primer carácter de la secuencia de nucleótidos, utilice una función SUBSTR en la secuencia de entrada.

El resultado de la función es una secuencia de péptidos, que utiliza los símbolos de aminoácidos estándares.

La función:
- Elimina los espacios de las secuencias de entrada.
- Ignora los nucleótidos extraños fuera de un marco de lectura.
- Devuelve una salida nula si la entrada es una secuencia de nucleótidos nula.

Información relacionada:
- "Función definida por el usuario LSNuc2Pep – ejemplo" en la página 82
- "Función definida por el usuario LSTransAllFrames" en la página 83

Función definida por el usuario LSNuc2Pep – ejemplo

Supongamos que desea convertir los datos de la secuencia de nucleótidos en una secuencia de péptidos. En este ejemplo se supone que el primer codón empieza en el primer carácter de la secuencia de nucleótidos.

Puede invocar la función con una sentencia values. La única entrada es una secuencia de nucleótidos, como en el ejemplo siguiente:

```
values db2ls.LSNuc2Pep('TTTTTCTTATTGTCTTCCTCATCGTATTTCTTATGTTGCTGATGT')
```

El resultado de la sentencia anterior es una secuencia de péptidos que utiliza los símbolos de aminoácidos estándares:

```
FFLLSSSSYFLCC*C
```

Si desea la conversión del marco de lectura +2, utilice el ejemplo siguiente:

```
values LSNuc2Pep(SUBSTR('TTTTTCTTATTGTCTTCCTCATCGTATTTCTTATGTTGCTGATGT',2))
```

El entero de la sentencia indica la posición inicial de la búsqueda del codón.

A continuación se proporciona un ejemplo sobre cómo utilizar esta función como un predicado en una consulta.

```
SELECT *
  FROM proteindata
  WHERE peptideseq=DB2LS.LSNuc2Pep('TTTTTCTTATTGTCTTCCTCATCG
                                   TATTTCTTATGTTGCTGATGT');
```

El resultado se muestra en la Tabla 37.

Tabla 37. Resultados utilizando la función LSNuc2Pep como un predicado

ID	PROTEINNAME	PEPTIDESEQ
1	proteinA	FSYCLPHRISYVAD

El ejemplo siguiente convierte una secuencia de nucleótidos en una secuencia de péptidos utilizando una tabla de conversión externa. El primer parámetro es la secuencia de nucleótidos y el segundo parámetro es la vía de acceso a la tabla de conversión externa.

```
values db2ls.LSNuc2Pep('TTTTCTTATTGTCTTCCTCATCGTATTTCTTATGTTGCTGATGT',
                       'C:\translation.txt')
```

El resultado de la sentencia anterior utilizando esta tabla de conversión determinada es la cadena siguiente:

```
FSYCLPHRISYVAD
```

El ejemplo siguiente combina dos de las funciones definidas por el usuario para demostrar las utilizaciones adicionales de las funciones:

```
values DB2LS.LSNuc2Pep(DB2LS.LSRevCompNuc('TTT..')
```

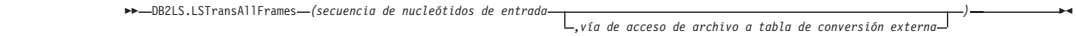
Tenga en cuenta que el ejemplo anterior devuelve el mismo resultado que la consulta siguiente:

```
select * from table (DB2LS.LSTransAllFrames ('TTT..')) as t
 where t.readframe = -1
```

Información relacionada:

- "Función definida por el usuario LSRevNuc - ejemplo" en la página 78
- "Función definida por el usuario LSTransAllFrames - ejemplo" en la página 84
- "Función definida por el usuario LSNuc2Pep" en la página 81

Función definida por el usuario LSTransAllFrames



secuencia de nucleótidos de entrada
Una representación de cadena de caracteres válida que describe una secuencia de nucleótidos. La secuencia de entrada puede contener códigos de ambigüedad de la IUPAC. Una representación de cadena de caracteres debe tener el tipo de datos VARCHAR y una longitud real no superior a 32672 bytes.

vía de acceso a tabla de conversión externa
Si utiliza una tabla de conversión personalizada, incluya la información de vía de acceso de archivo para encontrar la tabla de conversión. El valor de cadena de la vía de acceso no debe tener más de 255 caracteres.

El nombre de esquema es DB2LS.

Utilice la función definida por el usuario LSTransAllFrames para producir un conjunto de secuencias de péptidos a partir de una secuencia de nucleótidos determinada. Estas secuencias de péptidos representan las conversiones posibles de la secuencia de nucleótidos de entrada, en cada uno de los 6 marcos. Esta función es útil cuando la entrada contiene errores o si se desconoce el marco de lectura.

El resultado de la función es una tabla con dos columnas. La primera columna tiene la etiqueta READFRAME y representa el marco que se utiliza para la conversión. Esta columna tiene un valor entero que representa la posición inicial de la conversión. Un entero negativo indica una conversión de la hebra opuesta. La segunda columna, llamada PEPTIDE, es una cadena de caracteres con un tipo de datos VARCHAR y una longitud real no superior a 10890 bytes que representa la secuencia de péptidos.

La función:

- Elimina los espacios de las secuencias de entrada.
- Ignora los nucleótidos externos fuera de un marco de lectura.
- Devuelve una salida nula si la entrada es una secuencia de nucleótidos nula.

Información relacionada:

- "Función definida por el usuario LSTransAllFrames - ejemplo" en la página 84
- "Función definida por el usuario LSNuc2Pep" en la página 81

Función definida por el usuario LSTransAllFrames - ejemplo

Supongamos que desea convertir una secuencia de nucleótidos de todos los seis marcos de lectura utilizando la tabla de conversión incorporada. El ejemplo siguiente muestra cómo llevarlo a cabo:

```
SELECT * FROM table(DB2LS.LSTransAllFrames('TTTTTCTTATTGTCTTCCTCATCG
                                           TATTTCTTATGTTGCTGATGT')) as t;
```

La consulta devuelve los péptidos en una tabla, tal como se muestra en el ejemplo siguiente:

Tabla 38. Resultado de la conversión de una secuencia de nucleótidos

READFRAME	PEPTIDE
1	FFLLSSSSYFLCC*C
2	FSYCLPHRISYVAD
3	FLIVFLIVFLMLLM
–1	TSAT*EIR*GRQ*EK
–2	HQQHKKYDEEDNKK
–3	ISNIRNTMRKTIRK

El ejemplo siguiente utiliza una tabla de conversión personalizada para convertir una secuencia de nucleótidos de todos los seis marcos de lectura.

```
SELECT * FROM table
  (DB2LS.LSTransAllFrames
   ('TTTTTCTTATTGTCTTCCTCATCGTATTTCTTATGTTGCTGATGT',
    'C:\msvs6\MyProjects\alin_udf\test\files\translation.txt')) as t;
```

La tabla resultante es igual que la del ejemplo anterior puesto que la secuencia de entrada es la misma y la tabla de conversión es la misma que la incorporada en la función.

El ejemplo siguiente combina dos de las funciones definidas por el usuario para demostrar las utilizaciones adicionales de las funciones:

```
values DB2LS.LSNuc2Pep(DB2LS.LSRevCompNuc('TTT..')
```

Tenga en cuenta que el ejemplo anterior devuelve el mismo resultado que la consulta siguiente:

```
select * from table (DB2LS.LSTransAllFrames ('TTT..')) as t
where t.readframe = -1
```

El ejemplo siguiente selecciona un marco de lectura específico de la salida producida por la función LSTransAllFrames.

```
SELECT * FROM
  TABLE(db2ls.LSTransAllFrames('TTTTTCTTATTGTCTTCCTCATCG
                               TATTTCTTATGTTGCTGATGT')) AS t
  WHERE t.readframe=-2
```

El resultado de esta consulta es:

Tabla 39. Utilización de la función de marco de lectura

READFRAME	PEPTIDE
–2	HQQHKKYDEEDNKK

Información relacionada:

- “Función definida por el usuario LSNuc2Pep – ejemplo” en la página 82
- “Función definida por el usuario LSRevNuc - ejemplo” en la página 78
- “Función definida por el usuario LSTransAllFrames” en la página 83

Formato de la tabla de frecuencia de codón

Una tabla de frecuencia de codón muestra la frecuencia con que los aminoácidos se convierten de nuevo en un codón determinado. La función definida por el usuario LSPep2ProbNuc utiliza la tabla de frecuencia de codón para determinar una secuencia de nucleótidos a partir de una secuencia de péptidos determinada.

La lista siguiente describe el formato del archivo de tabla de frecuencia de codón:

- Dos puntos contiguos indican el principio de la tabla. Cualquier texto anterior a los puntos es comentario. Los dos puntos contiguos son necesarios incluso si no hay ningún comentario antes de ellos.
- La tabla contiene las columnas siguientes:
  1. Am-Ácido: un código de tres letras para el símbolo del aminoácido.
  2. Codón: el codón para el símbolo de aminoácido.

3. Número: el número de ocurrencias de dicho codón en los genes a partir de los que se compila la tabla.
4. x/1000: el número esperado de ocurrencias del aminoácido, par de codones por 1000 conversiones en genes.
5. Fracción: la fracción de ocurrencias del codón en su familia de codones sinónimos.

El producto proporciona tablas de frecuencia de codón de ejemplo en el subdirectorio sqllib/samples/lifesci/ls_udfs.

Información relacionada:

- "Función definida por el usuario LSPep2ProbNuc" en la página 51
- "Tabla de frecuencia de codón - ejemplo" en la página 86

Tabla de frecuencia de codón - ejemplo

La Figura 2 muestra el formato de la tabla de frecuencia de codón de ejemplo.

```
Am-Ácido Codón   Número       x/1000   Fracción   ..

Gly     GGG      198,00       18,34        0,23
Gly     GGA       71,00        6,58        0,08
Gly     GGT       66,00        6,11        0,08
Gly     GGC      527,00       48,81        0,61

Glu     GAG      534,00       49,46        0,88
Glu     GAA       71,00        6,58        0,12
Asp     GAT       31,00        2,87        0,06
Asp     GAC      481,00       44,55        0,94

Val     GTG      396,00       36,68        0,47
Val     GTA       22,00        2,04        0,03
Val     GTT       44,00        4,08        0,05
Val     GTC      384,00       35,57        0,45

Ala     GCG      446,00       41,31        0,39
Ala     GCA       71,00        6,58        0,06
Ala     GCT      116,00       10,74        0,10
Ala     GCC      503,00       46,59        0,44
... (truncado)
```

Figura 2. Tabla de frecuencia de codón de ejemplo

Información relacionada:

- "Función definida por el usuario LSPep2ProbNuc" en la página 51

- "Formato de la tabla de frecuencia de codón" en la página 85

Formato de la tabla de conversión

Este tema describe el formato de una tabla de conversión utilizada por las funciones definidas por el usuario de ciencias de la vida LSPep2AmbNuc, LSTransAllFrames y LSNuc2Pep.

La lista siguiente describe el formato del archivo de tabla de frecuencia de codón:

- Dos puntos contiguos indican el principio de la tabla. Cualquier texto anterior a los puntos es comentario.
- Cada línea de la tabla consta de un símbolo de aminoácido de una sola letra, el nombre de aminoácido de tres letras, los codones no ambiguos, un signo de exclamación y los codones ambiguos. Un espacio en blanco separa cada palabra de la línea.
- Cada codón y símbolo de aminoácido debe aparecer sólo una vez en el archivo.
- Los codones de finalización se convierten en el símbolo '*'.
- Los codones formados por letras en minúsculas son codones de inicio.
- Los demás codones están en mayúsculas.
- Los codones que no tienen una conversión a un símbolo de aminoácido correspondiente se convierten al símbolo 'X'.

El producto proporciona tablas de conversión de ejemplo en el subdirectorio sqllib/samples/lifesci/ls_udfs.

Tabla de conversión - ejemplo

La Figura 3 en la página 88 muestra el formato de una tabla de conversión de ejemplo.

```
                Tabla de conversión estándar

Símbolo  3 letras    Codones                  ! IUPAC     ..

 A        Ala         GCT GCC GCA GCG          ! GCX
 B        Asx                                  ! RAY
 C        Cys         TGT TGC                  ! TGY
 D        Asp         GAT GAC                  ! GAY
 E        Glu         GAA GAG                  ! GAR
 F        Phe         TTT TTC                  ! TTY
 G        Gly         GGT GGC GGA GGG          ! GGX
 H        His         CAT CAC                  ! CAY
 I        Ile         ATT ATC ATA              ! ATH
 K        Lys         AAA AAG                  ! AAR
 L        Leu         TTG TTA CTT CTC CTA CTG  ! TTR CTX YTR     ; YTX
 M        Met         atg                      ! ATG
 N        Asn         AAT AAC                  ! AAY
 P        Pro         CCT CCC CCA CCG          ! CCX
 Q        Gln         CAA CAG                  ! CAR
 R        Arg         CGT CGC CGA CGG AGA AGG  ! CGX AGR MGR     ; MGX
 S        Ser         TCT TCC TCA TCG AGT AGC  ! TCX AGY         ; WSX
 T        Thr         ACT ACC ACA ACG          ! ACX
 V        Val         GTT GTC GTA GTG          ! GTX
 W        Trp         TGG                      ! TGG
 X        Xxx                                  ! XXX
 Y        Tyr         TAT TAC                  ! TAY
 Z        Glx                                  ! SAR
 *        End         TAA TAG TGA              ! TAR TRA         ; TRR
```

Figura 3. Tabla de conversión de ejemplo

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 ofrecidos en los EE.UU. Es posible que IBM no comercialice en todos los países algunos productos, servicios o características descritos en este manual. Consulte al representante local de IBM para obtener información sobre los productos y servicios que actualmente pueden adquirirse en su zona. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se pueda utilizar dicho producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ninguno de los derechos de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patentes en tramitación que afecten al tema tratado en este documento. La posesión de este documento no confiere ninguna licencia sobre dichas patentes. Puede realizar consultas sobre licencias escribiendo a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
EE.UU.

Para realizar consultas sobre licencias referentes a información de doble byte (DBCS), puede ponerse en contacto con el Departamento de Propiedad Intelectual de IBM de su país/región o escribir a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokio 106, Japón

El párrafo siguiente no es aplicable al Reino Unido ni a ningún país/región en donde tales disposiciones sean incompatibles con la legislación local: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN “TAL CUAL”, SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN DE DERECHOS, COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN DETERMINADO. Algunos estados no permiten la exclusión de garantías

expresas o implícitas en determinadas transacciones, por lo que es posible que esta declaración no sea aplicable en su caso.

Esta publicación puede contener inexactitudes técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; dichos cambios se incorporarán a las nuevas ediciones de la publicación. IBM puede efectuar, en cualquier momento y sin previo aviso, mejoras y cambios en los productos y programas descritos en esta publicación.

Las referencias hechas en esta publicación a sitios Web que no son de IBM se proporcionan sólo para la comodidad del usuario y no constituyen un aval de esos sitios Web. La información contenida en esos sitios Web no forma parte de la información del presente producto IBM y el usuario es responsable de la utilización de dichos sitios Web.

IBM puede utilizar o distribuir cualquier información que se le facilite de la manera que considere adecuada, sin contraer por ello ninguna obligación con el remitente.

Los licenciatarios de este programa que deseen obtener información sobre él con el fin de habilitar: (i) el intercambio de información entre programas creados de forma independiente y otros programas (incluido éste) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, 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 IBM Corp. Programas de ejemplo. © Copyright IBM Corp. *_entre el o los años_*. Reservados todos los derechos.

Marcas registradas

Los términos siguientes son marcas registradas de International Business Machines Corporation en los EE.UU. y/o en otros países:

IBM
AIX
DB2
Domino
Informix
Lotus
Lotus Notes
QuickPlace
WebSphere

Los términos siguientes son marcas comerciales o registradas de otras empresas:

Microsoft, Windows, Windows NT y el logotipo de 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.

Java y todas las marcas registradas y logotipos basados en Java son marcas registradas de Sun Microsystems, Inc. en los Estados Unidos y/o en otros países.

Otros nombres de empresas, productos o servicios pueden ser marcas registradas o marcas de servicio de terceros.

Índice

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).