WebSphere® Information Integration

IBM

Version 9

Configuration Guide for Federated Data Sources

SC19-1034-00

WebSphere® Information Integration

IBM

Version 9

Configuration Guide for Federated Data Sources

SC19-1034-00

Note

Before using this information and the product that it supports, be sure to read the general information under "Notices and trademarks" on page 499.

Contents

Chapter 11. Informix data sources . . 213

Chapter 12. Microsoft SQL Server data sources 227

Chapter 13. ODBC data sources . . . 239

Chapter 14. OLE DB data sources . . 257

Chapter 15. Oracle data sources . . . 263

Chapter 16. Script data sources . . . 277

Chapter 17. Sybase data sources . . . 293

Chapter 18. Table-structured file data sources 307

Chapter 19. Teradata data sources 319

Chapter 1. Planning to configure access to data sources

Plan the federated data source configuration

Before you configure the federated server to access data sources, you should plan out the configuration that you want for your federated system.

Federated object naming rules

There are rules that you must follow when you name federated database objects.

Federated database objects include:

- Data type mappings
- Function mappings
- Index specifications
- Nicknames
- Server definitions
- User mappings
- Wrappers

When you register these objects, you must give the objects a name. Unless you use a delimiter with the name, the name of a federated object must begin with one of the following:

- A letter, including a valid accented letter, such as Ö
- A multibyte character, except a multibyte space for multibyte environments

The name of a federated object cannot begin with a number or with the underscore character.

When you specify the name as a delimited identifier, you can use any character anywhere in the name. The delimiter that you use with federated objects is the quotation mark, for example ″object_name″. Whenever you use the object name, for example in an SQL statement, you must include the delimiter.

Names without quotation marks are converted to uppercase.

The name can also include the following characters:

- A through Z
- 0 through 9
- @, #, $, and _ (underscore)

Federated object names cannot exceed 128 bytes.

Options (such as wrapper, server, and nickname options) and option settings are limited to 2048 bytes.

Preserving case-sensitive values in a federated system

Values that are case-sensitive at the data source can be preserved when you register these values with the federated server.

In a federated system you occasionally need to specify values that are case-sensitive at the data source, such as user IDs and passwords. To ensure that the case is correct when these values are passed to the data source, follow these guidelines:

- Specify the values in the required case and enclose them in the proper quotation marks. Double quotation marks are optional for object names, such as the name of a wrapper or nickname. Single quotation marks are required for option values, such as the REMOTE_AUTHID and REMOTE_PASSWORD user mapping options.
- For user IDs and passwords, you can set the FOLD_ID and FOLD_PW server options to automatically convert the values to the proper case. With this option, you do not have to remember the required case for each data source. You can type the values in any case and they will be converted automatically.

From a UNIX operating system command prompt

If you enclose a case-sensitive value in quotation marks at the federated server operating system command prompt, you must ensure that the quotation marks are parsed correctly:

SQL statements that contain double quotation marks, but that do not contain single quotation marks

If the SQL statement contains double quotation marks, but does not contain single quotation marks, you enclose the entire statement in single quotation marks.

For example, if you want to issue this SQL statement:

```
CREATE NICKNAME my_nickname FOR my_server."owner".my_table"
```

You enter the following text at the UNIX command prompt:

```
DB2 'CREATE NICKNAME my_nickname FOR my_server."owner".my_table"'
```

SQL statements that contain single quotation marks, but that do not contain double quotation marks

If the SQL statement contains single quotation marks, but does not contain double quotation marks, you enclose the entire statement in double quotation marks.

For example, if you want to issue this SQL statement:

```
CREATE USER MAPPING FOR USER SERVER my_server
    OPTIONS(REMOTE_AUTHID 'my_id', REMOTE_PASSWORD 'my_password')
```

You enter the following text at the UNIX command prompt:

```
DB2 "CREATE USER MAPPING FOR USER SERVER my_server
    OPTIONS(REMOTE_AUTHID 'my_id', REMOTE_PASSWORD 'my_password')"
```

SQL statements that contain both double and single quotation marks

If the SQL statement contains both single and double quotation marks:

- Enclose the entire statement in double quotation marks
- Precede each double quotation mark in the statement with a backward slash.

For example, to issue this SQL statement:

```
CREATE USER MAPPING FOR "local_id" SERVER my_server
    OPTIONS(REMOTE_AUTHID 'my_id', REMOTE_PASSWORD 'my_password')
```

You enter the following text at the UNIX command prompt:

```
DB2 "CREATE USER MAPPING FOR \"local_id\" SERVER my_server
    OPTIONS(REMOTE_AUTHID 'my_id',  REMOTE_PASSWORD 'my_password')"
```

The above examples assume that you are entering SQL statements from the UNIX command prompt and are passing the statement to the DB2 command, without the -f option. If you enter the SQL statements from a file using the DB2 command with the -f option, then you should enter the statements as shown in the first occurrence of each example.

From a Windows operating system command prompt

To preserve case-sensitive values on Windows, precede each double quotation mark with a backward slash. For example, you want to create the nickname *nick1* for the Microsoft SQL Server table *weekly_salary*. The table resides in the *NORBASE* database. The local schema is *my_schema*.

At the Windows command prompt on your federated server, you type:

```
DB2 CREATE NICKNAME nick1
    FOR NORBASE.\"my_schema\".\"weekly_salary\"
```

From the DB2 CLP or from an application program

When you specify a value from the DB2 command line prompt (CLP) or in an application program, you can preserve case-sensitive values by enclosing the values in the proper quotation marks.

For example, you want to create a user mapping for the user ID *local_id*. The remote user ID *my_id* and the remote password is *my_password*. You want all three of these values to be preserved in lowercase. At the DB2 command prompt you type:

```
CREATE USER MAPPING FOR "local_id" SERVER my_server
    OPTIONS(REMOTE_AUTHID 'my_id', REMOTE_PASSWORD 'my_password')
```

Update data source statistics

If you plan to access a relational data source, you should update the statistics at the remote data source before you configure the federated server to access the data source. By ensuring that the remote data source has current statistics, you can improve query performance.

The federated server relies on the data source statistics that are stored in the federated database to optimize query processing. These statistics are gathered when you create a nickname for a data source object. The federated database verifies the presence of the object at the data source, and then attempts to gather existing statistical data for the data source. Information useful to the query optimizer is read from the data source catalogs and added to the system catalog in the federated database.

Because some or all of the catalog information from the data source might be used by the query optimizer, it is recommended that you update statistics at the data source before you create a nickname. Use the command at the data source that is equivalent to the DB2 RUNSTATS command to update the data source statistics.

The federated database retrieves that statistical information for a data source object when you create a nickname for the object. If the data source updates its catalog statistics for an object after your create the nickname, the changes in the statistical information are not propagated to the system catalog in the federated database. To make sure that the system catalog in the federated database reflects the current statistics for the remote data source object, you must request that the federated server update the statistics. You can update the statistics by running the nickname statistics (NNSTAT) stored procedure.

Tip: Identify the data source objects that you want to access. These are objects that you will create nicknames for. Determine which of the data sources allow you to update statistics. List those data sources in the data source statistics table in the planning checklist.

Choose the correct wrapper

For most data sources there is only one wrapper that you can use to access the data source. However, for some data sources you have a choice as to which wrapper you use to access the data in the data source.

Data sources that support the ODBC API

You can access data sources that support the ODBC API either by using the wrappers that are designed for those data sources or by using the ODBC wrapper. Examples of these data sources include Oracle, Microsoft Excel, Microsoft SQL Server. Typically, query performance is better when you use the wrappers that are specifically designed for these data sources.

Use the ODBC wrapper to access any data source that has an ODBC driver but is not supported by specific data source wrappers that are included with WebSphere Federation Server. For example, use the ODBC wrapper to access RedBrick data sources.

DB2 for Linux, UNIX, and Windows data sources

Do not use the ODBC wrapper to access DB2 for Linux, UNIX, and Windows data sources. Using the ODBC wrapper to access DB2 for Linux, UNIX, and Windows data sources is not supported. Use the DRDA wrapper to access DB2 for Linux, UNIX, and Windows data sources.

Excel data sources

Depending on your needs, you can use the ODBC wrapper to access Excel data instead of using the Excel wrapper.

Informix data sources

Do not use the ODBC wrapper to access Informix data sources. Using the ODBC wrapper to access Informix data sources is not supported. To access Informix data sources, use the Informix wrapper.

Tip: Identify the wrappers that you will create for your federated system in the wrapper table in the planning checklist.

Methods of accessing Excel data

You can access data in Microsoft Excel worksheets by using either the Excel wrapper or the ODBC wrapper.

To query Excel data, both wrappers require a federated server that can open and read the worksheets in the Excel workbook. Therefore, the Excel workbook must be on the same computer as the federated server or on a network accessible drive.

If you use the Excel wrapper, the Excel application must be installed on the federated server.

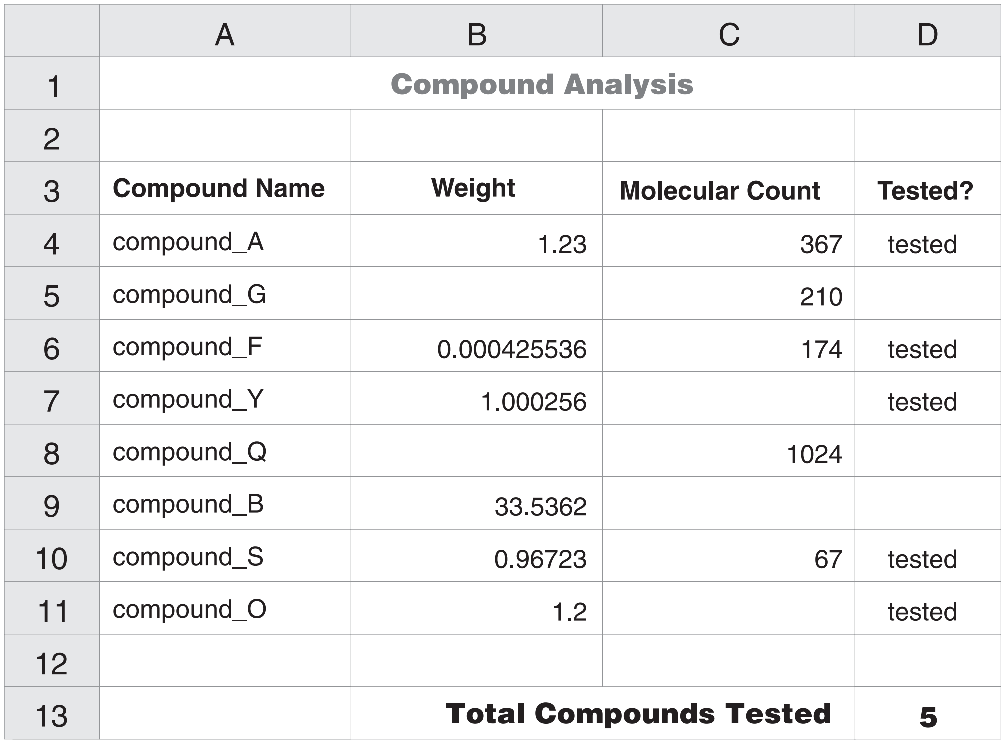
If you use the ODBC wrapper, the Excel ODBC driver must be on the federated server. This driver is installed automatically with Microsoft Windows®. The Excel application does not need to be installed on the federated server.

Each wrapper imposes some requirements on the location and layout of the data in the Excel workbooks. With the Excel wrapper, only the data in the first worksheet in the workbook can be accessed. With the ODBC wrapper, you can access data from any worksheet in the workbook.

The following examples show the worksheet layout requirements for these two wrappers.

Example of a worksheet that contains rows of labels and a formula

This example shows a worksheet that contains several rows of labels at the top of the worksheet, blank rows, and a formula in row 13. To access the data in the worksheet, you must identify the range of cells that you want to access.



	A	B	C	D
1	**Compound Analysis**			
2				
3	**Compound Name**	**Weight**	**Molecular Count**	**Tested?**
4	compound_A	1.23	367	tested
5	compound_G		210	
6	compound_F	0.000425536	174	tested
7	compound_Y	1.000256		tested
8	compound_Q		1024	
9	compound_B	33.5362		
10	compound_S	0.96723	67	tested
11	compound_O	1.2		tested
12				
13		**Total Compounds Tested**		**5**

Figure 1. A worksheet that contains several rows of labels and a formula

If you use the Excel wrapper

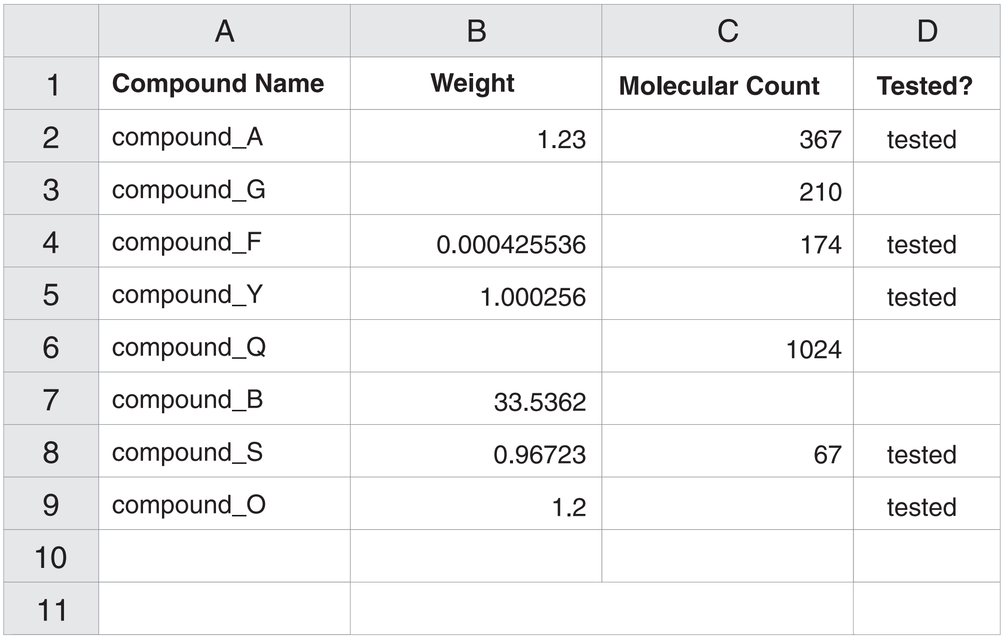
You specify the range of cells in the CREATE NICKNAME statement by using the RANGE option. Include only the data in the range that you specify. Do not include any column labels in the range. Cells that contain formulas, such as SUM, return the result of the formula and not the formula. Unless you want the formula results returned, do not include the cells that contain formulas in the range. In this example, the range of cells that you include in the RANGE option is A4:D11.

If you use the ODBC wrapper

You must create a name for the range of cells to explicitly designate the location of the data within the worksheet. Excel refers to this range of cells as a *named range*. The Excel ODBC driver recognizes only one row of labels, the first row in the range. No blank rows are allowed between the labels and the data. The named range must include only one row of column labels. You specify the named range in the CREATE NICKNAME statement. You must include one row of column labels in the range that you name. If you do not include one row of column labels in the named range, the first row of data is treated as column labels. Cells that contain formulas, such as SUM, return the result of the formula and not the formula. Unless you want the formula results returned, do not include the cells that contain formulas in the range. In this example, the range of cells that you name is A3:D11.

Example of a worksheet that contains one row of labels

This example shows a worksheet that contains only one row of column labels at the top of the worksheet. The layout does not include extra rows with labels, blank rows, or cells with formulas.



	A	B	C	D
1	**Compound Name**	**Weight**	**Molecular Count**	**Tested?**
2	compound_A	1.23	367	tested
3	compound_G		210	
4	compound_F	0.000425536	174	tested
5	compound_Y	1.000256		tested
6	compound_Q		1024	
7	compound_B	33.5362		
8	compound_S	0.96723	67	tested
9	compound_O	1.2		tested
10				
11				

Figure 2. A worksheet that contains one row of column labels in row 1

If you use the Excel wrapper

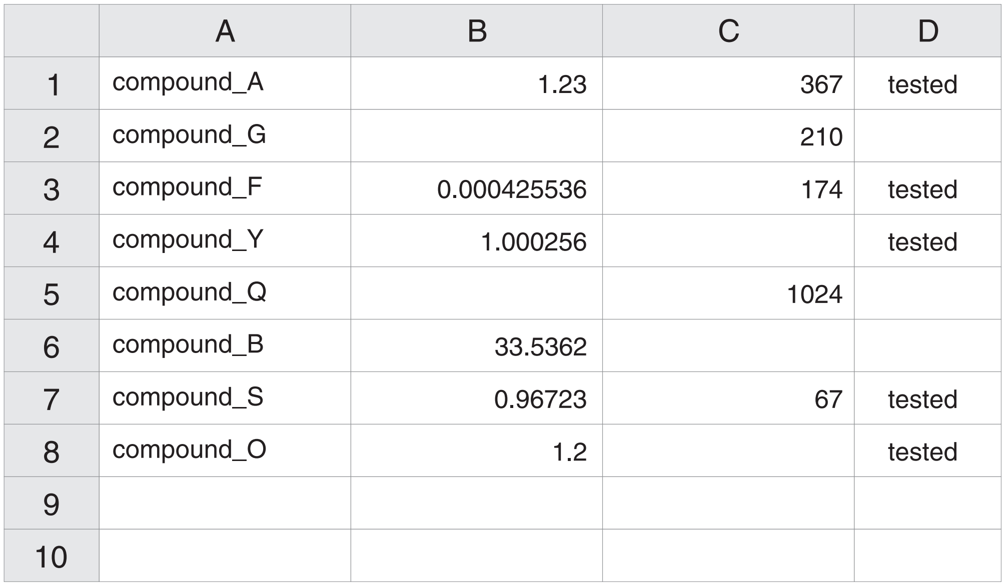
You must specify the range of cells in the CREATE NICKNAME statement by using the RANGE option. The range cannot include the column labels in row 1. The range of cells that you would specify is A2:D9.

If you use the ODBC wrapper

You can access this data without creating a named range. You specify the worksheet name in the CREATE NICKNAME statement. The wrapper reads the first nonblank row as labels and uses the information as column names for the nickname. Subsequent rows are read as data.

Example of a worksheet that contains only data

This example shows a worksheet that contains only data. There are no rows of column labels, no blank rows, and no cells with formulas.



	A	B	C	D
1	compound_A	1.23	367	tested
2	compound_G		210	
3	compound_F	0.000425536	174	tested
4	compound_Y	1.000256		tested
5	compound_Q		1024	
6	compound_B	33.5362		
7	compound_S	0.96723	67	tested
8	compound_O	1.2		tested
9				
10				

Figure 3. A worksheet that contains only data

If you use the Excel wrapper
If the data is in the first worksheet in the workbook, the wrapper will access the data without using the RANGE option. If the data is in another worksheet in the workbook, you must specify the RANGE option in the CREATE NICKNAME statement.

If you use the ODBC wrapper
When you use the ODBC wrapper to access Excel data, the wrapper is limited by what the Excel ODBC driver supports. The Excel ODBC driver requires a specific format for the worksheet. The driver assumes that the first nonblank row contains the column labels. If the first nonblank row contains data, the data in that row is treated as the column labels for the remaining data. If the worksheet does not contain a row of column labels, the first row is used as the labels and not as data. In effect, you lose the first row of data. You can overcome this requirement by modifying your worksheet. Insert a new row before the data and add labels for each column of data, so that it looks like the example that contains one row of labels.

Plan the user mappings

There are two methods for specifying user mappings with federated systems. You can use an external repository, such as LDAP, to store the user mappings or you can create the user mappings in the federated database catalog.

When a federated server needs to pushdown a request to a data source, the server must first establish a connection to the data source. For some data sources, the federated server establishes a connection by using a valid user ID and password to that data source. For these data sources, you must define an association between the federated server user ID and password and the data source user ID and password. This association must be created for each user ID that will be using the federated system to send distributed requests. This association is called a *user mapping*.

Using an external repository

You can store the user mappings in an external repository, such as a Lightweight Directory Access Protocol (LDAP) server. You must develop a plugin that retrieves the user mappings from the repository and passes the authentication information to the federated server. Federation includes a sample plugin that is designed for an LDAP repository. To work with other types of external repositories, you can develop your own plugin or extend the sample LDAP plugin.

Creating the mappings in the database catalog

You can use the DB2® Control Center to create a user mapping for a group of users that will access a data source with the same user ID and password.

Tip: Identify the user IDs that require a user mapping between the federated server and the data source. List the federated server user IDs and corresponding data source user IDs in the user mapping table in the planning checklist.

Plan the data type mappings

You must consider the mappings that you want between the data source data types and the federated database data types. Depending on your data source, there might be a set of default mappings which you can use or you might need to specify the mapping information.

Data source data types are referred to as *remote* data types, and federated database data types are referred to as *local* data types.

For some data sources, the wrappers contain the default mappings between the data source data types and federated database data types. When you create a nickname for a data source object, information about the columns is stored in the federated database system catalog. The data types for the columns comes from the default forward data type mappings.

Some of the nonrelational wrappers create all of the columns required to access a data source. These are called *fixed columns*.

For other data sources, you must specify some or all of the column information and the data type when you create the nickname.

Your applications might require data type mappings that are different than the default mappings. For the wrappers that allow you to specify data type mappings, you can override the default mappings to:

- Change a type mapping for all data source objects located on a specific server
- Change a type mapping for a specific data source object
- Change a type mapping for a specific data source type
- Change a type mapping for a specific data source type and version

Use the CREATE TYPE MAPPING statement to define new data type mappings. Mappings you create are stored in the federated database global catalog SYSCAT.TYPEMAPPINGS view.

Change a data type mapping *before* you create nicknames for the data source objects. When you create a nickname for a data source object, the federated server populates the global catalog with information about the table. This information includes the nickname, the data source table name, the column names and the data types that are defined for each table column.

Only nicknames created after a mapping is changed reflect the new type mapping. Nicknames created before the mapping is changed will use the default data type mapping.

If you create the data type mappings after you create the nicknames, you must either alter each nickname to reflect the new mapping or drop and create the nicknames again.

Important: If a data source table contain columns that are distinct or user-defined data types, you have two choices:

- You can create the type mapping in the federated database before you create a nickname for that data source table. By creating the type mappings before you create the nickname, the federated server will know what data type to map these columns to. If the mappings for these distinct or user-defined data types are not created before you issue the CREATE NICKNAME statement, you will receive an error.
- If the columns in the data source table meet either of the following conditions:
  - The columns are user-defined data types that are based on system or built-in data types
  - The columns have attributes that are not supported for data type mappings

  You can create a view at the data source in which the columns are associated with or *cast* to the underlying built-in data type. Then create a nickname for the view instead of for the table.

Tip: Identify the data type mappings that you want to define new mappings for. List the data sources and the type mappings you want to create in the data type mappings table in the planning checklist

Plan the function mappings

In a federated system, there are default function mappings between the existing built-in data source functions and built-in federated database functions.

You might need to create a function mapping to override a default function mapping or when a function mapping does not exist.

For most data sources, the default function mappings are in the wrappers. For some nonrelational data sources, you cannot alter the default function mappings.

There are several reasons why you might need to create a function mapping:

- You want to use a new data source function, such as a new built-in function or a new user-defined function. The mapping that you create is between the data source function and a counterpart function at the federated database.
- You want to use a data source function and the federated database does not have a counterpart function. In this situation, before you create the function mapping you must create a function template in the federated database.

Tip: Determine if you need to create function mappings for your data sources. List the function mappings needed in the function mappings table in the planning checklist.

Checklist for planning your federated system configuration

Configuring your federated system is easier when you follow this planning checklist. The checklist helps you to optimize the configuration of your federated system.

Checklist: Federated object naming rules

Are you familiar with the naming rules for federated objects?

Checklist: Preserving case-sensitive values

Relational data sources only

Do you intend to set the FOLD_ID and FOLD_PW server options to preserve case for user ID and password values sent to the data sources? Use the following table to identify which server definitions you will apply these options to.

Table 1. Planning checklist: FOLD_ID and FOLD_PW server options to set for the federated system

Data source	What name will you specify for the server in the server definition for this data source?	What setting will you specify for the FOLD_ID server option?	What setting will you specify for the FOLD_PW server option?

Checklist: Data source statistics

Relational data sources only

In the following table, list the data sources that will be part of your federated system. Indicate the data sources that you will update the statistics for before you configure the federated server to access the data source. DB2 for Linux, UNIX, and Windows is listed in this table as an example.

Table 2. Planning checklist: Data sources statistics to update for the federated system

Data source	Does this data source maintain catalog information? (Y/N)	Will you update statistics for this data source? (Y/N)	Name of the utility that the data source uses to update statistics
DB2 for Linux, UNIX, and Windows	Y	Y	RUNSTATS

Checklist: Wrappers

In the following table, identify the wrappers that you will create.

Table 3. Planning checklist: Wrappers to create for the federated system

Data source	Default wrapper name	Name that you will give the wrapper
BioRS	none	
BLAST	none	
Business applications (WebSphere Business Integration wrapper)	none	
DB2 for Linux, UNIX, and Windows® DB2 Universal Database for z/OS and OS/390® DB2 Universal Database for iSeries DB2 Server for VM and VSE	DRDA	
Entrez	none	
Excel	none	
HMMER	none	
Informix	INFORMIX	
Microsoft® SQL Server	MSSQLODBC3	
Oracle	NET8	
ODBC	ODBC	
OLE DB	OLEDB	
Script	none	
Sybase	CTLIB	
Table-structured files	none	
Teradata	TERADATA	
Web services	none	
XML	none	

Checklist: User mappings

User mappings stored in an external repository

Have you identified the type of external repository that you will use?

What is the name of the plugin that you will create?

____

Where will you specify the plugin name? On the Wrapper __________ On the Server definition __________

User mappings stored in the system catalog

In the following table, identify the federated server user IDs and corresponding user IDs for *each* data source that will be part of the federated system.

Table 4. Planning checklist: User mappings to create for the federated system

		Data source __________	**Data source** __________	**Data source** __________
User name	Federated server user ID	User ID	User ID	User ID

Checklist: Data type mappings

In the following table, identify the data source data types and the corresponding federated server data types that you need to create a mapping for.

Table 5. Planning checklist: Data type mappings to create for the federated system

Data source	**What name will you specify for the remote server in the server definition for this data source?**	**Data source data type**	**Federated server data type**

Chapter 2. Federated server and database

Checking the setup of the federated server

You can avoid potential configuration problems by checking the key settings on the federated server.

When you install WebSphere® Federation Server, the software attempts to configure the federated server for you. If problems occur during the installation, the federated server is not setup properly.

About this task

To check the setup of the federated server:

1. Confirm the link-edit of the wrapper library files to the data source client software (UNIX®).
2. Check that the FEDERATED parameter is set to YES.

After you check the setup of the federated server, you must create a federated database.

Confirming the link-edit of the wrapper library files (UNIX)

On federated servers that run UNIX, some wrappers must be link-edited with the client software for the data source.

About this task

The link-edit step is attempted when you install WebSphere Federation Server. The link-edit step creates a wrapper library for each data source that the federated server will communicate with.

This task applies to only the following data sources:

- Informix®
- Microsoft® SQL Server
- Oracle
- Sybase
- Teradata

Before you configure the federated server and database to access data sources, you should confirm that the link-edit of the wrapper library files was successful.

Procedure

To confirm the link-edit of the wrapper library files:

Determine the status of the link-edit:

- If the link-edit was successful, the wrapper library file appears in the directory where WebSphere Federation Server is installed.
- If the link-edit failed, check the error message file in the directory where WebSphere Federation Server is installed.

- If the link-edit was not performed, neither the library file nor the message file appears in the directory where WebSphere Federation Server is installed. You will have to manually run the link script.

Checking the wrapper library files (UNIX)

You must confirm that a wrapper library file exists on the federated server for each data source that you want to access.

About this task

This task applies to only the following data sources:

- Informix
- Microsoft SQL Server
- Oracle
- Sybase
- Teradata

Procedure

To check for the wrapper library files:

Check for the library files in the default directory path where WebSphere Federation Server is installed. If the library files are not in that directory, you must manually link the wrapper libraries to the data source client software.

The directory path for the wrapper library depends on the data source.

Wrapper library files

The wrapper library files are required so that you can access the data sources.

For some data sources, the wrapper library files are added to the federated server when you install WebSphere Federation Server. For other data sources a link-edit script must be run to create the library files. There are three wrapper library files for each data source.

The wrapper library files are required when you register the wrapper for the data source.

The wrapper library files are added to the default directory path. There should be a set of wrapper library files for each of the data sources that you want to access.

Checking the link-edit message files (UNIX)

If the link-edit fails, errors are listed in the message file in the library directory.

About this task

The existence of a message file does not mean that the link-edit failed. There is a message file in the library directory even if the link-edit is successful.

You must open the message file to determine if the link-edit failed.

This task is required for only the following data sources:

- Informix
- Microsoft SQL Server

- Oracle
- Sybase
- Teradata

Procedure

To determine if the link-edit failed:

1. Open the link-edit message files. The link-edit message files are in the directory where WebSphere Federation Server is installed, in the lib32 or lib64 subdirectory.

   The names of the link-edit message files are listed in the following table.

Table 6. Link-edit message file names by data source

Data source	Message file names
Informix	djxlinkInformix.out
Microsoft SQL Server	djxlinkMssql.out
Oracle	djxlinkOracle.out
Sybase	djxlinkSybase.out
Teradata	djxlinkTeradata.out

2. Resolve the link-edit failure.

   There are several reasons why the link might fail when you setup the federated server:

   - If the data source client software is not installed before the link-edit is attempted, then the link-edit will fail. For example, if you do not install the Informix client software before you install the DB2 server software, the link-edit will fail. Likewise, if you do not install the Sybase Open Client software before you install WebSphere Federation Server, the link-edit will fail. In these situations, you will have to perform the link manually.
   - Verify that the version of the data source client software is supported. If the version of the data source client software that you have installed is not supported, the link-edit will fail. You will have to install a client version that is supported and then perform the link manually.

Manually linking the wrapper libraries to the data source client software

If the wrapper library files are not in the directory path, you must manually link the wrapper libraries.

Before you begin

You need root authorization to run the link scripts.

The client software for the data sources that you want to access must be installed and configured on the federated server.

For the djxlink*xxx* scripts to issue their messages in your language, at least one database instance must exist on the federated server. If an instance does not exist on the federated server, the scripts will still work. However, the scripts will issue all messages in English.

About this task

This task applies only to the following data sources:

- Informix
- Microsoft SQL Server
- Oracle
- Sybase
- Teradata

Procedure

To link the wrapper libraries to the data source client software:

1. Decide which method you want to use to perform the link:

Method	Step
Use the product CDs.	• For Informix data sources, run the DB2 for Linux®, Unix, and Windows® installation again and specify the Custom installation option. • For Microsoft SQL Server, Oracle, Sybase, and Teradata data sources run the WebSphere Federation Server installation again. From the launchpad, click **Install Products** and follow the instructions in the wizard.
Run the link-edit scripts.	1. Open a UNIX command prompt and run the link-edit script for each data source that you want to access. The names of the link-edit script are: • djxlinkInformix • djxlinkMssql • djxlinkOracle • djxlinkSybase • djxlinkTeradata 2. Issue the db2iupdt command on each federated database instance to enable federated access to the data sources. There is another script, the **djxlink** script, that attempts to create a wrapper library for every data source that is supported by WebSphere Federation Server. If you run **djxlink** script and have the client software for only some of the data sources installed, you will receive an error message for each of the data sources that you do not have installed.

2. After the link is performed, check the permissions on the wrapper libraries. Make sure that the libraries can be read and run by the database instance owners.

Checking the FEDERATED parameter

The FEDERATED parameter must be set to YES to enable the federated server to access to the data sources.

Before you begin

- WebSphere Federation Server must be installed on a server that will act as the federated server

About this task

Before you add data sources to the federated server and database, you should check the FEDERATED parameter setting.

Procedure

To check the FEDERATED parameter setting:

1. Issue the following DB2 command to display all of the parameters and their current settings:
   ```
   GET DATABASE MANAGER CONFIGURATION
   ```
2. Check the CONCENTRATOR parameter setting. The CONCENTRATOR parameter and the FEDERATED parameter cannot be configured to YES at the same time. If the CONCENTRATOR parameter is set to YES, change the setting to NO. Issue the following DB2 command to change the setting:
   ```
   UPDATE DATABASE MANAGER CONFIGURATION USING CONCENTRATOR NO
   ```
3. Check the FEDERATED parameter setting. If the FEDERATED parameter is set to NO, change the setting to YES. Issue the following DB2 command to change the setting:
   ```
   UPDATE DATABASE MANAGER CONFIGURATION USING FEDERATED YES
   ```

Creating a federated database

After you set up the federated server, the instance owner must create a database on the federated server instance that will act as the federated database.

Before you begin

- You must have SYSADM or SYSCTRL authority to create a database.
- WebSphere Federation Server must be installed on a server that will act as the federated server

About this task

You must create a federated database before you can configure the federated server to access your data sources.

You can create the federated database by using the DB2 Control Center or from the DB2 command line.

Procedure

To create the federated database:

1. Determine the code set and collating sequence that you want to specify when you create the federated database.

2. Determine the method that you want to use to create the federated database:

Method	Step
Using the DB2 Control Center	Right-click on the **Databases** folder and click **Create –>Database Using Wizard**. The Create Database Wizard opens. Complete the steps in the wizard.

From the DB2 command line	Issue the CREATE DATABASE command.
	For example:
	`CREATE DATABASE` *`federated`*
	`USING CODESET` *`ISO8859-15`*
	`TERRITORY` *`BR`*

Creating the database:

- Initializes a new database
- Creates the three initial table spaces
- Creates the system tables
- Allocates the recovery log

If your instance uses a multiple partition configuration, all of the partitions that are listed in the db2nodes.cfg file are affected when you create the database. The database partition from which this command is issued becomes the catalog partition for the new database.

Federated database code sets and collating sequences

When you create a federated database, you must decide the language that you want your data stored in and the order in which character data is sorted in a database.

The database is created using the settings of the application that creates the database unless you explicitly specify a territory, code set, and collating sequence. For example, if you create the federated database from a client that uses the ISO8859-15 code set, the database is with the same code set.

Recommendation: If the remote data sources that you need to connect to are using different or incompatible code sets, define the federated database as a Unicode database. To define the federated database as a Unicode database, specify USING CODESET *UTF-8* in the CREATE DATABASE command.

Code sets, code pages, and territories

You can specify the language for the federated database by including the CODESET and TERRITORY options on the CREATE DATABASE command.

A *code set* is an ISO term for unique bit patterns that are mapped to the characters contained in a specific natural language. After you create the database, you cannot change the specified code set. When you choose a code set, make sure that it can encode all the characters in the language that you will be using.

Code sets are mapped to IBM-defined *code pages*.

A *territory code* is used to provide region-specific information for the code set that you specify. After you create the database, you cannot change the specified territory.

Collating sequences

You specify the collating sequence for the federated database by including the COLLATE USING option on the CREATE DATABASE command.

For example:

```
CREATE DATABASE federated USING CODESET UTF-8 COLLATE USING UCA400-NO
```

After you create the database, you cannot change the collating sequence.

For relational data sources, if the federated database and the data source use the same collating sequence, you should set the COLLATING_SEQUENCE server option to ’Y’. Setting the COLLATING_SEQUENCE server option to ’Y’ informs the federated server that the collating sequences of the federated database and the data source match. You set the COLLATING_SEQUENCE server option when you create the server definitions for the relational data sources.

The relational data sources that support the COLLATING_SEQUENCE server option are:

- DB2 family
- Informix
- Microsoft SQL Server
- ODBC
- OLE DB
- Oracle
- Sybase
- Teradata

Federated database national language considerations

When you create the federated database, you must decide which code set and territory to use. You can improve performance by using the same collating sequence that your data sources use.

For many data sources, the wrapper performs the following national language support (NLS) tasks when the wrapper connects to the data source:

1. Determines the code page and territory of the federated database.
2. Maps the code page and territory to a data source client locale name, if supported by the data source.
3. Depending on the data source, sets an environment variable. calls a data source API to tell the data source what the client locale is, or prepares to perform code set conversion.
4. The data is converted:
   - For data sources that perform code page conversion, the data source then converts character data between the code page of the remote database and the code page of the federated database.
   - For data sources that do not perform code page conversion, some of the wrappers perform the conversion.

   For example, if the federated database uses code page 819, territory US, the equivalent Oracle client locale is American_America.WE8ISO8859P1. The wrapper sets the NLS_LANG variable to the Oracle client locale value. When data is sent from the Oracle database to the wrapper, the Oracle database converts the data from code set American_America.WE8ISO8859P1 to code page 819. When data is sent from the Oracle database to the wrapper, the Oracle server or client converts the data from the code page of the Oracle database to code page 819. When data is sent from the wrapper to the Oracle database, the Oracle server or client converts the data from code page 819 to the code page of the Oracle database.

Tip: Defining the federated database to use the same code set, territory, and collating sequence as your data source can improve performance. If you define the federated database to use the same code set, territory, and collating sequence as your data source, then the code page conversion is not necessary. Using the same national language settings can improve performance when you transfer large amounts of character data.

To specify the code set and territory on the federated database, you use the USING CODESET and TERRITORY parameters on the CREATE DATABASE command.

Collating sequences in a federated system

The collating sequence that you specify for the federated database impacts how queries with character sorts or comparisons are processed.

When the federated server receives a query, the SQL compiler consults information in the global catalog and the data source wrapper to help federated server process the query. As part of the SQL compiler process, the *query optimizer* analyzes the query. The SQL compiler develops alternative strategies, called *access plans*, for processing the query. The access plans might call for the query to be:

- Processed by the data sources
- Processed by the federated server
- Processed partly by the data sources and partly by the federated server

If the query requires character sorting or comparisons, the SQL compiler uses collating sequence information to determine which access plan to use. Performing character sorts and comparisons at the data source usually improves performance.

By default, the federated database collating sequence is case-sensitive. However some of the federated data sources use collating sequences that are case-insensitive. If the collating sequences of the federated database and the data source are different, the query results might differ depending on where the character sorts and comparisons are performed. Generally, if the operation is a character sort the same data is returned but the order of the results will be different. If the operation is a character comparison, the results returned might be different.

Where the sorting or comparison is processed depends on several factors:

- If the federated database collating sequence is the same as the data source collating sequence, the character sort or comparison can take place at the data source. The query optimizer can decide which is the most efficient way to complete the query, a local operation or a remote operation. It is assumed that all types of character comparisons and sorts by the data source would yield the same results as if those actions were performed by the federated database.
- If the federated database collating sequence is different than the data source collating sequence, but the data source collating sequence is case-sensitive, the character sort or comparison will take place at the federated database. It is assumed that the data source will yield the same results on character data for

WHERE=, DISTINCT, and GROUP BY operations. But other operations, such as ORDER BY and WHERE with a greater than or less than predicate, will yield different results on character data.

- If the federated database collating sequence is different than the data source collating sequence, but the data source collating sequence is case-insensitive, the character sort or comparison will take place at the federated database. It is assumed that the data source will count uppercase and lowercase letters as equivalent and will include them both in a result, regardless of whether uppercase or lowercase was specified in the requested operation. WHERE=, WHERE with a greater than or less than predicate, ORDER BY, DISTINCT, and GROUP BY operations will not be pushed down to the data source.

For example, a case-insensitive data source assigns the same weights to the characters ″S″ and ″s″. A case-insensitive data source with an English code page considers the words `STEWART`, `SteWArT`, and `stewart` to be equal. However when a case-sensitive collating sequence is used, different weights are assigned to the characters. Depending on the sensitivity of the collating sequence, the result set of a character sort or comparison will be different.

When the collating sequences of the federated database and the data source differ, the federated server retrieves the data. The character sorts and comparisons are performed locally. The reason for performing these tasks locally is that the federated server users expect to see the query results ordered according to the collating sequence defined for the federated server. By ordering the data locally, federated server users are guaranteed that the result sets will be consistent. Retrieving data for local sorts and comparisons usually decreases performance.

If you need to see the character data ordered in the data source collating sequence, you can submit your query in a pass-through session.

To determine if a data source and the federated database have the same collating sequence, consider the following factors:

Code page
The code page scheme, such as ASCII and EBCDIC, that is used by the federated server and the data source impacts the results.

National language support (NLS)
The collating sequence is related to the language supported on a server. Compare the federated server NLS information for your operating system to the data source NLS information.

Data source characteristics
Some data sources are created using case-insensitive collating sequences, which can yield different results from the federated database in order-dependent operations.

Customization
Some data sources provide multiple options for collating sequences or allow the collating sequence to be customized.

There are several options that you can use to set the collating sequence:

- Set the collating sequence when you create the federated database
- Set the COLLATING_SEQUENCE option when you create the server definition for a data source. This option is available only for relational data sources.

Chapter 3. Overview of configuring access to data sources

Overview of configuring access to data sources

The basic steps to configure the federated server to access a data source are similar, regardless of the data source.

You must register a set of objects that include a wrapper, a server definition, and a nickname. For some data sources, you might need to register a set of user mappings.

Fast track to configuring your data sources

You can accomplish most of the steps required to configure access to a data source through the DB2 Control Center.

Use the DB2 command line or the DB2 Command Center for the steps that require a command line.

Before you configure access to a data source, make sure that the federated server has been set up properly.

The steps to configure the federated server to access a data source are similar, regardless of the data source. The basic steps and recommended interface are:

Table 7. The recommended interface and configuration steps

Configuration step	Recommended interface	Notes
1. Prepare the federated server for the data source.	Client Configuration Assistant	Required for only some data sources. This step might require you to install software, configure a file, or check a setting.
2. Set the required environment variables.	DB2 Control Center	Environment variables are required to access: • Informix • Microsoft SQL Server • Oracle • Sybase • Teradata Some data sources, such as ODBC, might require environment variables under specific circumstances.
2. Register the wrappers.	The Federated Objects wizard in the DB2 Control Center.	A wrapper is required for each data source that you want to access.
3. Register the server definitions.	The Federated Objects wizard in the DB2 Control Center.	Server definitions are associated with a wrapper and used when you register nicknames.

Table 7. The recommended interface and configuration steps (continued)

Configuration step	Recommended interface	Notes
4. Specify the user mappings: • Create the user mappings • Develop a plugin to use with an external repository, such as LDAP, for the user mappings	Use the Federated Objects wizard in the DB2 Control Center to create the user mappings. Specify the DB2_UM_PLUGIN option if you use an external repository for the user mappings.	User mappings are required for only some data sources. If you create the user mappings, when you attempt to retrieve the remote password associated with a user mapping from the SYSCAT.USEROPTIONS catalog view, the remote password value is displayed encrypted.
5. Test the connection to the data source server.	DB2 Command Center	Required for only some data sources.
6. Create the nicknames.	The Federated Objects wizard in the DB2 Control Center.	A nickname is required for each data source object that you want to access.

Adding data sources to a federated server using the DB2 Control Center

Use the DB2 Control Center to quickly configure the federated server to access your data sources.

You must provide the federated server with information about the data sources and objects that you want to access. The DB2 Control Center includes a wizard to guide you through the steps required to configure the federated server.

Procedure

To configure the federated server to access data sources by using the DB2 Control Center, use the Federated Objects wizard. To start the wizard, right-click the **Federated Database Objects** folder and click **Create Federated Objects**.

The steps that are required to configure the federated server are different for each data source.

You can configure multiple federated servers to access data sources by using the Action Output window.

Configuring multiple federated servers to access data sources

A federated system can consist of multiple federated servers. Instead of configuring each federated server separately, you can save time by using the DB2 Control Center to configure the federated servers.

Before you begin

- WebSphere Federation Server must be installed on a server that will act as the federated server
- A federated database must exist on the federated server

About this task

When you configure the first server, the **Action Output** window captures the DDL statements that are issued when you create the federated objects. You can reuse or modify these statements, and apply the statements to quickly configure additional federated servers.

The **Action Output** window remains active for the current session. If you close the **Action Output** window, the DDL statements for the current session continue to be stored in the **Action Output** window. However, if you close the DB2 Control Center all of the DLL statements from the current session are removed from the **Action Output** window.

Procedure

To configure multiple federated servers to access data sources:

1. Use the DB2 Control Center to configure the first federated server for the data sources that you want to access. This captures each DDL statement.
2. Display the **Action Output** page in the **Action Output** window. If you closed the **Action Output** window, right-click the **Federated Database Objects** folder and click **Show Actions** to open the **Action Output** window.
3. Delete any DDL statements that you do not want to use on the other federated servers. To delete a statement, right-click the statement and click **Remove**. For example, you might want to delete any statements that display Failed in the status column on the **Action Output** page.
4. Copy the statements that you want to use on the other federated servers to the **Command Editor** page.
   a. Select the statements that you want to copy. To select multiple statements, use the Ctrl key.
   b. Right-click on the selected statements and click **Copy to Command Editor**. The **Command Editor** page opens.
5. Change any DDL statements in the **Command Editor** page that you want to use on the other federated servers. For example, you might want to change any statement that specifies a local schema.

   You must change the user mappings to specify the proper passwords for the federated server. When the DDL for the CREATE USER MAPPING statements is captured in the **Action Output** window, the passwords are masked by asterisks. You must replace the asterisks with the proper passwords.
6. After you change the DDL statements that were generated in the **Action Output** window, run the DDL statements on the next federated server.

Setting the data source environment variables

The environment variables are set in the db2dj.ini file.

Before you begin

This task should be performed by the system administrator.

Restrictions

Restrictions for the db2dj.ini file

About this task

Setting the environment variables is required for the following data sources:

- Informix
- Microsoft SQL Server
- Oracle
- Sybase
- Teradata

Under specific circumstances, you might need to set environment variables for other data sources. For example with ODBC data sources, if the .odbc.ini configuration file is not in your $HOME directory, you must set the ODBCINI environment variable.

When you install WebSphere Federation Server, the installation process attempts to set the environment variables that are required by some of the data sources. The installation process might not be able to set the environment variables if, for example, you do not have the client software installed on the federated server before you install WebSphere Federation Server.

Procedure

To set the environment variables:

Follow the steps to add a data source to a federated server. You can check the environment variables and set them (if necessary). The steps to check the environment variables are different for each data source.

- If you use the DB2 Control Center to add data sources to the federated server, the requirements for the environment variables are automatically checked. You can set the environment variables when you create or alter a wrapper.
- If you use the DB2 command line to add data sources to the federated server, you must set the environment variables manually.

Restrictions for the db2dj.ini file

The db2dj.ini file contains the environment variables. There are restrictions on the entries in the db2dj.ini file.

The following restrictions apply to the `db2dj.ini` file:

- Entries must use the format *evname=value*

  evname is the name of the environment variable and *value* is its value.
- The environment variable name has a maximum length of 255 bytes.
- The environment variable value has a maximum length of 765 bytes.
- The maximum length of any line in the file is 1021 bytes. Data beyond that length is ignored.

Applying environment variables in a multi-partition instance configuration

If your federated server instance has a multi-partition configuration, you must apply the data source environment variables to all partitions.

About this task

The `db2dj.ini` file contains the data source environment variables. This file was added to the federated server when you installed WebSphere Federation Server.

The default path and file name for the `db2dj.ini` file can be overridden by the DB2_DJ_INI registry variable.

You can apply the DB2_DJ_INI registry variable to all of the partitions or to a subset of the partitions.

Attention: When you set the DB2_DJ_INI registry variable, you must set it to an absolute path. If the FEDERATED parameter is set to YES and the DB2_DJ_INI registry variable is set to a relative path, the database engine will not start.

Procedure

To apply the data source environment variables to all partitions:

1. You must add a copy of the same `db2dj.ini` file to all of the partitions in your multiple-partition instance configuration. The default name of the file is `db2dj.ini` file.
   - On UNIX federated servers, the default path for the `db2dj.ini` file is `INSTHOME/sqllib/cfg`, where `INSTHOME` is the home directory of the instance owner.
   - On Windows federated servers, the default path for the file is `x:\SQLLIB\cfg`, where `x:\SQLLIB` is the drive and directory specified in the DB2PATH registry variable or environment variable.
2. To apply the DB2_DJ_INI registry variable to the appropriate partitions on your federated server, use the db2set command. The db2set command displays, sets, or removes DB2 profile variables.
3. The syntax of the db2set command that you use depends on your database system structure.

Partitions	Command
To apply the DB2_DJ_INI registry variable to all of the database partitions within this instance, issue this command:	`db2set -g DB2_DJ_INI=$HOME/sqllib/cfg/my_db2dj.ini`
To apply the DB2_DJ_INI registry variable to only the current partition, issue this command:	`db2set DB2_DJ_INI=$HOME/sqllib/cfg/my_db2dj.ini`
To apply the DB2_DJ_INI registry variable to a specific partition, issue this command:	`db2set -i` *`INSTANCEX3`* `DB2_DJ_INI=$HOME/sqllib/cfg/`*`partition3.ini`* *INSTANCEX* The name of the instance. *3* The partition number as listed in the `db2nodes.cfg` file. *partition3.ini* The modified and renamed version of the `db2dj.ini` file.

Registering wrappers for a data source

You must register a wrapper to access a data source. Federated servers use wrappers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.

About this task

You register one wrapper for each type of data source that you want to access. For example, to access three DB2 for z/OS® database tables, one DB2 for iSeries™ table, and two BLAST search types, you need to register two wrappers. You need to register one wrapper for the DB2 databases and one wrapper for the BLAST search types.

Procedure

To register a wrapper:

You can register a wrapper by using the DB2 Control Center or from the DB2 command line:

- To register a wrapper by using the DB2 Control Center, use the Federated Objects wizard or right-click the **Federated Objects** folder and click **Create Wrapper**.
- To register a wrapper from the DB2 command line, use the CREATE WRAPPER statement.

After the wrappers are registered in the federated database, you can use these wrappers to access other objects from those data sources. For example, if you register a wrapper to access a DB2 for iSeries table, you can use the DB2 wrapper to access data sources from all of the DB2 family data source objects, including DB2 for Linux, UNIX, and Windows, DB2 for z/OS and OS/390®, DB2 for iSeries, and DB2 Server for VM and VSE.

Registering server definitions for a data source

Every data source object that you create a nickname for must be associated with a specific server definition.

About this task

The purpose of a server definition varies from data source to data source.

A server definition for relational data sources usually represents a remote database, a database partition, or a node. For nonrelational data sources, some server definitions map to a search type and daemon, to a web site, or to a web server. For other nonrelational data sources, a server definition is created only because it is required by federation.

For some data sources, you must specify a node when you register a server definition. The concept of a node varies from data source to data source. For relational data sources, a node reflects a server instance of the data source.

Procedure

To register server definitions for a data source:

You can register a server definition by using the DB2 Control Center or from the DB2 command line.

- To register a server definition by using the DB2 Control Center, use the Federated Objects wizard or right-click the **Server Definitions** folder and click **Create**.
- To register a server definition from the DB2 command line, use the CREATE SERVER statement.

The parameters and options that you specify when you register a server definition vary from data source to data source.

Registering user mappings for a data source

For some data sources, you must define an association between the federated server authorization ID and the data source user ID and password.

About this task

You create a user mapping for each user ID that uses the federated system to send distributed requests. You can either use an external repository to store the user mappings or you can register the user mapping in the catalog of the federated database.

Procedure

To registering user mappings for a data source:

You can create a user mapping from the DB2 Control Center or the DB2 command line:

Method	Action
To register user mappings by using an external repository	Create a user mapping plugin and specify the DB2_UM_PLUGIN option when you register the wrapper or server definition.
To register user mappings by using the DB2 Control Center	Use the Federated Objects wizard or right-click the **User Mappings** folder and click **Create**.
To register user mappings from the DB2 command line	Use the CREATE USER MAPPING statement.

Registering nicknames for a data source

You must register a nickname for each data source object that you want to access.

About this task

The steps and requirements for registering a nickname are different for each data source.

Recommendation: Use the DB2 Control Center to register nicknames. For most data sources, you can use the Discover tool in the DB2 Control Center. The Discover tool helps you to quickly identify data source objects that you might want to register nicknames for.

Procedure

To register a nickname for a data source:

You can register a nickname by using the DB2 Control Center or from the DB2 command line:

- To register a nickname by using the DB2 Control Center, use the Federated Objects wizard or right-click the **Nicknames** folder and click **Create**. Use the Discover tool to identify the objects that you want to create nicknames for.

- To register a nickname from the DB2 command line, use the CREATE NICKNAME statement.

You can define more than one nickname for the same data source object.

Specifying nickname columns for a nonrelational data source

For some nonrelational data sources, you must define a list of columns when you register a nickname. Each column that you specify is mapped to a particular field, column, or element in the data source object.

About this task

The wrappers for some of the nonrelational data sources require a set of fixed input and output columns. The fixed columns are automatically defined when you register the nickname and are added to the systems catalog on the federated database.

Procedure

To define a list of columns when you register a nickname, specify the column name and data type. You might also need to specify an option on the nickname column.

Optional configuration steps

There are several optional steps that you might need to take when you configure the federated server to access data sources.

Index specifications

Define an index specification for objects that do not have an index. For example, you would create an index specification when a table acquires a new index or if the data source object (such as a view) typically does not have an index.

Data type mappings

You can specify alternative data type mappings for only relational data sources.

Specify alternative data type mappings, instead of using the default data type mappings. You can specify a mapping that is used only for a specific data source object, such as a specific table within a database.

Function mappings

You can specify function mappings for only relational data sources.

Define alternative function mappings, instead of using the default function mappings. This is especially useful when you want to force the federated server to use a user-defined function at the data source.

Chapter 4. BioRS data sources

Configuring access to BioRS data sources

You can integrate the data that is in the BioRS databanks with information from other sources by using a federated system.

Procedure

To configure a federated server to access the BioRS data sources, you must provide the federated server with information about the data sources and objects that you want to access. After you configure the federated server, you can create queries and use the custom functions to access the BioRS data sources.

BioRS wrapper

BioRS is a query and retrieval system that you can use to retrieve information from multiple data sources. The BioRS wrapper uses the same APIs as the BioRS Web-based interface to run queries.

BioRS is a query and retrieval system that is developed by Biomax Informatics. You can use BioRS to retrieve information from multiple data sources, including flat files and relational databases. You usually download public data, such as SwissProt and GenBank, as flat files into your BioRS system. BioRS can integrate public data sources and proprietary data sources (for example, private databases that are maintained by your organization) into a common environment.

After a data source is integrated into the BioRS system, it is referred to as a *databank*. The elements that are contained in each databank entry are collectively referred to as a *schema*. Elements of a databank that are indexed can be used in the BIORS.CONTAINS, BIORS.CONTAINS_GE, and BIORS.CONTAINS_LE functions. The BioRS functions are specified in the WHERE clause of the SELECT statement. Elements that are not indexed can be referenced in the SELECT list and in other predicates in the WHERE clause. Elements that are not indexed are processed by the federated server.

You can establish relationships between entries in databanks, so that you can link databanks together in the BioRS system.

BioRS databanks can have a parent-child relationship (databanks can be nested). In such a relationship, the child databank contains a Reference data type element called PARENT. The PARENT element refers to the _ID_ element of the parent databank. Other than the presence of this predefined PARENT element, nested databanks contain the same data as unnested databanks.

BioRS provides a Web-based interface that enables users to run queries on the data in BioRS databanks. The BioRS wrapper uses the same application programming interfaces (APIs) as the BioRS Web-based interface to run queries. The BioRS wrapper works with BioRS Version 5.0.14 and Version 5.2.

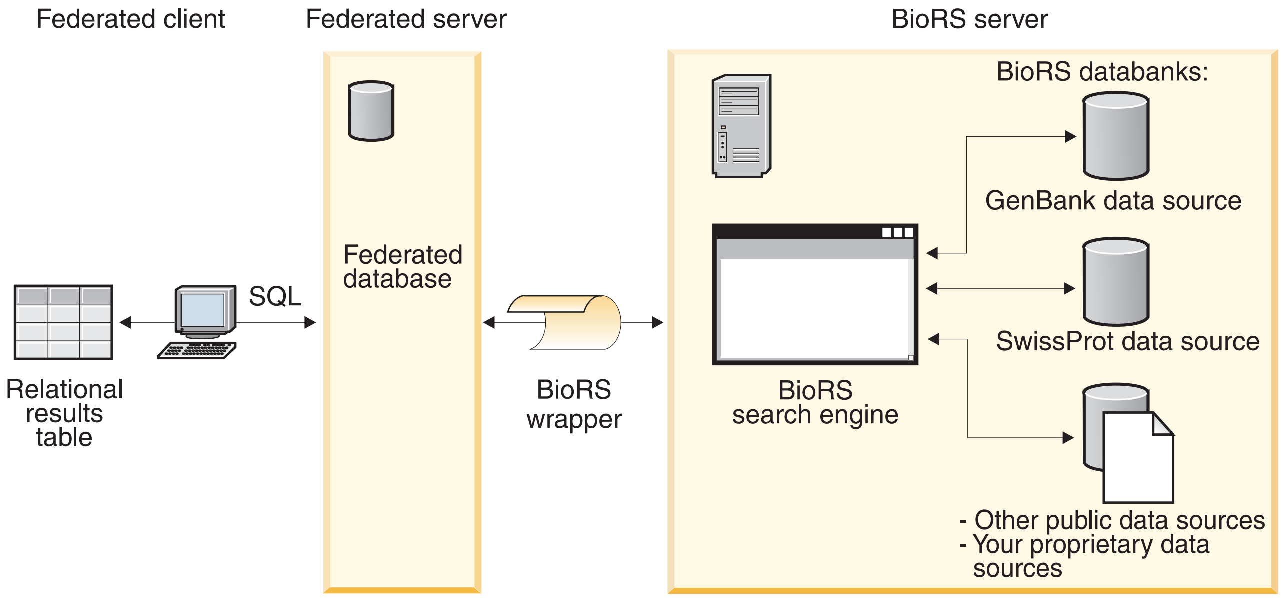


Figure 4. How the BioRS wrapper works

From the client, users or applications submit a query using SQL statements. Then, the query is sent to your federated system where the BioRS wrapper is installed. Depending on how the query is constructed, both the federated server and your BioRS server might be used to process the query. The BioRS server can be on a different computer from the federated system. Authentication information must be provided by the federated system to the BioRS server for each query. This information can be either a user ID and password combination, or an unauthenticated indication (usually a guest account).

For detailed information about the BioRS product, see the Biomax Web site at: http://www.biomax.com.

Adding BioRS data sources to a federated server

To configure a federated server to access BioRS data sources, you must provide the federated server with information about the data sources and objects that you want to access.

Before you begin

- IBM® WebSphere Federation Server must be installed on a server that will act as the federated server
- A federated database must exist on the federated server

About this task

You can configure a federated server to access data that is stored in BioRS data sources by using the Control Center or by issuing SQL statements on the command line. The Control Center includes a wizard to guide you through the steps that are necessary to configure the required federated objects.

Procedure

To add BioRS data sources to a federated server:

1. Register the custom functions for the BioRS wrapper.

2. Register the BioRS wrapper.
3. Register the BioRS server definitions.
4. Optional: Create the BioRS user mappings.
5. Register nicknames for BioRS databanks.

Registering the custom functions for the BioRS wrapper

You must register the BioRS custom functions before you register the BioRS wrapper. The BioRS custom functions are used with the BioRS wrapper to push down predicates to the BioRS query engine.

About this task

You must register all of the custom functions on each federated database instance where the BioRS wrapper is installed.

All of the custom functions for the BioRS wrapper must be registered with the schema name biors.

Procedure

To register the BioRS custom functions:

For each of the BioRS custom functions, issue the CREATE FUNCTION statement. You must specify two data types in each custom function. The first data type that you specify is the indexed column. The second data type that you specify is the search term. You must include the AS TEMPLATE, DETERMINISTIC, and NO EXTERNAL ACTION keywords in the CREATE FUNCTION statement.

The following example shows the syntax for the BIORS.CONTAINS function:

```
CREATE FUNCTION biors.contains (column_data_type, search_term_data_type)
    RETURNS INTEGER AS TEMPLATE
    DETERMINISTIC NO EXTERNAL ACTION;
```

Tip: Use the sample file create_function_mappings.ddl to register the BioRS custom functions. The sample file is located in the sqllib/samples/lifesci/biors directory on the federated server. The sample file contains the CREATE FUNCTION statements for each of the possible data type combinations. To register the custom functions, edit the create_function_mappings.ddl file to specify the data types for the index columns and search terms for each custom function. Then you must run the create_function_mappings.ddl file on each federated database instance where the BioRS wrapper is installed.

Custom functions for the BioRS wrapper

Descriptions and examples of the custom functions that are used with the BioRS wrapper.

The federated environment uses two query engines. For the BioRS wrapper, these query engines are the federated database query engine and the BioRS query engine. You can specify that predicates get pushed down to the BioRS engine by using the BioRS custom functions.

The custom functions for the BioRS wrapper are:

- BIORS.CONTAINS

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

You use the CREATE FUNCTION statement to register the BioRS custom functions.

The following table lists the four BioRS custom functions with examples of the valid data types that you can specify when you register the functions.

Table 8. Custom functions for the BioRS wrapper

Function	Description
BIORS.CONTAINS (VARCHAR(), VARCHAR()) BIORS.CONTAINS (VARCHAR(), CHAR()) BIORS.CONTAINS (VARCHAR(), DATE) BIORS.CONTAINS (VARCHAR(), TIMESTAMP)	Searches an indexed column for values that are equal (according to the BioRS query semantics) to the value that you specify. The first argument must be a reference to the indexed column and the second argument is the search value that you specify.
BIORS.CONTAINS_LE (VARCHAR(), VARCHAR()) BIORS.CONTAINS_LE (VARCHAR(), SMALLINT) BIORS.CONTAINS_LE (VARCHAR(), BIGINT) BIORS.CONTAINS_LE (VARCHAR(), DECIMAL) BIORS.CONTAINS_LE (VARCHAR(), DOUBLE) BIORS.CONTAINS_LE (VARCHAR(), REAL)	Searches an indexed column for values that are less than or equal (according to the BioRS query semantics) to the value that you specify. The first argument must be a reference to the indexed column and the second argument is the search value that you specify.
BIORS.CONTAINS_GE (CHAR(), CHAR()) BIORS.CONTAINS_GE (CHAR(), DATE) BIORS.CONTAINS_GE (CHAR(), TIMESTAMP) BIORS.CONTAINS_GE (CHAR(), INTEGER) BIORS.CONTAINS_GE (CHAR(), SMALLINT) BIORS.CONTAINS_GE (CLOB(), DATE)	Searches an indexed column for values that are greater than or equal (according to the BioRS query semantics) to the value that you specify. The first argument must be a reference to the indexed column and the second argument is the search value that you specify.
BIORS.SEARCH_TERM (VARCHAR(), VARCHAR()) BIORS.SEARCH_TERM (VARCHAR(), CHAR()) BIORS.SEARCH_TERM (CHAR(), VARCHAR()) BIORS.SEARCH_TERM (CHAR(), CHAR())	Passes a BioRS query expression to the BioRS search engine.

Registering the BioRS wrapper

You must register a wrapper to access BioRS data sources. Federated servers use wrappers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.

About this task

You can register a wrapper by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps that are necessary to register the wrapper.

If you use a proxy server and a keystore to access the BioRS files, you can specify the keystore information as options when you register the wrapper or server definition. If you specify the keystore information when you register the wrapper, the settings are used when you query any BioRS file. The wrapper settings are not used if you specify different settings when you register the server definitions.

Procedure

To register the BioRS wrapper:

Choose the method that you want to use to register the BioRS wrapper:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.

From the command line	Issue the CREATE WRAPPER statement.
	`CREATE WRAPPER` *`wrapper_name`*
	`LIBRARY` *`library_name`*`;`
	If you use a proxy server to access BioRS documents, the statement that you issue is:
	`CREATE WRAPPER` *`wrapper_name`*
	`LIBRARY` *`library_name`*
	`OPTIONS (PROXY_TYPE '`*`type`*`',`
	`PROXY_SERVER_NAME '`*`server_name`*`',`
	`PROXY_SERVER_PORT '`*`port_number`*`');`

You must specify the LIBRARY parameter in the CREATE WRAPPER statement. The name of the wrapper library file that you specify depends on the operating system of the federated server. See the list of BioRS wrapper library files for the correct library name to specify in the CREATE WRAPPER statement.

If you use a proxy server or a keystore to access BioRS files, you must specify several wrapper options when you register the BioRS wrapper.

BioRS wrapper library files

The BioRS wrapper library files are added to the federated server when you install WebSphere Federation Server.

When you install WebSphere Federation Server, three library files are added to the default directory path. For example, if the federated server is running on AIX®, the wrapper library files that are added to the directory path are libdb2lsbiors.a, libdb2lsbiorsF.a, and libdb2lsbiorsU.a. The default wrapper library file is libdb2lsbiors.a. The other wrapper library files are used with specific wrapper options.

You must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

The default directory paths and default wrapper library file names are listed in the following table.

Table 9. BioRS wrapper library locations and file names

Operating system	Directory path	Wrapper library file name
AIX	/usr/opt/`<install_path>`/lib32/ /usr/opt/`<install_path>`/lib64/	libdb2lsbiors.a
Linux	/opt/IBM/db2/`<install_path>`/lib32 /opt/IBM/db2/`<install_path>`/lib64	libdb2lsbiors.so
Solaris	/opt/IBM/db2/`<install_path>`/lib32 /opt/IBM/db2/`<install_path>`/lib64	libdb2lsbiors.so
Windows	%DB2PATH%\bin	db2lsbiors.dll

<install_path> is the directory path where IBM WebSphere Federation Server is installed on UNIX or Linux.

%DB2PATH% is the environment variable that is used to specify the directory path where IBM WebSphere Federation Server is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

CREATE WRAPPER statement - examples for the BioRS wrapper

Use the CREATE WRAPPER statement to register the BioRS wrapper. The examples show the parameters that are required to access BioRS documents with and without a proxy server.

Registering a wrapper

If you are not using a proxy server to access XML documents, the statement that you issue to register the wrapper is:

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

biors_wrapper
A name that you assign to the BioRS wrapper. Duplicate wrapper names are not allowed.

LIBRARY *'libdb2lsbiors.a'*
The name of the wrapper library file for federated servers that use AIX operating systems.

Registering a wrapper for an HTTP proxy server

To register a wrapper and specify an HTTP proxy server, use the following statement:

```
CREATE WRAPPER biors_proxy LIBRARY 'libdb2lsbiors.a'
    OPTIONS (PROXY_TYPE 'HTTP',
        PROXY_SERVER_NAME 'proxy.mysite.com',
        PROXY_SERVER_PORT '81');
```

PROXY_TYPE *'HTTP'*
Specifies the proxy type that is used to access the Internet when the federated server is behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy.mysite.com'*
Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'81'*
Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Registering a wrapper for a SOCKS proxy server

To register a wrapper and specify a SOCKS proxy server without authentication information, use the following statement:

```
CREATE WRAPPER biors_wrapper LIBRARY 'libdb2lsbiors.so'
    OPTIONS (PROXY_TYPE 'SOCKS',
        PROXY_SERVER_NAME 'proxy_socks',
        PROXY_SERVER_PORT '1081');
```

LIBRARY *'libdb2lsbiors.so'*

The name of the wrapper library file for federated servers that use the Linux and Solaris operating systems.

PROXY_TYPE *'SOCKS'*

Specifies the proxy type that is used to access the Internet when the federated server is behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy_socks'*

Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'1081'*

Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Registering the server definition for a BioRS data source

You must register each BioRS server that you want to access in the federated database.

About this task

You can register a server definition by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps that are necessary to register the server definition.

If you use a proxy server and a keystore to access BioRS files, you can specify the proxy server information as options when you register the wrapper or server definition. If you specify the proxy server information when you register the server definition, those settings override the proxy server settings that you specify when you register the wrapper.

Procedure

To register a server definition for a BioRS data source:

Choose the method that you want to use to register the server definition:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.

Method	Procedure

From the command line	Issue the CREATE SERVER statement. For example:
	`CREATE SERVER server_definition_name VERSION version_number WRAPPER wrapper_name OPTIONS (NODE 'node_name');`
	If you use a proxy server or a keystore to access XML files, you must specify several options when you register either the BioRS wrapper or the server definition. To specify the proxy server information when you register the BioRS wrapper, the statement that you issue is:
	`CREATE SERVER server_definition_name VERSION version_number WRAPPER wrapper_name OPTIONS (PROXY_TYPE 'type', PROXY_SERVER_NAME 'server_name', PROXY_SERVER_PORT 'port_number');`

When you register the server definition, you specify server options in the CREATE SERVER statement. There are required server options and optional server options. The NODE server option is required when the BioRS server is running on a different system than IBM WebSphere Federation Server.

After the server definition is registered, use the ALTER SERVER statement to add or drop server options.

CREATE SERVER statement - Examples for the BioRS wrapper

Use the CREATE SERVER statement to register server definitions for the BioRS wrapper. The examples show the required parameters, the optional parameters, and the additional server options that you can specify.

Registering a server definition with the required parameters

The following example shows you how to register a server definition for a BioRS wrapper by issuing the CREATE SERVER statement:

```
CREATE SERVER biors_server VERSION '5.2' WRAPPER biors_wrapper
     OPTIONS(NODE 'biors.myco.com');
```

biors_server
: A name that you assign to the BioRS server. Duplicate server definition names are not allowed.

VERSION *'5.2'*
: The version of the BioRS server that you want to access. The supported BioRS versions are 5.0.14 and 5.2. If you are accessing a BioRS server that is version 5.2, you must specify *'5.2'* as the value for the VERSION parameter. You do not need to specify this option if you are using version 5.0.14. The default value of *'1.0'*, which equates to version 5.0.14, is used for this parameter if you do not specify the value.

WRAPPER *biors_wrapper*
: The wrapper name that you specified in the CREATE WRAPPER statement.

NODE *'biors_server2.com'*
Specifies the host name of the system on which the BioRS query tool is available. The default value is *localhost*. This value is case sensitive.

Although the name of the node is specified as an option in the CREATE SERVER statement, it is required for BioRS data sources.

Registering a server definition using optional parameters and server options

The following example shows additional parameters and server options that you can specify when you register a server definition for a BioRS wrapper:

```
CREATE SERVER biors_server TYPE BioRS VERSION '5.2'
     WRAPPER biors_wrapper
     OPTIONS (NODE 'biors_server2.com', PORT '2525', TIMEOUT 30 ,
     CASE_SENSITIVE 'N');
```

TYPE *BioRS*
Specifies the type of data source server to which you are configuring access. For the BioRS wrapper, the server type must be `BioRS`. This parameter is optional.

PORT *'5555'*
Specifies the number of the port to be used to connect to the BioRS server. The default value is '5014'.

TIMEOUT *30*
Specifies the time, in minutes, that the BioRS wrapper waits for a response from the BioRS server. The default value is 10. This parameter is optional.

CASE_SENSITIVE *'N'*
Specifies whether the BioRS server treats names in a case sensitive manner. Valid values are 'Y' or 'N'. The default value is 'Y'.

In the BioRS product, a configuration parameter controls the case sensitivity of the data that is stored on the BioRS server machine. The CASE_SENSITIVE option is the federated server counterpart to that BioRS system configuration parameter. You must synchronize the BioRS server case sensitivity configuration settings in your BioRS system and in the federated server. If you do not keep the case sensitivity configuration settings synchronized between the BioRS server and the federated server, errors will occur when you attempt to access BioRS data through the federated server.

Important: You cannot change or delete the CASE_SENSITIVE option after you create a new BioRS definition. If you need to change the CASE_SENSITIVE option, you must drop and then create the server definition again. If you drop the BioRS server definition, you must also create all of the BioRS nicknames that referenced that server definition. The federated server automatically drops all nicknames that correspond to a dropped server.

Registering a server definition that includes a proxy server

```
CREATE SERVER biors_proxy_serv VERSION 5.2 WRAPPER biors_proxy
     OPTIONS (NODE 'biors.mysite.com',
     PORT '5555',
     PROXY_TYPE 'HTTP'
```

PROXY_TYPE *'HTTP'*
Specifies the proxy type that is used to access the Internet when behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy.mysite.com'*
Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'81'*
Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Registering a server definition that includes a proxy server with authentication information

To register a server definition and specify a SOCKS proxy server with authentication information, use the following statement:

```
CREATE SERVER biors_proxy_serv VERSION 5.2 WRAPPER biors_proxy
     OPTIONS (NODE 'biors.mysite.com',
         PORT '5555',
         PROXY_TYPE 'SOCKS'
         PROXY_SERVER_NAME 'proxy_socks',
         PROXY_SERVER_PORT '1081',
         PROXY_AUTHID 'argle'
         PROXY_PASSWORD 'bargle')
```

PROXY_AUTHID *'argle'*
Specifies the user ID on the proxy server. This server option is required when the value for the PROXY_TYPE server option is 'SOCKS'.

PROXY_PASSWORD *'bargle'*
Specifies the password on the proxy server that is associated with the user name *'argle'*. This server option is required when the value for the PROXY_TYPE server option is 'SOCKS'.

Creating the user mappings for a BioRS data source

When you attempt to access a BioRS server, the federated server establishes a connection to the BioRS server. Depending on the account access methods that are used in your BioRS system, you might not need to create user mappings.

A user mapping is an association between each federated server user ID and password and the corresponding data source user ID and password.

There are two methods for specifying user mappings with federated systems. You can use an external repository, such as LDAP, to store the user mappings or you can create the user mappings in the federated database catalog.

Before you begin

The account access methods that are used in your BioRS system will determine if you need to create user mappings:

- If your BioRS server is configured for guest access for all user accounts, you do not need to create user mappings. The federated server uses a guest account to access the BioRS server.
- If you have an external repository, such as LDAP, to store the user mappings, you do not need to create user mappings. You must specify the

DB2_UM_PLUGIN option on the BioRS wrapper. You can specify this option when you register or alter the wrapper. The schema in the external repository must include guest access.

- If your BioRS server is configured to authenticate user accounts with IDs and passwords, you must create user mappings in the federated database for any user accounts that will use the BioRS wrapper to access BioRS data sources.
- If your BioRS server is configured to use a mixture of guest and authenticated user accounts, you must create user mappings for the authenticated user accounts that will use the BioRS wrapper to access BioRS data sources.

About this task

User mappings provide a way to authenticate the access of users or applications that query a BioRS data source with the BioRS wrapper. If a user mapping is not defined for a user or application, the federated server uses a guest account. If a databank that is being queried requires authentication, an error message might be returned.

To ensure that the correct user ID and password get passed to the BioRS server, create user mappings in your federated database for users who are authorized to search BioRS data sources. When you create a user mapping, the remote password is stored in an encrypted format in a federated database system catalog table.

Procedure

To map a local user ID to the BioRS server user ID and password:

Choose the method that you want to use to create the user mappings:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.

Using the command line	Issue the CREATE USER MAPPING statement. For example:
	`CREATE USER MAPPING FOR local_userID`
	`SERVER server_definition_name`
	`OPTIONS (REMOTE_AUTHID 'remote_userID',`
	`REMOTE_PASSWORD 'remote_password')`
	`PROXY_AUTHID 'proxy_server_userID',`
	`PROXY_PASSWORD 'proxy_server_password';`

If you specify authentication information for the proxy server when you register a server definition and a user mapping, the values that you specify in the CREATE USER MAPPING statement take precedence over the values that you specify in the CREATE SERVER statement.

For example, you have ten people in your organization and you specify authentication information when you register the server definition. You create user mappings for three of the ten people. When the three people access the federated system, the authentication information that you specified when you created the user mappings is used. For the remaining seven people, the authentication information that you specified when you registered the server definition is used.

CREATE USER MAPPING statement - Examples for the BioRS wrapper

Use the CREATE USER MAPPING statement to map a federated server user ID to a BioRS server user ID and password.

You can create a user mapping by specifying a guest user account, an authenticated user account, the USER special register, or a proxy server.

Creating a user mapping for a guest user account

The GUEST user option specifies if the BioRS wrapper should use a guest account to access the BioRS server.

The following example shows how to specify that a guest user account is used to access the BioRS server:

```
CREATE USER MAPPING FOR charlie SERVER biors_server
      OPTIONS (GUEST 'Y');
```

charlie Specifies the local user ID that you are mapping to a user ID that is defined at the BioRS server.

SERVER *biors_server*
Specifies the server definition name that you registered in the CREATE SERVER statement for the BioRS server.

GUEST *'Y'*

Specifies that the BioRS wrapper uses a guest user account to authenticate this user.

Creating a user mapping for an authenticated user account

The following example shows how to map a federated server user ID to a BioRS server user ID and password:

```
CREATE USER MAPPING FOR charlie SERVER biors_server
      OPTIONS (REMOTE_AUTHID 'charlene',
      REMOTE_PASSWORD 'all4one');
```

charlie Specifies the local user ID that you are mapping to a user ID that is defined at the BioRS server.

SERVER *biors_server*
Specifies the server definition name that you registered in the CREATE SERVER statement for the BioRS server.

REMOTE_AUTHID *'charlene'*
Specifies the user ID at the BioRS server to which you are mapping *charlie*. This remote ID must be in a format that is expected by the BioRS server. This option is required if you do not use a guest account to access the BioRS server.

REMOTE_PASSWORD *'all4one'*
Specifies the password that is associated with *'charlene'*. This option is required if you do not use a guest account to access the BioRS server.

Creating a user mappings by using a special register

You can use the federated database special register USER to map the authorization ID of the person who is issuing the CREATE USER MAPPING statement to the data source authorization ID that is specified in the REMOTE_AUTHID user option.

The following example shows a CREATE USER MAPPING statement that includes the special register USER:

```
CREATE USER MAPPING FOR USER SERVER biors_server
       OPTIONS (REMOTE_AUTHID 'charlene', REMOTE_PASSWORD 'all4one');
```

Creating a user mapping for a proxy server

The following example shows how to map a federated server user ID to a BioRS server user ID and password:

To register a server definition and specify a SOCKS proxy server with authentication information, use the following statement:

```
CREATE USER MAPPING FOR charlie SERVER biors_proxy
      OPTIONS (REMOTE_AUTHID 'charlene',
      REMOTE_PASSWORD 'all4one'
      PROXY_AUTHID 'chuck'
      PROXY_PASSWORD 'them2us');
```

PROXY_AUTHID *'chuck'*
Specifies the user ID on the proxy server. This user mapping option is required when the proxy server requires authentication.

PROXY_PASSWORD *'them2us'*
Specifies the password on the proxy server that is associated with the user name *'chuck'*. This user mapping option is required when the proxy server requires authentication.

Registering nicknames for BioRS data sources

For each BioRS server definition that you register, you must register a nickname for each databank that you want to access. Use these nicknames, instead of the names of the databanks, when you query the BioRS servers.

Before you begin

- If a BioRS databank name does not conform to the syntax required by the CREATE NICKNAME statement, you must use the REMOTE_OBJECT nickname option when you register the nickname.
- If a BioRS element name does not conform to the syntax required by the CREATE NICKNAME statement, you must use the ELEMENT_NAME column option when you register the nickname.

Restrictions

Do not use the BioRS AllText element as the first column for a nickname. You can use the BioRS AllText element in any other column position (for example, as the second column or as the third column).

About this task

After a data source has been integrated into the BioRS system, it is referred to as a *databank* in BioRS. Databanks in BioRS equate to nicknames in a federated system.

The names that you give the nicknames can be up to 128 characters in length.

You can register a nickname by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps that are necessary to register the nickname.

Procedure

To register a nickname for a BioRS databank:

Choose the method that you want to use to register the nickname:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.

Using the command line	Issue the CREATE NICKNAME statement. For example:
	`CREATE NICKNAME nickname`
	`(`
	`column_name data_type`
	`   OPTIONS (nickname_column_options),`
	`column_name data_type`
	`   OPTIONS (nickname_column_options),`
	`column_name data_type`
	`   OPTIONS (nickname_column_options)`
	`)`
	`FOR SERVER server_definition_name`
	`OPTIONS (nickname_options);`

When you create a BioRS nickname, you define a list of nickname columns. The specified nickname columns must correspond to elements of a specific BioRS databank format. BioRS defines five possible data types for elements: Text, Number, Date, Author, and Reference. The BioRS data types can be mapped only to the CHAR, CLOB, or VARCHAR data types that are used by the federated database.

Repeat this step for each BioRS databank that you want to create a nickname for.

CREATE NICKNAME statement - examples for the BioRS wrapper

Use the CREATE NICKNAME statement to register a nickname for a BioRS databank that you want to access.

Creating simple, but limited, nicknames for databanks

The simplest way to register a nickname for a BioRS databank is to give the nickname the same name as the BioRS databank.

For example:

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

The name of the nickname is *SwissProt*, which is the same as the name of the corresponding BioRS databank.

Using this simple CREATE NICKNAME syntax limits you in two ways:

1. You are limited to one family of nicknames for each federated database schema. For example, you have two databanks that have a parent-child relationship. The databanks are SWISSPROT and SPFEAT. These databanks form a family. If you use the default syntax for the CREATE NICKNAME statement, you will have the (SWISSPROT nickname for the SWISSPROT databank and the SPFEAT nickname for the SPFEAT databank. To have more than one nickname for SWISSPROT in the schema, you must use the REMOTE_OBJECT nickname option when you register the nickname.
2. You are limited to databanks whose names can be used as nickname names. The databank name must conform to the syntax that is supported by the federated server. For example, if the databank name includes a period or space, you must use the REMOTE_OBJECT nickname option when you register the nickname.

Creating multiple nicknames for the same databank

The REMOTE_OBJECT nickname option specifies the name of the BioRS databank that is associated with the nickname. The name that you specify in the REMOTE_OBJECT nickname option determines the schema and the BioRS databank for the nickname. The REMOTE_OBJECT nickname option also specifies the relationship of the nickname to other nicknames.

The following example shows the same set of nickname characteristics as the previous example, but changes the nickname name. The example uses the REMOTE_OBJECT nickname option to specify the BioRS databank for which the nickname is being defined:

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

Creating nicknames for databanks that do not conform to federated syntax

The following example shows how to create a nickname for a remote BioRS databank that does not conform to the syntax required by the federated server:

```
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');
```

SwissFT
A unique nickname that is used to identify the BioRS databank.

ID *VARCHAR(32)* **OPTIONS (ELEMENT_NAME** *'_ID_'***)**

The name and data type for a table column. The ELEMENT_NAME column option is specified for the ID column.

The ELEMENT_NAME column option specifies the BioRS element name. The case sensitivity of this name depends on the case sensitivity of the BioRS server and on the value of the CASE_SENSITIVE server option. You need to specify the BioRS element name only if it is different from the column name. Column option values must be enclosed in single quotation marks.

In general, use the ELEMENT_NAME column option under the following circumstances:

- When a BioRS element name contains characters, such as periods and spaces, that do not conform to valid federated syntax. For example, if your databank has an element named Pub.Date, you cannot use the element name as the column name. Characters such as periods and spaces are not supported. You must map the element name to a valid column name.
- When the syntax of a BioRS element name does not conform to standards that you or your organization have established for your federated system. For example, if your organization has established that the conventions for schemas, nicknames, and columns must include a prefix, a BioRS element name might not be able to be used as a column name.
- When the BioRS element name might not be obvious to federated users.

ALLTEXT *VARCHAR(128)*

The name and data type for a table column.

ENTRYDATE *VARCHAR(64)*

The name and data type for a table column.

FtLength *VARCHAR(16)*

The name and data type for a table column.

SERVER *biors1*

The name that you assigned to the BioRS server in the CREATE SERVER statement.

OPTIONS (REMOTE_OBJECT *'SwissProt.Features'***)**

Specifies the name of the BioRS databank that is associated with the nickname. This name determines the schema and the BioRS databank for the nickname. This name also specifies the relationship of the nickname to other nicknames.

The case sensitivity of this name depends on the case sensitivity of the BioRS server and on the value of the CASE_SENSITIVE server option.

Important: You cannot change or delete this name with the ALTER NICKNAME statement. If the name of the BioRS databank changes, you must delete and then create the nickname again.

You must specify the REMOTE_OBJECT nickname option when the name of a BioRS databank does not conform to valid federated syntax. In this example, the databank name "SwissProt.Features" does not conform for several reasons. The databank name contains a character, a period, that is not valid federated syntax and contains a mixture of uppercase and lowercase letters.

In general, use the REMOTE_OBJECT nickname option under the following circumstances:

- When a BioRS databank name contains characters, such as periods and spaces, that do not conform to valid federated syntax. You must map the databank name to a valid federated name.
- When the case sensitivity of a BioRS databank name does not conform to standards that you or your organization have established for your federated system. For example, if your organization has established that the conventions for schemas, nicknames, and columns must include a prefix, a BioRS databank name might not be able to be used a name.
- When the BioRS databank name might not be obvious to federated users.

Creating nicknames for a databank linked to another BioRS databank

The following example shows how to create a nickname for a table that uses a BioRS databank that is linked to another BioRS databank:

```
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');
```

The name of this nickname is SwissFT2. The table columns are ID, ALLTEXT, FtKey, FtLength, FtDescription, and Parent. The ELEMENT_NAME column option is specified for the ID column. The REMOTE_OBJECT option is used to specify the name of the BioRS databank to which the nickname corresponds.

Additionally, the Parent column uses the REFERENCED_OBJECT option. You must specify this option for columns that correspond to BioRS Reference data type elements. The REFERENCED_OBJECT option specifies the name of the BioRS databank to which the column refers. In this case, the Parent element refers to the BioRS SwissProt databank.

Custom functions and BioRS queries

The BioRS custom functions are used with the BioRS wrapper to push down predicates to the BioRS query engine.

The federated environment uses two query engines. For the BioRS wrapper, these query engines are the federated database and BioRS. You can specify that predicates get pushed down to the BioRS engine by using the four BioRS custom functions.

All of the custom functions for the BioRS wrapper must be registered with the schema name BIORS. You must include the BIORS schema whenever you use the functions.

The custom functions for the BIORS wrapper are:

- BIORS.CONTAINS
- BIORS.CONTAINS_LE

- BIORS.CONTAINS_GE
- BIORS.SEARCH_TERM

The BioRS CONTAINS functions

The custom functions BIORS.CONTAINS, BIORS.CONTAINS_LE and BIORS.CONTAINS_GE require a search term column argument and a query text argument. The following example shows a BIORS.CONTAINS statement:

```
BIORS.CONTAINS (search_term_column,query_term)
```

The value of the search term column argument must refer to an indexed BioRS column. The use of a non-indexed column produces the error message SQL30090N (″Operation invalid for application execution environment″).

The value of the query term argument is the value that is used to search the indexed element that is specified in the search term column argument.

The value of the query term argument can be only a literal, a host variable, or a column reference. You cannot use arithmetic or string concatenation. Also, the value of the query term argument cannot be NULL, even if the search term column that is used is defined as allowing null values.

The case of the query term argument does not matter.

Valid data types

The valid data types and formats of the query term argument depend on the BioRS data type of the search term column that is used. BioRS defines five possible data types: Text, Author, Date, Number, and Reference.

The BioRS data types and the function query terms that are valid for each data type are listed in the following table.

Table 10. BioRS data types and valid custom function query terms

Data type of search term column	Valid query term	Format
Text	VARCHAR() or CHAR()	BioRS text term, including wildcards.
Author	VARCHAR() or CHAR()	BioRS author reference in the form ″<last>, <init>″. ″<last>″ is the author’s last name. ″<init>″ is the author’s initials, without periods. White space between the comma and initials is accepted. Alternatively, <last> can be specified alone, without the comma or initials.
Date	VARCHAR(), CHAR(), DATE, or TIMESTAMP	If a character string, date format valid for the federated database, yyyy/mm/dd.
Number	VARCHAR() or CHAR(), INTEGER, SMALLINT, BIGINT REAL, DOUBLE, DECIMAL	Number formats valid for the federated database.
Reference	VARCHAR() or CHAR()	BioRS text term.

All other combinations of BioRS data type search term columns and query term arguments produce the error message SQL30090N (″Operation invalid for application execution environment″).

Using wildcard characters

The query term argument for Text, Author, and Reference data type search term columns must match a BioRS query language pattern. In BioRS, query term arguments can consist of alphanumeric strings and wildcards. The BIORS.CONTAINS function supports two wildcards: **?** (question mark) and ***** (asterisk).

The **?** wildcard matches a single character. For example, the predicate `BioRS.CONTAINS (description, 'bacteri?')=1` matches the term bacteria but not the term bacterial.

The ***** wildcard character matches zero or more characters. For example, the predicate `BioRS.CONTAINS (description, 'bacteri*')=1` matches the terms bacteri, bacteria, and bacterial.

For detailed information about BioRS query language patterns, see your BioRS documentation.

Specifying BioRS CONTAINS functions in queries

The BIORS.CONTAINS function can be specified for all BioRS column types.

The BIORS.CONTAINS_GE and BIORS.CONTAINS_LE custom functions can be specified only for columns whose underlying BioRS data type is Number or Date. The BIORS.CONTAINS_GE function selects rows where the column contains a value that is greater than or equal to the value that is represented by the query term argument. The BIORS.CONTAINS_LE function selects rows where the column contains a value that is less than or equal to the value that is represented by the query term argument.

The BIORS.CONTAINS, BIORS.CONTAINS_GE, and BIORS.CONTAINS_LE functions return an integer result. When any of the three CONTAINS functions are used in a predicate, the return value must be compared to the value 1 using the **=** or **<>** operators. For example:

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

The expression `NOT (BioRS.Contains (col,value) = 1)` is equivalent to the expression `BioRS.CONTAINS (col,value) <> 1`.

The BioRS SEARCH_TERM function

The BIORS.SEARCH_TERM custom function requires a search term column argument and a query term. The following example shows the syntax for the SEARCH_TERM custom function:

```
BIORS.SEARCH_TERM (search_term_column,query_term)
```

The value of the search term column argument must refer to the column that represents the _ID_ element.

The value of the query term argument is an expression that can reference multiple elements.

The value of the query term argument can be only a literal, a host variable, or a column reference. You cannot use arithmetic or string concatenation. Also, the value of the query term argument cannot be NULL, even if the search term column that is used is defined as allowing null values.

The case of the query term argument does not matter.

Specifying BioRS SEARCH_TERM function in queries

You can run queries that might not otherwise be possible by issuing the BIORS.SEARCH_TERM function. You can use this function to specify a search term using the BioRS format. The BIORS.SEARCH_TERM function requires two arguments. The first argument is a reference to the _ID_ column of the nickname to which the term is to be applied. The second argument is a character string that contains the term without a databank name.

The following example selects all of the columns for the entries in the MyEMBL databank where the SeqLength element contains a value greater than or equal to 100.

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

The following example selects the MolWeight column from the Swiss nickname where the value of the MolWeight element is greater than or equal to 100368.

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

Equality operations in BioRS queries

You can use an equality operator (=) in literal expressions or in join queries, with certain limitations.

If you use the equality operator in a literal expression or in a join query, equality operator must reference the _ID_ element of a BioRS databank for the query to be pushed down to the BioRS server. Queries that include an equality operator but that do not reference the _ID_ element are not pushed down for processing by the BioRS server.

You can use the equality operator in a literal expression. For example:

```
ID = 'swissprot:100K_RAT'
```

You can use an equality predicate in a join between a BioRS databank and another local table or nonBioRS nicknames. For example:

```
SELECT n.ID, n.EntryDate, t.C1 FROM w46851_n1 n, w46851_t1 t WHERE t.ID = n.ID
```

A join between BioRS databanks must reference the _ID_ element of one databank and a reference type element for the other databank.

However, the use of an equality predicate can return results that differ from the expected results under these conditions:

Case-insensitive matching

The operation is not case sensitive. For example, ID=’100k_rat’ matches both of the following strings:

- ’100k_rat’
- ’100K_RAT’

Wildcard matching

The statement ID='100K_R*' matches both '100K_RAT' and '100K_RODENT.'

Databank prefixing

The operation returns a prefix that indicates the source databank. For example, ID='100K_RAT' in a join on the SwissProt databank might return a value of 'swissprot:100K_RAT.'

Note: Do not create applications that depend on any of the described behaviors.

The following example illustrates the behavior of the equality predicate in a join.

The local table, w46851_t1, contains the following values:

```
ID                              C1
------------------------------ -----------
swissprot:100K_RAT                       0
swissprot:RAT                            1
swissprot:100K_R                         2
swissprot:100K_R*                        3
swissprot:100k_rat                       4
100K_RAT                               100
RAT                                    101
100K_R                                 102
100K_R*                                103
100k_rat                               104
```

You can join the table w46851_t1 with a nickname w46851_n1 that is based on the SwissProt databank. The following statement shows the join query with an equality operation:

```
SELECT n.ID, n.EntryDate, t.C1 FROM w46851_n1 n, w46851_t1 t WHERE t.ID = n.ID
```

The following results are returned:

```
ID                              ENTRYDATE       C1
------------------------------ --------------- -----------
swissprot:100K_RAT              01-NOV-1997               0
swissprot:100K_RAT              01-NOV-1997               3
swissprot:100K_RAT              01-NOV-1997               4
swissprot:100K_RAT              01-NOV-1997             100
swissprot:100K_RAT              01-NOV-1997             103
swissprot:100K_RAT              01-NOV-1997             104

  6 record(s) selected.
```

However, the expected behavior is that only row 0 would be returned.

Equijoin predicates for the BioRS wrapper

You must specify predicates for the BioRS engine when you use the BioRS custom functions, with one exception. The exception is when you perform equijoin operations during a query.

A *join* operation involves retrieving data from two or more tables based on matching column values. An *equijoin* is a join operation in which the join condition has the form expression = expression. For BioRS queries, equijoin terms must contain the _ID_ element of one databank and a Reference type element of another databank.

Example for nickname definitions and an equijoin query

This example shows sample nickname definitions and an equijoin query that uses the sample nicknames.

You want to query two BioRS databanks, SwissProt and SwissProt.features. The SwissProt.features databank is a child of the SwissProt databank, and contains an element called Parent. The Parent element contains references to entries that are identified by the _ID_ element of SwissProt. You register two nickname definitions for the two databanks.

Nickname definition for the SwissProt databank

The SwissProt databank is the parent databank.

```
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');
```

Nickname definition for the SwissProt.features databank

The SwissProt.features databank is a child databank of the SwissProt databank. This nickname contains the **Parent** element.

```
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');
```

The query that references both nicknames in an equijoin

In this query, two predicates are applied to the tc600sprot nickname (SwissProt databank). These two predicates filter the rows that contain the term anopheles and have a publication year of 1997. One predicate is applied to the tc600feat nickname (SwissProt.features databank), which filters those rows whose FtKey element contains the term signal. The two nicknames are joined using the term *f.Parent = s.ID*.

```
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;
```

The final result set contains only the rows that meet these criteria, and the entries in the SwissProt.features databank that reference a matching entry in the SwissProt databank.

The BioRS AllText element

Every databank in the BioRS system contains an element called AllText. The AllText element is an indexed element that BioRS automatically creates for all databanks.

By using the AllText element, you can search on all of the text in an entry, not just on specific indexed elements. For example, searching on the term `muscus` can return entries where the word `muscus` appears in the title, abstract, description, or organism.

To use the AllText element in a federated query, you must map the AllText element to a nickname column. You map the AllText element to a nickname column when you specify columns in the CREATE NICKNAME statement. A nickname column that is mapped to the AllText element returns a NULL value in SELECT statements. When you specify a column as an AllText element, the column must not be the first column declared in a CREATE NICKNAME statement.

After the AllText element is properly mapped to a nickname column, you can use that nickname column in a BIORS.CONTAINS custom function.

BioRS data source - Example queries

These examples include a comprehensive set of sample queries that you can use to access BioRS data sources and show you the statements that are necessary to create the nicknames that are used in the examples.

These examples show you how to:

- Structure your queries to optimize system performance
- Use the custom functions and wildcards in your queries
- Use queries to access specific BioRS data type columns
- Use relational predicates to form an equijoin between parent and child nicknames

The sample queries use the nicknames swiss and swissft.

CREATE NICKNAME statement for the swiss nickname

The parent nickname swiss was registered for the SwissProt databank by using the following CREATE NICKNAME statement:

```
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');
```

CREATE NICKNAME statement for the swissft nickname

The child nickname swissft was registered for the SwissProt.Features databank by using the following CREATE NICKNAME statement:

```
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');
```

Query structures impact federated server performance

The queries and results in the following table show how you can structure your queries to optimize the workload between the federated system and the BioRS server.

Table 11. Samples of different queries that produce identical results

Query	Result

SELECT s.id FROM Swiss s WHERE BIORS.CONTAINS(s.id, ’100K_RAT’) = 1 FETCH FIRST 3 ROWS ONLY;	`ID`
	`---------------`
	`100K_RAT`
	
	`1 record(s) selected.`

SELECT s.id FROM Swiss s WHERE s.id LIKE ’%100K_RAT%’ FETCH FIRST 3 ROWS ONLY;	`ID`
	`---------------`
	`100K_RAT`
	
	`1 record(s) selected.`

Both of the queries in this table produce the same results. However, the first query runs much faster than the second query. The first query uses the BIORS.CONTAINS function to specify the input predicate. As a result, the BioRS server selects the data in the SwissProt databank, then passes the selected data to the federated server. In the second query, the LIKE input predicate is specified directly on the `swiss` nickname. As a result, the BioRS server transfers the entire SwissProt databank to the federated server. After the databank contents are transferred, the federated server then selects the data.

Query performance is typically much better when predicates are sent to the data source for processing.

Queries that use wildcards in the BIORS.CONTAINS custom function

The queries and results in the following table show the use of wildcard characters in the BIORS.CONTAINS custom function. All of the query results are identical, even though different wildcard characters are used.

Table 12. Sample queries that use wildcards in the BIORS.CONTAINS custom function

Query	Result

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 record(s) selected.`

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 record(s) selected.`

Table 12. Sample queries that use wildcards in the BIORS.CONTAINS custom function (continued)

Query	Result

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 record(s) selected`

Queries that access BioRS Author data type columns

The queries and results in the following table show how you can access information in BioRS Author data type elements with the BIORS.CONTAINS custom function. The syntax of all of the queries is nearly identical. The only difference is the presence or absence of the first initial in the query term, and the amount of space between the first name and the last initial.

Table 13. Sample queries that access BioRS Author data type columns

Query	Result

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 record(s) selected.`

SELECT s.authors FROM Swiss s WHERE BIORS.CONTAINS(s.authors, ’Mueller,D’) = 1 FETCH FIRST 3 ROWS ONLY;	`AUTHORS`
	`---------------`
	
	`  0 record(s) selected.`

SELECT s.authors FROM Swiss s WHERE BIORS.CONTAINS(s.authors, ’Mueller ,D’) = 1 FETCH FIRST 3 ROWS ONLY;	`AUTHORS`
	`---------------`
	
	`  0 record(s) selected.`

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 record(s) selected.`

Queries that access BioRS Date data type columns

The queries and results in the following table who how you can access information in BioRS Date type elements with the BIORS.CONTAINS custom function.

When a BioRS Date type field contains a sequence of dates, the results can contain extra information, as shown in the second example in the table. BioRS Numeric data type elements (Date and Number) can contain multiple values. Therefore, the results of the queries that are run on BioRS Date or Number elements can also contain multiple values. Multiple values are always separated by spaces.

Table 14. Sample queries that access BioRS Date data type columns

Query	Result
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 record(s) selected.
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 record(s) selected.

Queries that use the BIORS.CONTAINS_LE and BIORS.CONTAINS_GE custom functions

The queries and results in the following table show how you can use the BIORS.CONTAINS_LE and the BIORS.CONTAINS_GE custom functions.

Table 15. Sample queries that use the BIORS.CONTAINS_LE and BIORS.CONTAINS_GE custom functions

Query	Result
SELECT s.molweight FROM Swiss s WHERE BIORS.CONTAINS_LE(s.molweight, 100368) = 1 FETCH FIRST 3 ROWS ONLY;	MOLWEIGHT --------------- 100368 10576 8523 3 record(s) selected.
SELECT s.molweight FROM Swiss s WHERE BIORS.CONTAINS_GE(s.molweight, 100368) = 1 FETCH FIRST 3 ROWS ONLY;	MOLWEIGHT --------------- 100368 103625 132801 3 record(s) selected.
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 record(s) selected.

Queries that use the BIORS.SEARCH_TERM custom function

The queries and results in the following table show how you can use the BIORS.SEARCH_TERM custom function to specify a search term using the BioRS format.

Table 16. Sample queries that use the BIORS.SEARCH_TERM custom function

Query	Result

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 record(s) selected.`

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 record(s) selected.`

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 record(s) selected.`

Using relational predicates to form an equijoin between two databanks that have a parent-child relationship

The following query shows how to use relational predicates to form an equijoin between two databanks that have a parent-child relationship:

```
SELECT s.id, f.id, f.parent FROM Swiss s, Swissft f
    WHERE (f.parent = s.id) FETCH FIRST 10 ROWS ONLY;
```

In the following query results, the 100K_RAT record is a parent to nine child records (100K_RAT.1 through 100K_RAT.9).

```
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 record(s) selected.
```

Optimizing BioRS wrapper performance

You can improve query performance to BioRS data sources by optimizing the performance of the BioRS wrapper.

Guidelines for optimizing BioRS wrapper performance

The structure of your queries and statistical information about the BioRS databanks impact query performance.

Minimize the amount of data that is transferred between search engines.
The federated environment uses two search engines. For the BioRS wrapper, these search engines are the federated database and BioRS. Some configurations include more than one federated database engine. The federated database engine processes predicates (relational operators, such as =, BETWEEN, LIKE, and <>) specified on nickname columns. The BioRS engine processes predicates that are specified using four custom functions for the BioRS wrapper.

To minimize the amount of data that is transferred between the two search engines, structure your queries so that data processing gets pushed down to the BioRS system whenever possible.

If you need to perform join operations in a query, take advantage of any parent-child relationships that already exist in the BioRS databanks and perform equijoin operations whenever possible. Equijoin operations are processed in BioRS, which also minimizes the amount of data transferred between the federated database and BioRS search engines.

Important: Do not interrupt federated queries to BioRS, for example using Ctrl-D or Ctrl-Z in the command line processor, or stopping an application program. Interrupting a query leaves ″dead″ processes running on the BioRS server. These ″dead″ processes will rapidly degrade both the BioRS server and the federated server performance. If enough of these ″dead″ processes are running, unexpected errors can occur during federated query processing. For example, a valid query might return 0 rows, when rows are expected. In extreme situations, the BioRS server, the federated server, or both servers can stop or abnormally end.

Maintain BioRS statistical information in the federated environment.
In a federated system, the federated database relies on catalog statistics for nicknamed objects to optimize query processing. Maintaining current statistics about the BioRS data sources is essential to optimize the performance of the BioRS wrapper. If the statistical data or structural characteristics for a remote object on which a nickname is defined have changed, you must update the corresponding nickname column cardinality statistics in your federated system.

To optimize BioRS wrapper performance, perform these updates on the federated server at regular intervals.

BioRS statistical information

Current BioRS statistical information is essential for optimizing the performance of the BioRS wrapper.

In a federated system, the federated database relies on catalog statistics for objects with nicknames to optimize query processing. These statistics are retrieved from BioRS data sources when you register a nickname using the CREATE NICKNAME statement. The federated database verifies the presence of the object at the data source, and then attempts to gather existing data source statistical data. Information is read from the data source catalogs and put into the system catalog in the federated database.

For BioRS data sources, critical statistical information includes:

- The cardinality of a nickname. For BioRS data sources, nickname cardinality is equivalent to the number of entries in the corresponding BioRS databank.
- The cardinality of the column that corresponds to the BioRS _ID_ element. The cardinality of this column must match the cardinality of the nickname in which the column is referenced.
- The cardinality of all columns that the BioRS wrapper might need to use.

Tip: to optimize the performance of the BioRS wrapper, you should maintain current statistics about the BioRS data sources. If the statistical data or structural characteristics change for a remote object on which a nickname is defined, you must update the corresponding cardinality statistics in your federated system. The cardinality statistics are stored in the federated database system catalog in the SYSSTAT.TABLES view and in the SYSSTAT.COLUMNS view.

To maintain BioRS cardinality statistics in your federated system:

1. Determine the cardinality statistics for the nickname, if necessary.
2. Update the cardinality statistics for the nickname in the catalog views.
3. Update the cardinality statistics for the columns in the catalog views.

Determining BioRS databank cardinality statistics

You must determine BioRS databank cardinality statistics before you can update nickname statistics or update the cardinality of the column that corresponds to the BioRS _ID_ element.

Procedure

To determine cardinality statistics for a specific databank in BioRS:

Use either the BioRS admin_find utility program or the www_find.cgi utility program and specify the `-c` option, which is for cardinality . For more information about these two BioRS utility programs, see your BioRS documentation.

Updating BioRS nickname cardinality statistics

You must update the nickname cardinality statistics when the contents of a BioRS databank changes significantly.

Before you begin

You must determine the cardinality number of the BioRS databank that corresponds to the nickname whose statistics you want to update.

About this task

To update the BioRS nickname cardinality statistics in your federated system, you must modify the SYSSTAT.TABLES catalog view.

Maintaining correct cardinality statistics for nicknames enables the optimizer and the BioRS wrapper to choose the best performing data access plan.

Procedure

To update the BioRS nickname cardinality statistics:

Issue the UPDATE statement to modify the SYSSTAT.TABLES catalog view and specify the correct cardinality number. The syntax of the UPDATE statement is:

```
UPDATE SYSSTAT.TABLES SET CARD=cardinality_number
    WHERE TABSCHEMA=nickname_schema
    AND TABNAME=nickname_name;
```

For example, if the nickname is JONES.SWISS, you use the following UPDATE statement to update the statistics:

```
UPDATE SYSSTAT.TABLES SET CARD=15312191
    WHERE TABSCHEMA='JONES'
    AND TABNAME='SWISS';
```

SYSSTAT.TABLES
The system catalog view in the federated database where nickname statistics are stored.

SET CARD=15312191
The BioRS databank cardinality number that corresponds to the nickname that you are updating the statistics for.

TABSCHEMA= *'JONES'*
The name of the schema for the nickname that you want to update.

TABNAME=*'SWISS'*
The name of the nickname that you want to update.

Updating BioRS column cardinality statistics

To update BioRS column cardinality statistics in your federated system, you must modify the SYSSTAT.COLUMNS catalog view.

You can update BioRS column cardinality statistics before you create the BioRS nicknames or you can update BioRS column cardinality statistics when you want to improve query performance for BioRS data sources.

Restrictions

Do not use this procedure to update the cardinality statistics for columns that correspond to the BioRS _ID_ element. You must use a different procedure to update the cardinality statistics for columns that correspond to the BioRS _ID_ element.

You should ensure that the cardinality statistics for BioRS columns are current so that the optimizer and the BioRS wrapper can choose the best data access plan during query processing.

About this task

Procedure

To update BioRS column cardinality statistics:

Issue the UPDATE statement to modify the SYSSTAT.COLUMNS catalog view. The syntax of the UPDATE statement is:

```
UPDATE SYSSTAT.COLUMNS SET COLCARD=(SELECT COUNT(DISTINCT column_name)
    FROM nickname_schema.nickname_name)
    WHERE
    TABSCHEMA=nickname_schema
    AND TABNAME=nickname_name
    AND COLNAME=column_name;
```

SYSSTAT.COLUMNS
The system catalog view in the federated database where column statistics are stored.

COLCARD=(SELECT COUNT(DISTINCT *column_name*
The name of the column in the nickname that you are updating the statistics for.

TABSCHEMA= *nickname_schema*
The name of the schema for the nickname that you want to update.

TABNAME=*nickname_name*
The name of the nickname that you want to update.

COLNAME=*column_name*
The name of the column whose cardinality statistics you want to update.

The query might take several minutes to run because all of the entries for the databank that is associated with the nickname must be retrieved.

If a column contains multiple values (for example, the PublicationYear element of the SwissProt database format), the calculation becomes too complex to use an SQL query. For such columns, you must manually calculate the cardinality value, and then update the SYSSTAT.COLUMNS catalog view. To calculate the cardinality value, divide the number of distinct values in the column by the average number of values per row. The calculated cardinality value cannot be greater than the cardinality of the table.

For example, if the nickname has the following three rows of values for the PublicationYear column, there are nine distinct values and the average number of values in a row is four.

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

The cardinality for this PublicationYear column is 9 divided by 4, or 3 (2.25 rounded to the next highest integer). You can update the SYSSTAT.COLUMNS catalog view using the following UPDATE statement:

```
UPDATE SYSSTAT.COLUMNS SET CARDCOL=3
    WHERE
    TABSCHEMA=nickname_schema
    AND TABNAME=nickname_name
    AND COLNAME=column_name;
```

Updating BioRS _ID_ column cardinality

To update BioRS column cardinality statistics for the column that maps to the BioRS _ID_ element, you must modify the SYSSTAT.COLUMNS catalog view.

Before you begin

You must determine the cardinality number of the BioRS databank that corresponds to the nickname in which the column is referenced. The cardinality number of the column that maps to the BioRS _ID_ element must match the cardinality of the nickname in which the column is referenced.

You should ensure that the cardinality statistics for the column that maps to the BioRS _ID_ element are current. The optimizer and the BioRS wrapper use these statistics to choose the best data access plan to process your queries.

About this task

To update the BioRS _ID_ column cardinality you must select entries in the SYSCAT.COLOPTIONS view that contain the ELEMENT_NAME option. This BioRS wrapper uses this option to map between nickname column names in the federated database and element names in the BioRS server.

Procedure

To update BioRS _ID_ column cardinality statistics:

Issue the UPDATE statement to modify the catalog view.

For example, :

```
UPDATE SYSSTAT.COLUMNS SET COLCARD=cardinality_number
    WHERE
    TABSCHEMA=nickname_schema
    AND TABNAME=nickname_name
    AND COLNAME=column_name
        IN (SELECT column_name FROM SYSCAT.COLOPTIONS
        WHERE
        TABSCHEMA=nickname_schema
        AND TABNAME=nickname_name
        AND OPTION='ELEMENT_NAME';
        AND SETTING='_ID_')
```

SYSSTAT.COLUMNS
The system catalog view in the federated database where column statistics are stored.

SET COLCARD=*cardinality_number*
The BioRS databank cardinality number that corresponds to the nickname of the column that you are updating the statistics for.

TABSCHEMA= *nickname_schema*
The name of the schema for the nickname that you want to update.

TABNAME=*nickname_name*

The name of the nickname that you want to update.

COLNAME=*column_name*

The name of the column whose cardinality statistics you want to update.

IN (SELECT *column_name* **FROM SYSCAT.COLOPTIONS**

This SELECT statement determines the name of the column that maps to the BioRS _ID_ element. SYSCAT.COLOPTIONS is the system catalog view in the federated database where column options are stored.

OPTION=*′ELEMENT_NAME′*

The value in the rows in the SYSCAT.COLOPTIONS view which indicates that a nickname column name is mapped to BioRS element name.

SETTING=*′_ID_′*

Specifies that the column for the ELEMENT_NAME option is ′_ID_′.

Chapter 5. BLAST data sources

Configuring access to BLAST data sources

You can integrate the data that is in BLAST data sources with information from other sources by using a federated system.

Procedure

To configure a federated server to access BLAST data sources, you must provide the federated server with information about the data sources and objects that you want to access. After you configure the federated server, you can create queries and use the custom functions to access the BLAST data sources.

BLAST wrapper

BLAST is a utility that is used to scan a nucleotide or amino acid sequence database for ″hits.″ The BLAST wrapper is used to perform searches on BLAST-able data sources.

BLAST (Basic Local Alignment Search Tool) is a utility that is maintained by the National Center for Biotechnology Information (NCBI). BLAST is used to scan a nucleotide or amino acid sequence database for ″hits.″ A BLAST hit contains one or more high-scoring segment pairs (HSPs). A HSP is a pair of sequence fragments, whose alignment is locally maximal, and whose similarity score exceeds some threshold value. NCBI provides the blastall executable file that is used to perform BLAST searches on BLAST-able data sources, such as GenBank and SWISS-PROT.

The BLAST wrapper supports all five types of BLAST searches: BLASTn, BLASTp, BLASTx, tBLASTn, and tBLASTx.

Table 17. BLAST search types supported by the BLAST wrapper

BLAST search type	Description
BLASTn	A type of BLAST search in which a nucleotide sequence is compared with the contents of a nucleotide sequence database to find sequences with regions homologous to regions of the original sequence.
BLASTp	A type of BLAST search in which an amino acid sequence is compared with the contents of an amino acid sequence database to find sequences with regions homologous to regions of the original sequence.
BLASTx	A type of BLAST search in which a nucleotide sequence is compared with the contents of an amino acid sequence database to find sequences with regions homologous to regions of the original sequence. The query sequence is translated in all six reading frames, and each of the resulting sequences is used to search the sequence database.
tBLASTn	A type of BLAST search in which an amino acid sequence is compared with the contents of a nucleotide sequence database to find sequences with regions homologous to regions of the original sequence. The sequences in the sequence database are translated in all six reading frames, and the resulting sequences are searched for regions homologous to regions of the query sequence.

Table 17. BLAST search types supported by the BLAST wrapper (continued)

BLAST search type	Description
tBLASTx	A type of BLAST search in which a nucleotide sequence is compared with the contents of a nucleotide sequence database to find sequences with regions homologous to regions of the original sequence. In a tBLASTx search, both the query sequence and the sequence database are translated in all six reading frames, and the resulting sequences are compared to discover homologous regions.

The following figure shows how BLAST works with your federated system.

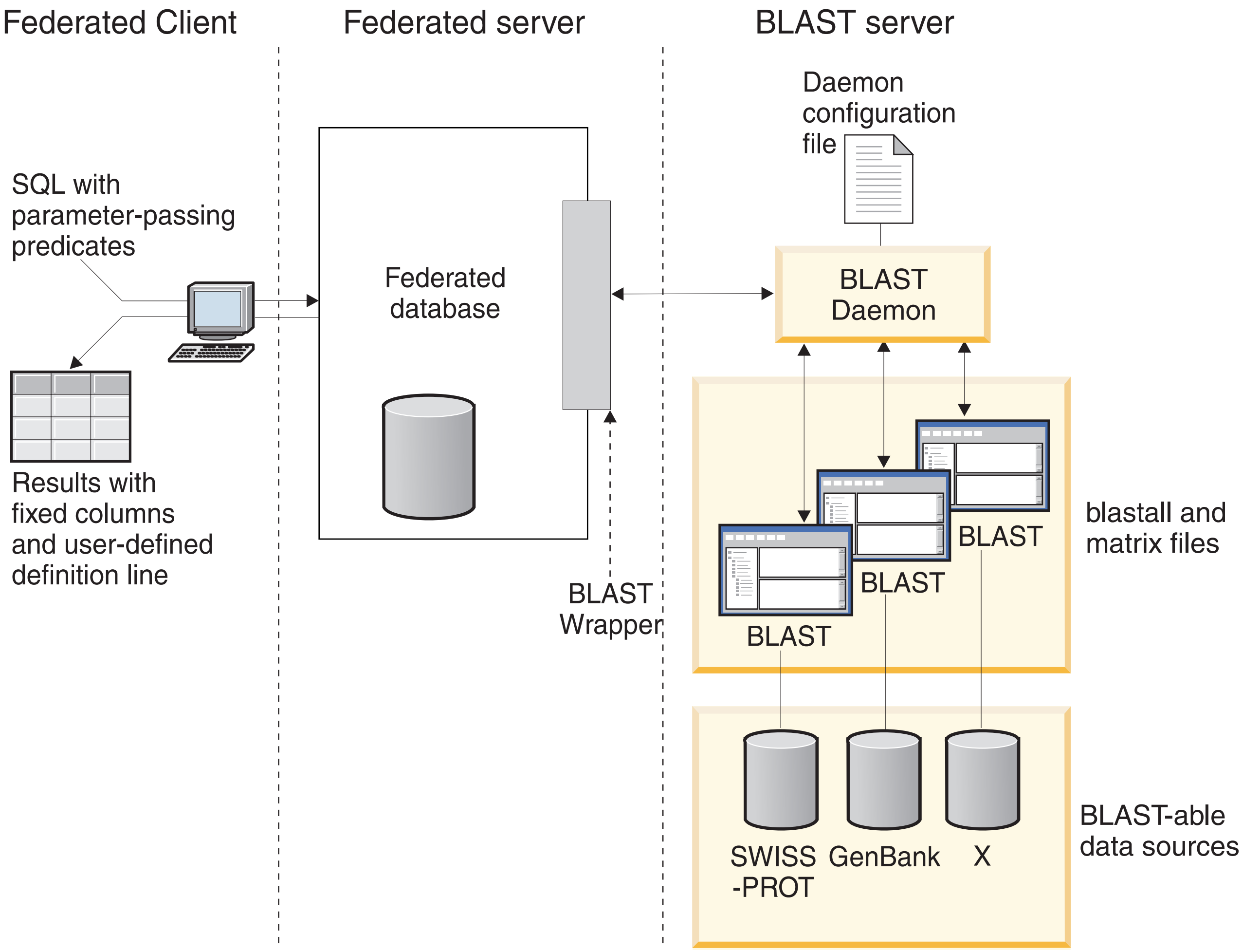

Federated Client
Federated server
BLAST server
Daemon configuration file
SQL with parameter-passing predicates
Federated database
BLAST Daemon
Results with fixed columns and user-defined definition line
BLAST
BLAST
BLAST
blastall and matrix files
BLAST Wrapper
SWISS -PROT
GenBank
X
BLAST-able data sources

Figure 5. How the BLAST wrapper works

On the client side, users or applications submit SQL statements with BLAST-specific parameter-passing predicates that map to standard BLAST options. The SQL statements with the input predicates are sent to your federated system with the BLAST wrapper installed.

The BLAST wrapper transforms the query into a format understandable by the BLAST application and sends the transformed query to your BLAST server. This server can be on a separate computer from the computer for the federated server. A special daemon program runs on your BLAST server. This daemon, using information from a daemon configuration file, receives the query request from the federated server and sends it to the BLAST application. The BLAST application then runs against a BLAST-able data source in the usual manner.

The results are returned to BLAST and then to the daemon. The daemon returns the retrieved data to the BLAST wrapper. The wrapper transforms the data into a relational table format, and returns this table to you or application. The returned data contains two parts:

- A series of standard, fixed columns familiar to BLAST users, and
- User-configured definition line information.

The following example illustrates how relational information is extracted from BLAST-able data sources. Data moves from raw FASTA file format to a BLAST-able data set to a relational table that can be joined with other data sources in your federated system.

The following figure shows a sample FASTA file that contains four definition line and nucleotide sequence records.

```
>7:4986 PMON5744
GTTCTTCCCAGTGCCCAAGTCCATTCTGACATCAATGAAGAAGGTAAAATCCCTGCGTGATCCCTCTGCC
AAGATGTCGAAATCAGACCCGGATAAACTAGCTGCTGTCAGAATAACAGACAGCCCGGAGGAGATCGTGC
AGAAGTTCCGCAAGGCTGTGACGGACTTCACCTCGGAGGTCACCTACGACCCGGCCAGGCGAGGAGGCGT
GTCCAACTTGGTGGCCATCCACGCGGCAGTGACCGGACTCCCGGTGGAGGAGGTGGTCCGCCGAAGTGCT
GGCATCAACACCGCTGGCTACAAGTTGGTGGTGGCGGAGGCTGTGATTGAGAGATTTGCACCAATTAAGA
GTGAAATTGAAAAACTGAAGAGGAACAAGGACCACCTAGAGAAGGTTTTACAAGTTGGGTCGGCAAAAGC
CAAAGAATTAGCATATCCCGTGTGCCAGGAGGTGAAGAAATTGGTGGGGTTTCTATAGGCAGTCTCACCT
AGTCCCAGAAAATGTTTTTTATCTTGTGGTCTGCTTGCACACTCAGTCTAATAAAGGCAGCTTTCCTAAG
ACGCCAACAATTCCAGTTTGGGGATGCTTAGTTTACT
>8:9747 PMON5699
AAGAAGTTCTTGTTAGAACTTTCCACCTCCGGCTTCCCCTCCACCTCTCTTACTGTCCCAACCTTCTGAG
ACGCTTTTTCTCCTCCCGAGGATTTATCTCTTTCTCTCTCTCTCTCTCTCTCTCTTTTTTTTTTTCCCCT
TTTCCCCCCCCGAGGCTGGTTTTGCTTTGGGGAGGGGGGGTTTTTTAAAGGGGCCGGGGGGGCCCCCTTT
CTCCCCCCCTAATGGGGTTAATTAATAATGGGGGGGGGGTTTTTTTTTTTTAAACCCCTATTTGGTCCGG
CCCGGGGATTTCCCCCCCCCCCCCCTTGCCCGGTTCCGGGGCCCGGAGGAGGGGGGGAAAAGGGCGGGAA
CCTTTGGTAGTTTCCCCTCGGAAAAAAAATTTTTCGGGGGGGAAAACCTCCCT
>13:6512 PMON5498
GATAAGAGGCAGAATAGAAGACTGGACTACTTCTCTCCTAAAAACACATTTAAAACTAAGCCTGAGCAAT
CTCCACCCAAATGGACCGGAAACCTTAAAAAAGAATCCTACTCCTGAAGAAAAAGAGGAGGACACATCAA
GAGGTAGAAGGGGCGATTTCATGATATAAACAACCCCATACCTCCAGAGTGGGAAGCTCCACAGACTGAA
AACTAACTGGTTCACAGAAACTCACCTACAGGAGTGAGCCCCACATCAAACCCTCGAATGTGGGGATCTG
GCACTGGTAGAAAGAGCCCCTGGAGCATCTGGCATTGAAGGCCAGTGGGGCTTGTGTGCAGGAGATCCAC
AGGACTAGGGGAAACGGAGACCCCCATTCTTAAAAGGTGCACACAGACTTTTACGTGCACTGGGTCCCAG
TGCAAAGCAAAGTCTCCATAGGAATCTGGGTCAAACCTGACTGCAGTTCTTGGAGGACCTCCTGGGAAAG
CAAGGGTGAATGTGGCTTCTTGTGGGGAAAGGACATTGGAAGCAAAGCTCTTGGGAATATTCATCAGTGT
GC
>15:8924 PMON5426
GGAGAAACTGACTCCTGAGCAGCTGCAATTCATGCGGCAGGTGCAGCTCGCCCAGTGGCAGAAGACGCTG
CCACAGCGGCGGACCCGGAACATCGTGACCGGCCTGGGCATCGGGGCGCTGGTGTTGGCAATTTGTATCC
GTTTGGACTGTAGACTCAGGGAGACCGCATTTAGGGGAACAGGAAGGGCAGCAGGGGCGTGTAGGAGGGC
AGTGTGGGGGTGGTAGAAGGAGCCCGAGATATGAAAACCTTGGCTCCTTTTAACTCTGAATCAAGCGTTT
GGTGTACCTTACGTTGTCATTTTAAAGGTGTATTTTAGTATAATTGATTAATGATTACGGAGTCGGGTGA
GGGCTCCCAGGAGCAGACGGCAGAAGATCGAATTTGGGAGGATGATCAGCAGCGGTGGTTGAGCAAGTGT
GGGAAAAGGGAATGCGCACATTCCACGTGGTTTCCTGAACCCACCTCCCCAGATGGTTACACCTTCTACT
CGGTGTCCCAGGAGCGTTTCTTGGATGAGCTGGAGGATGAGGCCAAAGCTGCTC
```

Figure 6. Sample FASTA file, nucleo1

The standard formatdb application from NCBI transforms the FASTA file to a BLAST-able data set. The data is then ready to be queried.

The following query is transformed by the BLAST wrapper and runs against the BLAST-able data set.

```
SELECT Unique_ID, Experiment_Number, Organism_Number, HSP_Info, Score
FROM nucleo1
WHERE BlastSeq = 'ACATTCTTATAGAGTATTGCTACTCCTCCAGGATAGAGTCATCTCT
 GGTCTCCAGAGCCACCGCTGGCTACAAGTTGGTGGTGGCGGAGGCTGTGATTGAGAGATTTG
 CACCAATACAGAAACTCACCTACAGGAGTGAGCGGGTGGTAGAAGGAGCCCGAGATATGAAA
 ACCTTGTTTCAAGACCCCATTGTCACCGGGG';
```

The results of the query are transformed by the BLAST wrapper into a relational table format shown in the following table.

Table 18. BLAST returns results in relational table form when integrated into your federated system

Unique ID	Experiment number	Organism number	HSP_INFO	SCORE
PMON5744	4986	7	Identities = 57/201 (28%), Positives = 57/201 (28%), Gaps = 0/201 (0%)	+1.13487000000000E+002
PMON5426	8924	15	Identities = 35/201 (17%), Positives = 35/201 (17%), Gaps = 0/201 (0%)	+6.98754000000000E+001
PMON5498	6512	13	Identities = 26/201 (13%), Positives = 26/201 (13%), Gaps = 0/201 (0%)	+5.20342000000000E+001

The data is in a fully relational form and can be joined with data from other data sources used by your laboratory. Combining the results of several data sources can lead to insights not readily or efficiently discovered prior to the implementation of your federated system.

Adding BLAST data sources to a federated server

To configure a federated server to access BLAST data sources, you must provide the federated server with information about the data sources and objects that you want to access. To configure a federated server to access BLAST data sources, you must configure the BLAST daemon, a wrapper, a server definition, and nicknames for the BLAST searches.

Before you begin

- IBM WebSphere Federation Server must be installed on a server that will act as the federated server
- A federated database must exist on the federated server

About this task

You can configure a federated server to access data that is stored in BLAST data sources by using the Control Center or by issuing SQL statements on the command line. The Control Center includes a wizard to guide you through the steps that are necessary to configure the required federated objects.

Procedure

To add BLAST data sources to a federated server:

1. Verify that the correct version of the blastall executable and matrix files are installed.
2. Configure the BLAST daemon.
3. Start the BLAST daemon.
4. Register the wrapper.
5. Register the user mappings (optional).
6. Register the server definition.
7. Register nicknames for BLAST searches.

Verifying the versions of the BLAST server files

You must have the latest version of the blastall executable file and the matrix file installed on your BLAST server.

Before you begin

Verify that you have:

- The latest version of the blastall executable file installed on your BLAST server computer.
- The BLOSUM62, BLOSUM80, PAM30, and PAM70 matrix files installed on your BLAST server computer. The matrix files must be in the same directory as the blastall executable file.

Procedure

To check the version level of your blastall executable and matrix files:

Run a BLAST search from the command line and note the version number located in the output file.

Configuring the BLAST daemon

A BLAST daemon is used by the BLAST wrapper to access BLAST data sources. You must configure the BLAST daemon before you register the BLAST wrapper.

Before you begin

The BLAST daemon must have:

- Execute access to the blastall executable file so that it can run BLAST searches.
- Write access to a directory in which it can write temporary files.
- Read access to at least one BLAST-able data source on which BLAST searches can be run. The blastall executable file must have read access to both the data file and the BLAST index files that are generated by the formatdb program. The formatdb program is an executable program that is part of the BLAST utility.

Restrictions

The BLAST daemon might not run properly if the executable file path contain spaces. For example, you should not install the BLAST executable file in C:\Program Files on Windows servers.

About this task

The BLAST wrapper requires a BLAST daemon. The BLAST daemon must be running on a server that you can access through TCP/IP from your federated system. The following files must be on the same server, whether they are on the federated server, or a separate BLAST server for the blastall executable and daemon to work correctly:

- The blastall executable file
- The MATRIX files
- The daemon executable file
- The daemon configuration file
- The OSS runtime library file

The BLAST daemon runs separately from the BLAST wrapper and the federated database. The BLAST daemon listens for BLAST job requests from the BLAST wrapper.

Procedure

To configure the BLAST daemon:

1. Ensure that the BLAST daemon executable files are on the correct server.

Operating system	Procedure
On federated servers that run UNIX	The name of the BLAST daemon executable file is db2blast_daemon. This file is installed in the $DB2PATH/bin directory.
On federated servers that run Windows	The names of the BLAST daemon executable files are db2blast_daemon.exe and db2blast_daemon_svc.exe. These files are installed in the %DB2PATH%\bin directory.

   It might be necessary to copy the BLAST daemon executable files to another server.

   During the installation of IBM WebSphere Federation Server, the BLAST daemon executable files are installed on the federated server.

   If you use a separate BLAST server, you must copy the BLAST daemon executable files from the federated server to the BLAST server. The BLAST daemon executable files can run in any directory on the BLAST server that does not contain spaces in the names in the directory path.

2. Ensure that the OSS library file is on the correct server.

Operating system	Procedure
On federated servers that run UNIX	The name of the BLAST daemon executable file is: • libdb2osse.a on AIX operating systems • libdb2osse.so on Linux or Solaris operating systems This file is installed in the $DB2PATH/bin directory.
On federated servers that run Windows	The name of the OSS library file is db2osse.dll. This files is installed in the %DB2PATH%\bin directory.

3. Ensure that the BLAST daemon configuration file is on the correct server.

Operating system	Procedure
On federated servers that run UNIX	The daemon configuration file is installed in the $DB2PATH/bin directory. $DB2PATH is the directory in which IBM WebSphere Federation Server is installed.
On federated servers that run Windows	The daemon configuration file is installed in the %DB2PATH%\bin directory. %DB2PATH% is the directory where IBM WebSphere Federation Server is installed. The default path is C:\Program Files\SQLLIB\IBM.

   When IBM WebSphere Federation Server is installed, a sample daemon configuration file is installed on the federated server. The name of the sample daemon configuration file is BLAST_DAEMON.config.

   The configuration file must be on the same server where the daemon is running. You can copy the configuration file to another location. If you use a BLAST server, you must copy the daemon configuration file from the directory on the federated server to a directory on the BLAST server. You can copy the daemon configuration file to any directory on the BLAST server that the daemon can access.

4. Edit the daemon configuration file.
   a. Optional: You can rename the configuration file.
   b. Ensure that the first line in the configuration file is an equal sign. If the equal sign is missing, the daemon will not start. An error message will indicate that the DAEMON_PORT was not specified.
   c. Ensure that the last line in the configuration file ends with a new line. The sample configuration file that is provided with IBM WebSphere Federation Server ends with a new line. If the last line does not end with a new line, you will receive an error message when you attempt to run your first BLAST query using the data source that is listed on the last line.
   d. Specify the following options in the configuration file.

      For options that require paths, you can specify relative paths. Relative paths are relative to the directory from which the daemon process was started.

      DAEMON_PORT
      This is the network port on which the daemon listens for BLAST job requests submitted by the wrapper.

      MAX_PENDING_REQUESTS
      This is the maximum number of BLAST job requests that can be blocking on the daemon at any one time. This number does not represent the number of BLAST jobs that are running concurrently, only the number of job requests that can block at one time. It is recommended that you set this to a number greater than five. The BLAST daemon does not restrict the number of BLAST jobs that can run concurrently.

      DAEMON_LOGFILE_DIR
      This is the directory in which the daemon creates its log file. This file contains useful status and error information generated by the BLAST daemon.

Q_SEQ_DIR_PATH

This is the directory in which a temporary query sequence data file is created by the daemon. This temporary file is cleaned up once the BLAST job completes.

BLAST_OUT_DIR_PATH

This is the directory in which the daemon creates the temporary file to store the BLAST output data. Data is read from this file and passed back to the wrapper through the network connection. After the data is passed to the wrapper, the daemon cleans up the temporary file

BLASTALL_PATH

This is the fully-qualified name of the BLAST executable file that is on the server that is running the daemon.

database specification entry

Specifies the location of a BLAST-able data source. Make note of the database *data_source_name* that you specify in the configuration file. For the daemon to function properly, you must specify the database *data_source_name* when you create the nickname for the data source. The name is case-sensitive. The database *data_source_name* is specified in the DATASOURCE option of the CREATE NICKNAME statement.

The configuration file must contain at least one database specification entry that specifies the fully qualified path for the BLAST-able data source. For example:

```
data_source_name=data_source_path
```

On federated servers that run UNIX

For example, to specify the GenBank BLAST-able data source you would add the following line to the daemon configuration file:

```
genbank=/dsk/1/nucl_data/genbank
```

On federated servers that run Windows

For example, to specify the GenBank BLAST-able data source you would add the following line to the daemon configuration file:

```
c:\vnr_data\genbank_nonest1.fasta
```

The path indicated in a database specification entry must contain the three index files.

- For nucleotide data sources, the index files have these extensions:
  - .nhr
  - .nin
  - .nsq
- For amino acid data sources, the index files have these extensions:
  - .phr
  - .pin
  - .psq

The database specification entry must indicate the name of the file that contains the original FASTA-formatted data. The three index files must have the same root name as the file containing the original FASTA-formatted data.

BLAST daemon configuration file - examples

The examples show the required settings and specifications for GenBank and SWISS-PROT data sources. There are daemon configuration file examples for both UNIX and Windows servers.

The following examples show the contents of a sample BLAST daemon configuration file. The BLAST daemon can run on the federated server or on a separate BLAST server. The BLAST daemon configuration file must be on the same server as the BLAST daemon and BLAST executable files.

BLAST daemon configuration file (UNIX)

This example shows the required options and the BLAST-able data source specifications for GenBank and SWISS-PROT for a server that runs UNIX.

```
=
DAEMON_PORT=4007
MAX_PENDING_REQUESTS=10
DAEMON_LOGFILE_DIR=/home/blastuser/logs
Q_SEQ_DIR_PATH=/tmp
BLAST_OUT_DIR_PATH=/tmp
BLASTALL_PATH=/home/blastuser/blast/blastall
genbank=/dsk/1/nucl_data/genbank
swissprot=/dsk/1/prot_data/swissprot
```

BLAST daemon configuration file (Windows)

This example shows the required options and the BLAST-able data source specifications for GenBank and SWISS-PROT for a server that runs Windows.

```
=
DAEMON_PORT=4007
MAX_PENDING_REQUESTS=10
DAEMON_LOGFILE_DIR=C:\blast\logs
Q_SEQ_DIR_PATH=C:\temp
BLAST_OUT_DIR_PATH=C:\temp
BLASTALL_PATH=C:\blast\blastall.exe
genbank=c:\vnr_data\genbank_nonest1.fasta
swissprot=c:\vnr_data\swissprot
```

Starting the BLAST daemon

You must start the BLAST daemon before you register the BLAST wrapper.

Before you begin

Before you start the BLAST daemon:

- You must configure the BLAST daemon
- You must have write access to all of the paths that are listed under the following entries in the configuration file
  - DAEMON_LOGFILE_DIR
  - BLAST_OUT_DIR_PATH
  - Q_SEQ_DIR_PATH

About this task

The executable file starts a new process in which the BLAST daemon runs.

Procedure

To start the BLAST daemon:

1. Open the directory where the daemon executable files are located.
2. Issue the `db2blast_daemon` command to run the executable files.

You might want to include one or more options when you issue the command to run the executable files.

db2blast_daemon command - options and examples

The `db2blast_daemon` command can be used on UNIX and Windows servers. Some of the options listed in the syntax can be used only on Windows servers.

Options - db2blast_daemon command

The options for the `db2blast_daemon` command are:

```
db2blast_daemon -a action -c config_file -d debug_level
    -u user_id -p password
```

-a *action*

Performs the specified activity. Valid actions are *status*, *install*, *start*, *stop*, and *remove*.

You can specify this option only on Windows servers.

-c *config_file*

Instructs the daemon to use the specified configuration file. If you do not specify the configuration file, the daemon searches for the `BLAST_DAEMON.config` file in the directory where the daemon executable files are installed.

You can specify this option on UNIX and Windows servers. On Windows servers, you can specify this option only with the *install* or *start* actions.

-d *debug_level*

Sets the daemon debug level to the specified value. The valid values are 1, 2, or 3. You can use this option with the *install* and *start* actions.

You can specify this option on UNIX and Windows servers.

-u *user_id*

Sets the daemon to run under the specified user ID. You can use this option with the *install* action.

You can specify this option only on Windows servers.

-p *password*

Specifies the password for the specified user ID. The password is valid and required only when you specify the -u option. If the -p option is not specified when you set the -u option, the program prompts you for the password. You can use this option with the *install* action.

You can specify this option only on Windows servers.

The options that are specified with the *start* action apply only until the daemon is stopped, and override the values that are specified with the *install* action. If the daemon is stopped and then started without the option, then the value of the option is whatever was specified with the install action, or the built-in default for the daemon.

For example, if the commands are issued in the following order, the BLAST1.config file is specified with the *install* action. The BLAST2.config file is specified with the *start* action and used until the daemon is stopped. When the daemon is started again, the BLAST1.config file is used.

```
[1]db2blast_daemon -a install -c BLAST1.config
[2]db2blast_daemon -a start -c BLAST2.config
[3]db2blast_daemon -a stop
[4]db2blast_daemon -a start
```

Examples - db2blast_daemon command

The following examples show you how to use the options with the db2blast_daemon command.

Start the daemon

To start the daemon, issue the following command:

```
db2blast_daemon -a start
```

This command assumes that the daemon configuration file is in the same directory as the executable file.

Daemon configuration file

If you changed the name of the daemon configuration file or if the configuration file is not in the same directory as the daemon executable file, you must use the -c option when you run the executable file. This option specifies the directory path and name for the daemon configuration file.

For example, if the daemon configuration information in a file called BLAST_D.config in the subdirectory cfg on a UNIX server, issue the following command:

```
db2blast_daemon -c cfg/BLAST_D.config
```

Debugging

If you want to start the daemon with debugging turned with a debug level of 2, issue the following commands:

```
db2blast_daemon -a install -d 2

db2blast_daemon -a start
```

Status of the daemon (Windows)

To check the status of the daemon on a Windows server, issue the following command:

```
db2blast_daemon -a status
```

This command assumes that the daemon configuration file is in the same directory as the executable file.

Stop the daemon

To stop the daemon, issue the following command:

```
db2blast_daemon -a stop
```

This command assumes that the daemon configuration file is in the same directory as the executable file.

Remove the daemon

To remove the daemon service from a server, for example to change the server that the daemon runs on, issue the following command:

```
db2blast_daemon -a remove
```

Registering the BLAST wrapper

You must register a wrapper to access the BLAST data sources. Wrappers are used by federated servers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.

Procedure

To register the BLAST wrapper:

Issue the CREATE WRAPPER statement and specify a name for the BLAST wrapper and the name of the wrapper library file. For example to register a wrapper with the name blast_wrapper on the federated server that uses the AIX operating system, issue the following statement:

```
CREATE WRAPPER blast_wrapper LIBRARY 'libdb2lsblast.a';
```

The name of the wrapper library file that you specify depends on the operating system of the federated server. See the list of BLAST wrapper library files for the correct library name to specify in the CREATE WRAPPER statement.

BLAST wrapper library files

The BLAST wrapper library files are added to the federated server when you install WebSphere Federation Server.

When you install IBM WebSphere Federation Server, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2lsblast.a, libdb2lsblastF.a, and libdb2lsblastU.a. The default wrapper library file is libdb2lsblast.a. The other wrapper library files are used with specific wrapper options.

You must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

The default directory paths and default wrapper library file names are listed in the following table.

Table 19. BLAST wrapper library locations and file names

Operating system	Directory path	Wrapper library file name
AIX	/usr/opt/<install_path>/lib32/ /usr/opt/<install_path>/lib64/	libdb2lsblast.a
Linux	/opt/IBM/db2/<install_path>/lib32 /opt/IBM/db2/<install_path>/lib64	libdb2lsblast.so
Solaris	/opt/IBM/db2/<install_path>/lib32 /opt/IBM/db2/<install_path>/lib64	libdb2lsblast.so

Table 19. BLAST wrapper library locations and file names (continued)

Operating system	Directory path	Wrapper library file name
Windows	/opt/IBM/db2/<install_path>/lib32 /opt/IBM/db2/<install_path>/lib64	db2lsblast.dll

<install_path> is the directory path where IBM WebSphere Federation Server is installed on UNIX or Linux.

%DB2PATH% is the environment variable that is used to specify the directory path where WebSphere Federation Server is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

CREATE WRAPPER statement - examples for the BLAST wrapper

Use the CREATE WRAPPER statement to register the BLAST wrapper. The examples show the parameters that are required to access BLAST documents with and without a proxy server.

Example of registering a wrapper

If you are not using a proxy server to access BLAST documents, the statement that you issue to register the wrapper is:

```
CREATE WRAPPER blast_wrapper LIBRARY 'libdb2lsblast.a';
```

blast_wrapper
A name that you assign to the BLAST wrapper. Duplicate wrapper names are not allowed.

LIBRARY *'libdb2lsblast.a'*
The name of the wrapper library file for federated servers that use AIX operating systems.

Example of registering a wrapper for an HTTP proxy server

To register a wrapper and specify an HTTP proxy server, use the following statement:

```
CREATE WRAPPER blast_proxy LIBRARY 'libdb2lsblast.a'
    OPTIONS (PROXY_TYPE 'HTTP',
        PROXY_SERVER_NAME 'proxy.mysite.com',
        PROXY_SERVER_PORT '1081');
```

PROXY_TYPE *'HTTP'*
Specifies the proxy type that is used to access the Internet when the federated server is behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy.mysite.com'*
Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'1081'*
Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Example of registering a wrapper for a SOCKS proxy server

To register a wrapper and specify a SOCKS proxy server without authentication information, use the following statement:

```
CREATE WRAPPER blast_wrapper LIBRARY 'libdb2lsblast.so'
    OPTIONS (PROXY_TYPE 'SOCKS',
        PROXY_SERVER_NAME 'proxy_socks',
        PROXY_SERVER_PORT '1081');
```

LIBRARY *'libdb2lsblast.so'*
The name of the wrapper library file for federated servers that use Linux and Solaris operating systems.

PROXY_TYPE *'SOCKS'*
Specifies the proxy type that is used to access the Internet when the federated server is behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy_socks'*
Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'1081'*
Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Registering the server definition for a BLAST data source

You must register a server definition for each type of BLAST search that you want to run.

About this task

You can register a server definition by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps to register the server definition.

Procedure

To register a server definition for a BLAST data source:

1. Choose the method that you want to use to register the server definition:

Method	Procedure
Using the Control Center	Use the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**. Follow the steps in the wizard.

From the command line	Issue the CREATE SERVER statement. For example:
	`CREATE SERVER server_definition_name`
	`TYPE blast_search_type`
	`VERSION version_number`
	`WRAPPER wrapper_name`
	`OPTIONS (NODE 'node_name');`

   When you register the server definition, you must specify the NODE server option. There are other server options that you can include when you register the server definitions.

2. If you have more than one computer on which the BLAST daemon is installed, you must register the server definitions on each computer.

After the server definition is registered, use the ALTER SERVER statement to add or drop server options.

CREATE SERVER statement - examples for the BLAST wrapper

Use the CREATE SERVER statement to register server definitions for the BLAST wrapper.

The following examples show you how to specify the required parameters, register server definitions for multiple search types, specify additional server options, and specify proxy server information in the server definition.

Required parameters and options

The following example shows you how to register a server definition for the BLAST wrapper by issuing the CREATE SERVER statement:

```
CREATE SERVER server1_blastn TYPE blastn VERSION 2.1.2 WRAPPER blast_wrapper
    OPTIONS(NODE 'big_rs.company.com');
```

server1_blastn
A name that you assign to the BLAST server. Duplicate server definition names are not allowed.

TYPE *blastn*
The type of BLAST search that this associated with the *server1_blastn* server definition. The valid values for TYPE are blastn, blastp, blastx, tblastn, and tblastx.

VERSION *2.1.2*
The version of the BLAST server that you want to access. Set this value to the version of the blastall executable file that you are running on the server where the BLAST daemon is installed.

WRAPPER *blast_wrapper*
The wrapper name that you specified in the CREATE WRAPPER statement.

NODE *'big_rs.company.com'*
Specifies the host name of the system on which the BLAST daemon process is running. This value is case sensitive.

Although the name of the node is specified as an option in the CREATE SERVER statement, it is required for BLAST data sources.

Registering server definitions for multiple search types

To register the server definitions to use the blastn and blastx searches, issue the following CREATE SERVER statements:

```
CREATE SERVER server1_blastn TYPE blastn VERSION 2.1.2 WRAPPER blast_wrapper
    OPTIONS(NODE 'big_rs.company.com');
CREATE SERVER server1_blastx TYPE blastx VERSION 2.1.2 WRAPPER blast_wrapper
    OPTIONS(NODE 'big_rs.company.com');
```

Optional server options

The following example shows additional server options that you can specify when you register a server definition for the BLAST wrapper:

```
CREATE SERVER server1_blastn TYPE blastn VERSION 2.1.2 WRAPPER blast_wrapper
   OPTIONS(NODE 'big_rs.company.com', DAEMON_PORT '3232', USE_CLOB_SEQUENCE 'N');
```

DAEMON_PORT *'3232'*
: Specifies the port number that the daemon will listen on for BLAST job requests. The port number must be the same number that you specified in the DAEMON_PORT option of the daemon configuration file. The default port number is 4007.

USE_CLOB_SEQUENCE *'N'*
: Specifies the data type that the federated server uses for the BlastSeq column. The default value is 'Y'. When set to 'N', you are specifying that the values in the BlastSeq column are VARCHAR data types. If this option is omitted or set to 'N', you are specifying that the values in the BlastSeq column are CLOB data types.

Server definitions when a proxy server is used

You must use the proxy server options in the CREATE SERVER statement if all of the following conditions are true:

- You want to retrieve data using a URI
- The URI used will retrieve data from behind a firewall, through a proxy
- The firewall or proxy used is HTTP or SOCKS

The options that you specify depend on the type of proxy server that you want to access.

Check with your network administrator for information about the type of proxy that you use, and the settings that you should specify in the proxy options.

Registering a server definition for an HTTP proxy server

To register a server definition and specify an HTTP proxy server, use the following statement:

```
CREATE SERVER blast_server_http
   WRAPPER blast_wrapper
   OPTIONS (PROXY_TYPE 'HTTP', PROXY_SERVER_NAME 'proxy_http',
      PROXY_SERVER_PORT '8080');
```

blast_server_http
: A name that you assign to the BLAST server definition. Duplicate server definition names are not allowed.

WRAPPER *blast_wrapper*
: The wrapper name that you specified in the CREATE WRAPPER statement.

PROXY_TYPE *'HTTP'*
: Specifies the proxy type that is used to access the Internet when behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy_http'*
: Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'8080'*
Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Registering a server definition for a SOCKS proxy server

To register a server definition and specify a SOCKS proxy server when authentication information is not required, use the following statement:

```
CREATE SERVER blast_server_socks
    WRAPPER blast_wrapper
    OPTIONS (PROXY_TYPE 'SOCKS', PROXY_SERVER_NAME 'proxy_socks',
        PROXY_SERVER_PORT '1081');
```

blast_server_socks
A name that you assign to the BLAST server definition. Duplicate server definition names are not allowed.

WRAPPER *blast_wrapper*
The wrapper name that you specified in the CREATE WRAPPER statement.

PROXY_TYPE *'SOCKS'*
Specifies the proxy type that is used to access the Internet when behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy_socks'*
Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'1080'*
Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Registering a server definition for a SOCKS proxy server with authentication information

To register a server definition and specify a SOCKS proxy server with authentication information, use the following statement:

```
CREATE SERVER blast_server_socks
    WRAPPER blast_wrapper
    OPTIONS (PROXY_TYPE 'SOCKS', PROXY_SERVER_NAME 'proxy_socks',
        PROXY_SERVER_PORT '1081', PROXY_AUTHID 'Duncan',
        PROXY_PASSWORD 'the1best');
```

PROXY_AUTHID *'Duncan'*
Specifies the user name on the proxy server. This server option is required when the value for the PROXY_TYPE server option is 'SOCKS'.

PROXY_PASSWORD *'the1best'*
Specifies the password on the proxy server that is associated with the user name *'Duncan'*. This server option is required when the value for the PROXY_TYPE server option is 'SOCKS'.

Creating the user mappings for a BLAST data source (optional)

When you attempt to access a BLAST server, the federated server establishes a connection to the BLAST server. If your federated system uses a proxy server to access BLAST data sources, you must create user mappings.

A user mapping is an association between each federated server user ID and password, and the corresponding data source user ID and password.

About this task

There are two methods for specifying user mappings with federated systems. You can use an external repository, such as LDAP, to store the user mappings or you can create the user mappings in the federated database catalog.

If you have an external repository, such as LDAP, to store the user mappings, you do not need to create user mappings. You must specify the DB2_UM_PLUGIN option on the BLAST wrapper. You can specify this option when you register or alter the wrapper.

Procedure

To map a local user ID to the BLAST server user ID and password:

Choose the method that you want to use to register the user mappings:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.

Using the command line	Issue the CREATE USER MAPPING statement. For example:
	`CREATE USER MAPPING FOR` *`local_userID`*
	`SERVER` *`server_definition_name`*
	`OPTIONS (REMOTE_AUTHID '`*`remote_userID`*`',`
	`REMOTE_PASSWORD '`*`remote_password`*`')`
	`PROXY_AUTHID '`*`proxy_server_userID`*`',`
	`PROXY_PASSWORD '`*`proxy_server_password`*`';`

If you specify authentication information for the proxy server when you register a server definition and a user mapping, the values that you specify in when you register the user mapping take precedence.

For example, you have ten people in your organization and you specify authentication information when you register the server definition. You create user mappings for three of the ten people. When the three people access the federated system, the authentication information that you specified when you created the user mappings is used. For the remaining seven people, the authentication information that you specified when you registered the server definition is used.

CREATE USER MAPPING statement - examples for the BLAST wrapper

Use the CREATE USER MAPPING statement to map a federated server user ID to a BLAST server user ID and password.

You can create a user mapping by specifying a proxy server.

Proxy server example

The following example shows how to map a federated server user ID to a BLAST proxy server user ID and password:

To register a server definition and specify a SOCKS proxy server with authentication information, use the following statement:

```
CREATE USER MAPPING FOR groucho SERVER blast_proxy
      OPTIONS (REMOTE_AUTHID 'grouchom',
      REMOTE_PASSWORD 'all4one'
      PROXY_AUTHID 'marxg'
      PROXY_PASSWORD 'them2us');
```

groucho
: Specifies the local user ID that you are mapping to a user ID that is defined at the BLAST proxy server.

SERVER *blast_proxy*
: Specifies the server definition name that you registered in the CREATE SERVER statement for the BLAST server.

REMOTE_AUTHID *'grouchom'*
: Specifies the user ID at the BioRS server to which you are mapping *groucho*. This remote ID must be in a format that is expected by the BLAST server.

REMOTE_PASSWORD *'all4one'*
: Specifies the password that is associated with *'grouchom'*.

PROXY_AUTHID *'marxg'*
: Specifies the user ID on the proxy server. This user mapping option is required when the proxy server requires authentication.

PROXY_PASSWORD *'them2us'*
: Specifies the password on the proxy server that is associated with the user name *'marxg'*. This user mapping option is required when the proxy server requires authentication.

Registering nicknames for BLAST data sources

For each BLAST server definition that you register, you must register a separate nickname for each type of BLAST search that you want to run on a given BLAST-able data source. Use these nicknames when you query the BLAST servers.

For example, if you register server definitions for the blastp and tblastp search types and you want to access the SWISS-PROT data source, you must register two nicknames. One nickname for the SWISS-PROT data source references the server definition for the blastp search type, and the other nickname for the SWISS-PROT data source references the server definition for the tblastp search type.

About this task

You can register nicknames by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps to register the nicknames.

When you register a BLAST nickname, you specify column information for the definition line portion of the data source.

A set of fixed input and output columns are also automatically created with the nickname. The fixed columns are part of the definition for the nickname and are created in the federated database system catalog. You can reference the fixed columns in SQL queries.

Procedure

To register a nickname for a BLAST search type:

1. Determine the definition line information that you must specify when you register the nickname.
2. Determine which columns, including the fixed input columns that you want to specify when you register the nickname.
3. Choose the method that you want to use to register the nickname. Nicknames can be up to 128 characters in length.

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**. Follow the steps in the wizard.

From command line	Issue the CREATE NICKNAME statement. For example:
	`CREATE NICKNAME nickname`
	`(`
	`column_name data_type`
	`   OPTIONS (nickname_column_options),`
	`column_name data_type`
	`   OPTIONS (nickname_column_options),`
	`column_name data_type`
	`   OPTIONS (nickname_column_options)`
	`)`
	`FOR SERVER server_definition_name`
	`OPTIONS (nickname_options);`

Repeat this step for each BLAST search type that you want to create a nickname for.

Definition line parsing

The definition line is like a key for each sequence in the BLAST-able data source and is returned as part of each BLAST hit. The definition line is also called the *defline*.

The value that is returned and parsed by the BLAST wrapper for a definition line will not always be identical to the definition line in the original FASTA file. For example, if there is data in the Accession Number field of a BLAST hit, the definition line that is returned contains the Accession Number data followed by the Definition field data. The wrapper then parses the data that is returned.

Tip: To determine how the wrapper will return and parse the definition line, create a nickname with a single definition line column. Then run a query to see the format that is returned by the wrapper of the definition line for your particular data source.

To include the definition line information in your results table, you must specify the definition line columns in the CREATE NICKNAME statement. Each column that you specify must include the INDEX option and the DELIMITER option, except for the last column. The last column that you define will contain the remainder of the definition line information. You should not specify the DELIMITER option on the last column.

Valid data types for the definition line columns are CLOB, DOUBLE, FLOAT, INTEGER, and VARCHAR.

Fixed input columns for BLAST nicknames

A set of fixed input columns are automatically created when you register a nickname for a BLAST data source.

Input columns are used as parameter-passing predicates in SQL queries. The fixed input columns are specified in the WHERE clause. The input columns pass standard BLAST switches to BLAST. BLAST then runs on the specified data source using these switches. Fixed input columns can also be referenced in the query SELECT list and are returned as part of the results table.

You can override the default data type for a column when you create a nickname. For example, some columns can return a large amount of data, such as the HSP_H_Seq and HSP_Midline columns. To return the first 50 bytes of a column, you can define the column with the data type VARCHAR(50). Only the first 50 bytes will be copied into the output column.

The following table lists the fixed columns that you can use in the WHERE clause.

Table 20. Fixed input columns for BLAST nicknames

Name	Data type	Operators	Description
BlastSeq	VARCHAR (32000) or CLOB	=	Passes the query sequence to the BLAST wrapper.
DB_Genetic_Code	INTEGER	=	DB genetic code for tblastn and tblastx queries uses default = 1.
E_Value	DOUBLE	<	Both an input and an output parameter. As an input parameter, this column indicates to the BLAST wrapper the upper limit of the expected values that should be returned from blastall.
ExtendedGapCost	INTEGER	=	Specifies the value that blastall deducts from the score of an alignment if a gap that was already introduced in either the query sequence or the hit sequence must be extended by one nucleotide or amino acid to allow the length of the alignment to grow.
FilterSequence	CHAR(1)	=	Indicates to blastall whether to perform filtering to remove biologically uninteresting segments from the query sequence. If the search type is BLASTn, the filter used is DUST. Otherwise, filtering is performed by SEG.

Table 20. Fixed input columns for BLAST nicknames (continued)

Name	Data type	Operators	Description
Final_XDropoff	INTEGER	=	The X dropoff value for the final gapped alignment, measured in bits. The value 0.0 invokes the default behavior. For blastn and megablast queries the default is 50 bits. For tblastx queries the default is 0 bits. For all other query types the default is 25 bits [INTEGER data types].
GapCost	INTEGER	=	Specifies the value that blastall deducts from the score of an alignment if a gap must be introduced in either the query sequence or the hit sequence to allow the length of the alignment to grow.
GapAlign	CHAR(1)	=	Indicates to the wrapper whether gapped alignments are permitted in the BLAST output.
Mask_Lower_Case	CHAR(1)	=	Use lowercase filtering with a FASTA sequence.
Matrix	VARCHAR(50)	=	Determines which substitution matrix is used by blastall to determine the degree of similarity between the pairings of amino acids. Only those BLAST search types that compare amino acids to amino acids use this predicate.
NMatchReward	INTEGER	=	Specifies the value that blastall adds to the score of an alignment for each of the pairs of nucleotides in the homologous region that do match. Only those BLAST search types that compare nucleotides to nucleotides use this predicate.
NMisMatchPenalty	INTEGER	=	Specifies the value that blastall deducts from the score of an alignment if one of the pairs of nucleotides in the homologous region does not match. Only those BLAST search types that compare nucleotides to nucleotides use this predicate.
NumberOfAlignments	INTEGER	=	Specifies how many HSP alignments to include in the BLAST output.
Query_Genetic_Code	INTEGER	=	Query genetic code uses default = 1.

Table 20. Fixed input columns for BLAST nicknames (continued)

Name	Data type	Operators	Description
QueryStrands	INTEGER	=	Specifies which strands should be compared when performing a BLASTn search. A value of 1 indicates that the top strand should be used, 2 indicates the bottom strand, and 3 indicates that both strands should be compared.
ThresholdEx	INTEGER	=	Indicates the score threshold below which BLAST does not attempt to extend a hit any further.
WordSize	INTEGER	=	Indicates to blastall the length of the initial hits that blastall initially searches in the database.
XDropoff_Gapped	INTEGER	=	The X dropoff value for gapped alignment, measured in bits. The value 0.0 invokes the default behavior. For blastn queries the default is 30 bits. For megablast queries the default is 20 bits. For all other query types the default is 15 bits [INTEGER data types].
XDropoff_Ungapped	DOUBLE	=	The X dropoff value for ungapped extenstion, measured in bits. The value 0.0 invokes the default behavior. For blastn queries the default is 20 bits. For megablast queries the default is 10 bits. For all other query types the default is 7 bits [REAL data types].

BLAST search types and switches for fixed input columns

You need to know the BLAST search types and switches that each of the BLAST fixed input columns can support.

The supported BLAST search types and switches for each fixed input column are listed in the following table.

Table 21. BLAST search types and switches supported by the input fixed columns

Fixed input column	BLAST search types	BLAST switch	Required	Default
BlastSeq	n, p, x, tn, tx	-i	Yes	N/A
DB_Genetic_Code	tn, tx	-D	No	1
E_Value	n, p, x, tn, tx	-e	No	10
ExtendedGapCost	n, p, x, tn, tx	-E	No	1
FilterSequence	n, p, x, tn, tx	-F	No	T
Final_XDropoff	n, p, x, tn, tx	-Z	No	0
GapAlign	n, p, x, tn, tx	-g	No	T
GapCost	n, p, x, tn, tx	-G	No	11
Mask_Lower_Case	n, p, x, tn, tx	-U	No	F

Table 21. BLAST search types and switches supported by the input fixed columns (continued)

Fixed input column	BLAST search types	BLAST switch	Required	Default
Matrix	p, x, tn, tx	-n	No	BLOSUM62
NMisMatchPenalty	n	-q	No	–3
NMatchReward	n	-r	No	1
NumberOfAlignments	n, p, x, tn, tx	-b	No	250
Query_Genetic_Code	n, p, x, tn, tx	-Q	No	1
QueryStrands	n	-S	No	3
ThresholdEx	n, p, x, tn, tx	-f	No	0
WordSize (for Blastn, a value less than 7 is invalid)	n, p, x, tn, tx	-W	No	11 –BLASTn 3 –BLASTp
XDropoff_Ungapped	n, p, x, tn, tx	-y	No	0.0
XDropoff_Gapped	n, p, x, tn, tx	-X	No	0

Fixed output columns for BLAST nicknames

A set of fixed output columns are automatically created when you register a nickname for a BLAST data source.

The following table lists the fixed columns that you can use in the WHERE clause.

Table 22. Fixed output columns for BLAST nicknames

Name	Data type	Description
E_value	DOUBLE	Both an input and an output parameter. As an output parameter, this column provides the computed score for an HSP as reported in the BLAST results.
HIT_NUM	INTEGER	The hit number as reported in the BLAST results, starting with 1.
HSP_ALIGNMENT_LENGTH	INTEGER	The length of the HSP alignment.
HSP_IDENTITY	INTEGER	The percent identity of the alignment defined as the number of identities divided by the alignment length.
HSP_Info	VARCHAR(100)	The information string for the given HSP, as reported by BLAST. This string contains information about the number of nucleotides or amino acids that matched between the query sequence and the hit sequence.
HSP_GAPS	INTEGER	The percent gaps in the alignment defined as the number of gaps divided by the alignment length.
HSP_H_Start	INTEGER	The numeric position of the first homologous nucleotide or amino acid on the hit sequence.

Table 22. Fixed output columns for BLAST nicknames (continued)

Name	Data type	Description
HSP_H_End	INTEGER	The numeric position of the last homologous nucleotide or amino acid on the hit sequence.
HSP_H_Seq	VARCHAR(32000)	The segment of the hit sequence beginning at HSP_H_Start and ending at HSP_H_End. You can override the default data type for this column and specify CLOB, with a maximum length of 5 megabytes.
HSP_HIT_FRAME	INTEGER	The reading frame of the alignment in the hit sequence. Only available for blastx, tblastn, and tblastx type servers.
HSP_Midline	VARCHAR(32000)	The string output by BLAST that indicates the degree of homology between the amino acids or nucleotides at each position in the homologous regions of the query and hit sequences. You can override the default data type for this column and specify CLOB, with a maximum length of 5 megabytes.
HSP_NUM	INTEGER	The HSP number as reported in the BLAST results, starting with 1.
HSP_Q_Start	INTEGER	The numeric position of the first homologous nucleotide or amino acid on the query sequence.
HSP_Q_End	INTEGER	The numeric position of the last homologous nucleotide or amino acid on the query sequence.
HSP_Q_Seq	VARCHAR(32000)	The segment of the query sequence beginning at HSP_Q_Start and ending at HSP_Q_End. You can override the default data type for this column and specify CLOB, with a maximum length of 5 megabytes.
HSP_POSITIVE	INTEGER	The percent positives of the alignment defined as the number of positives divided by the alignment length.
HSP_QUERY_FRAME	INTEGER	The reading frame of the alignment in the query sequence. Only available for blastx, tblastn, and tblastx type servers.
HSP_Score	DOUBLE	Use lowercase filtering with a FASTA sequence.

Table 22. Fixed output columns for BLAST nicknames (continued)

Name	Data type	Description
Length	INTEGER	The length of the hit sequence as reported in the BLAST results.
Score	DOUBLE	The computed score for an HSP as reported in the BLAST results.

CREATE NICKNAME statement - example for the BLAST wrapper

Use the CREATE NICKNAME statement to register a nickname for a BLAST data source that you want to access.

The CREATE NICKNAME statement for a genbank nickname

The following example defines the nickname *genbank*. This example assumes that the definition field in a BLAST result set contains the following information:

```
>276342 15:8924 PMON5426
```

276342 The accession field of the BLAST result.

15:8924 PMON5426
The definition field in a BLAST result set that contains an organism number (15) followed by an experiment number (8924) and then a unique identifier (PMON5426).

With this information, you can create the following nickname:

```
CREATE NICKNAME genbank
     (
     accession_num INTEGER  OPTIONS(INDEX '1', DELIMITER ' '),
     organism_num  INTEGER  OPTIONS(INDEX '2', DELIMITER ':'),
     experiment_num INTEGER OPTIONS(INDEX '3', DELIMITER ' '),
     unique_id  VARECHAR(10)  OPTIONS(INDEX '4'))
     FOR SERVER server_blastn
        OPTIONS(DATASOURCE 'genbank', TIMEOUT '300');
```

In this example, the columns use the INDEX options to refer to portions of the definition field:

- The INDEX 1 refers to the accession field 276342
- The INDEX 2 refers to the organism number 15
- The INDEX 3 refers to the experiment number 8924
- The INDEX 4 refers to the unique identifier PMON5426

After you submit the CREATE NICKNAME statement, you can use the *genbank* nickname to query your federated system. You can also join the *genbank* nickname with other nicknames and tables in your federated system.

Querying BLAST data sources

After you create the BLAST nicknames, you can query BLAST data sources. The choices that you make when you create the queries impact the performance of the BLAST wrapper when it accesses those data sources.

Setting up TurboBlast to work with the BLAST wrapper

TurboBlast is a proprietary version of the BLAST program. To use TurboBlast with the BLAST wrapper, you must complete several additional configuration steps.

Restrictions

TurboBlast does not support certain blastall command options. For example, the gapped alignment option `-g F` is not supported. If you specify the `F` option for GapAlign column in your BLAST nickname, TurboBlast generates an error. For a complete list of unsupported options, refer to the *TurboBlast 2.0 User Guide*.

About this task

The BLAST wrapper and TurboBlast support the AIX, Linux, Solaris and Windows operating systems. However, the BLAST daemon is not available on Windows operating systems. The daemon will work with TurboBlast on Windows operating systems when the BLAST daemon is available on those operating systems.

Procedure

To set up TurboBlast to work with the BLAST wrapper:

1. Install and configure the BLAST wrapper.
2. Run a query on a BLAST-able database to test your setup.
3. Install and configure TurboBlast according to the TurboBlast 2.0 Installation and Reference Guide. You can install and set up the TurboBlast system in various ways. To allow the BLAST wrapper to work with TurboBlast, you need to install and set up the TurboBlast Client on the computer on which you have your BLAST daemon. The BLAST daemon can invoke the tblastall program. Be sure to test the TurboBlast system after you have installed and configured TurboBlast. Follow the instructions in the *TurboBlast 2.0 Installation and Reference Guide*.
4. Edit the BLAST_DAEMON.config file:
   a. Add the BLASTALL_PATH parameter and specify the complete path of the executable file for the tblastall program.

      For example:

      ```
      BLASTALL_PATH=/home/blasttst/turboblast/TBlast-2.1/tblastall
      ```

   b. Add the BLAST-able database specification entry and specify the BLAST-able database name that you used to upload your BLAST-able database to TurboBlast. The database names display when you enter the listdatabase -l command under TurboBlast. This TurboBlast database name should be used instead of the path to the BLAST-able data source. For example:

      ```
      genbank=genbank_database_name_in_TurboBlast
      ```

5. Restart the BLAST daemon.

   The BLAST daemon invokes the tblastall program instead of the blastall program to search the BLAST-able databases.

   The log files that are related to the tblastall program are written to the directory that is specified in the DAEMON_LOGFILE_DIR parameter in the BLAST_DEAMON.config file. Also check the STDERR.log and STDOUT.log that are produced by the BLAST daemon in the same directory.

Construct BLAST SQL queries

How you construct your BLAST queries impacts the quality of the results that are returned. Learn how to use predicates and custom functions in your BLAST queries.

Predicates on input and output columns

Predicates on input columns are used to pass standard BLAST switches to the blastall executable program.

Predicates on the output columns are processed by the federated server.

The BlastSeq input column predicate

To be valid, every query passed to the BLAST wrapper must contain at least a predicate with the `BlastSeq` input column. All other predicates are optional.

Wrappers that require predicates, such as the BlastSeq fixed input column, cannot process queries that result in a left outer join on the required predicate. For example, the following query returns an SQL0901N error:

```
SELECT n1.Score FROM blastNN1 n1
    LEFT OUTER JOIN myseqs n2 ON N1.BlastSeq = n2.seq
```

Function templates for CLOB columns

If the BlastSeq column for a BLAST nickname is defined as a CLOB data type, you must create sequence match function templates before you can query BLAST data sources. The SQL statements that you issue on the federated database to create the function templates are:

```
CREATE FUNCTION lblast.sequence_match (CLOB, CLOB)
    RETURNS INTEGER AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION;

CREATE FUNCTION lblast.sequence_match (CLOB, VARCHAR(1))
    RETURNS INTEGER AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION;

CREATE FUNCTION lblast.sequence_match (VARCHAR(1), CLOB)
    RETURNS INTEGER AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION;

CREATE FUNCTION lblast.sequence_match (VARCHAR(1), VARCHAR(1))
    RETURNS INTEGER AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION;
```

Using the SUBSTR scalar function in queries

You can use the SUBSTR scalar function in the query predicate. The SUBSTR function returns part of a string.

For example:

```
SELECT BlastSeq FROM t1, t2
    WHERE BlastSeq = SUBSTR (t2.sequence, 15, 300);
```

You can use the SUBSTR function in a SEQUENCE_MATCH function for CLOB data types.

For example:

```
SELECT BlastSeq FROM t1, t2
    WHERE LSBlast.SEQUENCE_MATCH (BlastSeq,SUBSTR(t2.sequence, 15, 300))=1;
```

Important: Using the SUBSTR() function does not produce the same results as the -L option with the blastall program on the command line. Use the SUBSEQUENCE custom function instead of the SUBSTR() function when you want to produce the same results as -L option.

Function templates for the SUBSEQUENCE custom function

To produce the same results as the -L option on the command line with the blastall program, use the SUBSEQUENCE custom function. You must create subsequence function templates before you can query BLAST data sources. The SQL statements that you issue on the federated database to create the function templates are:

```
CREATE FUNCTION lsblast.subsequence (CLOB, INTEGER, INTEGER)
   RETURNS VARCHAR(1) AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION;
CREATE FUNCTION lsblast.subsequence (VARCHAR (1), INTEGER, INTEGER)
   RETURNS VARCHAR(1) AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION;
```

For more details about the -L option, refer to the blastall documentation.

BLAST data source – example queries

These examples show you how to structure your queries to optimize system performance. Learn how to use the custom functions and wildcards in your queries, access specific BioRS data type columns, and to form an equijoin between parent and child nicknames.

Use the input predicates in the WHERE clause of your BLAST SQL queries.

In these queries, the name that is used for each nickname indicates the type of BLAST search and the data source. Some of the queries make use of other hypothetical data sources so that the examples show the behavior of the BLAST wrapper when the BLAST data is joined with data from other sources.

Input predicates example

This query searches the GenBank database for homologues, using a partial sequence and shows three input predicates: `BlastSeq`, `GapCost`, and `NMisMatchPenalty`:

```
SELECT * FROM genbank b
WHERE
BlastSeq = 'GTCCAGCC...' AND
GapCost = -10 AND
NMisMatchPenalty = -4;
```

BLASTn search that returns all available columns

When the following SELECT statement is run, the wrapper performs a BLASTn search of GenBank using the indicated sequence. The wrapper returns all of the available columns, including both the input parameter columns and the BLAST result columns.

```
SELECT * FROM blastn_genbank
    WHERE
    BlastSeq =  'caacccctccagccgagttgtcaatggcgaggaagctgttccccac';
```

BLASTn search of GenBank that passes parameters to the daemon

When the following SELECT statement is run, the wrapper performs a BLASTn search of GenBank using the indicated sequence. In addition, the wrapper passes the `GapCost` and the `NmisMatchPenalty` parameters to the daemon. These parameters are then passed to the blastall command line. The wrapper returns all of the available columns, including both the input parameter columns and the BLAST result columns.

```
SELECT * FROM blastn_genbank
    WHERE
    BlastSeq =  'caacccctccagccgagttgtcaatggcgaggaagctgttccccac'
    AND GapCost = 8
    AND NmisMatchPenalty = -4;
```

BLASTp search of SWISS-PROT that is based on the number of sequences returned

When the following SELECT statement is run, the wrapper performs zero or more BLASTp searches of SWISS-PROT, depending on the number of sequences that are returned from a hypothetical protein sequence database. This statement will be separated into two queries by the federated database. One query will perform a BLASTp search that is run for each row that is returned from the hypothetical protein database. The wrapper returns all of the available columns, including both the input parameter columns and the BLAST result columns.

```
SELECT blp.* FROM blastp_swissprot blp, protein_db prdb
    WHERE prdb.keyword = 'malic enzyme'
    AND blp.BlastSeq = prdb.sequence;
```

BLASTx search of SWISS-PROT that is based on an input sequence

When the following SELECT statement is run, the wrapper performs a BLASTx search of SWISS-PROT using the indicated sequence. In this example, blastall translates the input sequence in all six reading frames and perform the homology search using each of the six newly created protein sequences. The HSPs in the results will contain amino acid-amino acid alignments, rather than nucleotide-nucleotide alignments. The supplied parameter is passed to the daemon and then to blastall command line. The wrapper returns only those columns that are specifically requested in the query.

```
SELECT Score, E_Value, HSP_Info, HSP_Q_Seq, HSP_H_Seq, HSP_Midline
    FROM blastx_swissprot
    WHERE BlastSeq = 'gagttgtcaatggcgagg'
    AND GapCost = 8;
```

tBLASTx search of GenBank that is based on the number of sequences returned

When the following SELECT statement is run, the wrapper performs zero or more tBLASTx searches of GenBank, depending on the number of sequences returned from a hypothetical gene expression database. The statement is broken into two separate queries by the federated database. One query will perform a tBLASTx search and is run for each row that is returned from the hypothetical gene expression database. In this case, blastall will translate the input sequence and all of the sequences in GenBank in all six reading frames. Additionally, blastall will perform the homology search by using each of the six newly created query protein sequences and all of the newly created database protein sequences. The HSPs in the results will contain amino acid-amino acid alignments, rather than nucleotide-nucleotide alignments. The supplied parameters are passed to the daemon and then to blastall command line. The wrapper returns only those columns that are specifically requested in the query.

```
SELECT tblx.Score, tblx.E_Value, tblx.HSP_Info tblx.HSP_Q_Seq,
   HSP_H_Seq, HSP_Midline
    FROM tblastx_genbank tblx, gen_exp_database gedb
    WHERE tblx.BlastSeq = gedb.sequence
    AND gedb.organism = 'interesting organism'
    AND GapCost = 8
    AND FilterSequence = 'F';
```

Guidelines for optimizing BLAST query performance

The location of the BLAST daemon can impact query performance.

To improve query performance, you should use a separate BLAST server for the BLAST daemon. The federated server and the BLAST server should be on separate hardware. The BLAST daemon should reside on the BLAST server.

Chapter 6. Business application data sources

Configuring access to business application data sources

You can integrate the data that is in the business application databases with information from other sources by using a federated system. To configure the federated server, you must register a wrapper, a server definition, and nicknames for the business application databases.

Before you begin

- IBM WebSphere Federation Server must be installed on a server that acts as the federated server.
- A federated database that uses a 32-bit DB2 database instance must exist on the federated server.

Procedure

To configure access to business application data sources:

1. Configure the WebSphere Business Integration Adapters.
2. Register the WebSphere Business Integration wrapper (DB2 command line).
3. Register the server definition for business application data sources (DB2 command line).
4. Register nicknames for business application data sources:
   - Register nicknames for business application data sources (DB2 Control Center)
   - Register nicknames for business application data sources (DB2 command line)

WebSphere Business Integration wrapper

The WebSphere Business Integration wrapper provides an SQL interface to business applications, such as those produced by SAP, Siebel, and PeopleSoft.

By using the WebSphere Business Integration wrapper, you can use the federated systems functions to join business data from business applications with data on other federated data sources. The WebSphere Business Integration wrapper extracts business object definitions into a hierarchy of nicknames. The wrapper supports result sets that return multiple business objects.

The WebSphere Business Integration wrapper is a read-only wrapper and uses the WebSphere Business Integration Adapters to access business applications. To return result sets that include multiple business objects, the WebSphere Business Integration wrapper requires the WebSphere Business Integration Adapters and Framework, version 2.6. The WebSphere Business Integration wrapper can be used with earlier versions of the WebSphere Business Integration Adapters and Framework, subject to the limitations that are described in “Configuring the WebSphere Business Integration Adapters” on page 99.

Figure 7 on page 98 shows the relationship between the WebSphere Business Integration wrapper and the adapters in the DB2 database environment, and to other federated data sources.

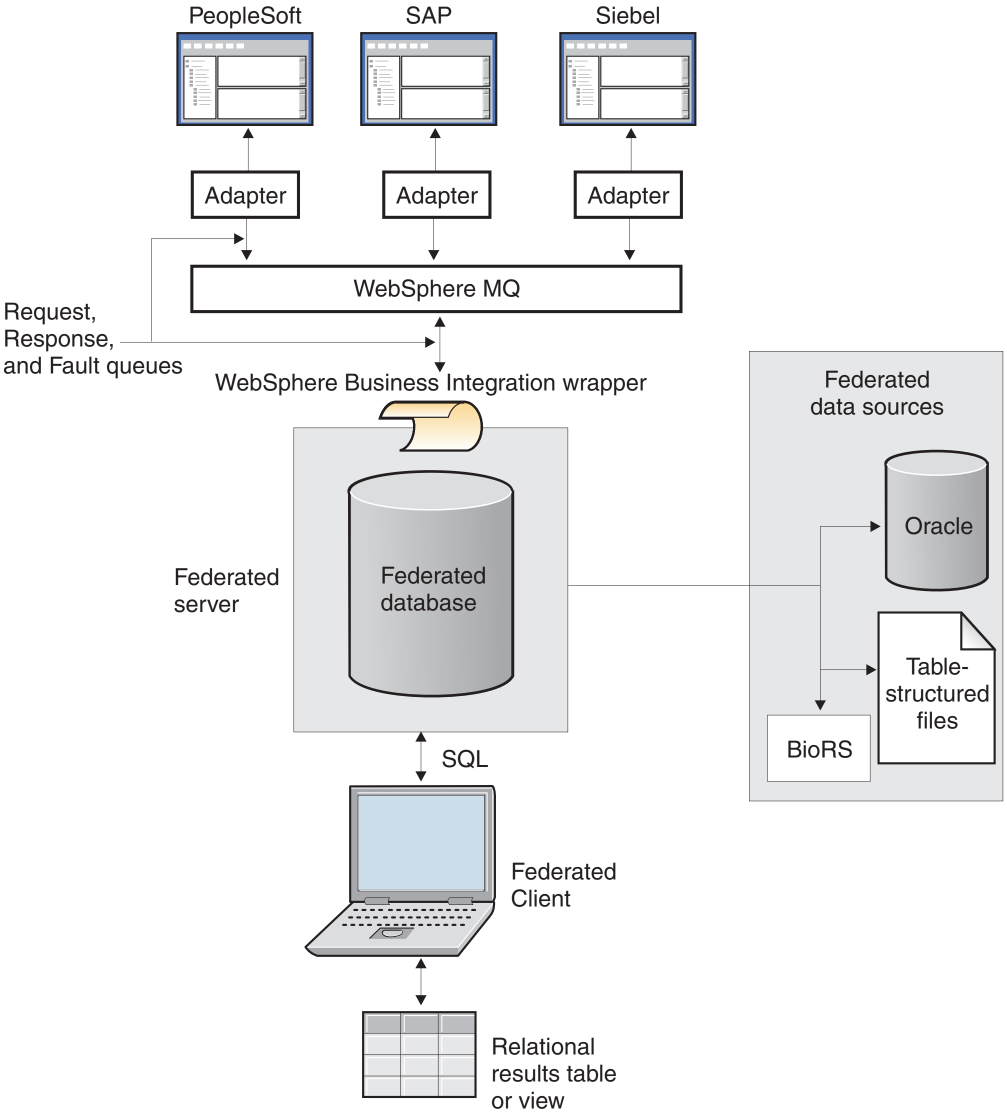


Figure 7. WebSphere Business Integration wrapper in the DB2® database environment

The following steps describe the process for accessing business application data in a federated system:

1. A user sends a query to the federated server to access a nickname that maps to a data source such as a Siebel application.
2. The wrapper transforms the query into a business object.
3. The wrapper places the business object on a WebSphere MQ message request queue.
4. The WebSphere Business Integration adapter for the particular application reads the business object, which is the request, from the message request queue.
5. The WebSphere Business Integration adapter works with the business application to prepare a response business object.
6. The WebSphere Business Integration adapter puts the response business object on the message response queue.
7. The wrapper reads the response business object from the response queue.

8. The wrapper extracts the response business object into a result set based on the relational schema that is defined with the nickname definition.

Configuring the WebSphere Business Integration Adapters

For each business application that you want to access with SQL statements by using the federated wrapper functions, you must install and configure a WebSphere Business Integration Adapter. Each adapter maps to a federated server definition.

Before you begin

- See the IBM WebSphere Business Integration Information Center for installation information for each adapter.
- See the WebSphere Business Integration Adapters documentation for help with a specific business application.
- Install all of the most current fix packs for the particular adapter that you want to use. You can get the pertinent support information for the Adapters from the WebSphere Business Integration Adapters Support site.
- See theIBM WebSphere MQ Information Center for information about configuring the message queues.
- See the installation information in the IBM WebSphere Business Integration Information Center for information about the adapters and the configuration properties.
- See the documentation for the adapter that you are configuring for information about how to configure and use an Object Discovery Agent tool.

Supported adapters are as follows:

- WebSphere Business Integration Adapter for mySAP.com
- WebSphere Business Integration Adapter for Siebel eBusiness Applications
- WebSphere Business Integration Adapter for PeopleSoft

Procedure

To configure a WebSphere Business Integration Adapter:

1. Configure the Object Discovery Agent tool and the Business Object Designer tool, and build the business object definitions.
2. Define a configuration file by using the Connector Configurator tool from the WebSphere Business Integration Adapter interface. The configuration file must contain the following information:
   a. The business objects that the adapter supports.
   b. The configuration properties for the adapter.
3. Define the WebSphere MQ message queues that the wrapper requires: request_queue, response_queue, and fault_queue.
4. Define the additional message queues that are required by the adapter:
5. Define the WebSphere MQ user authorization by using either of the following methods:
   - Define the DB2 instance owner ID as part of the MQManager group.
   - Ensure that the MQManager administrator sets the MCAUSER value while creating the ServerConnection channel. The value of MCAUSER must be a user ID that is part of the MQManager group or the Administrator group.

The WebSphere Business Integration Adapters, version 2.6, support result sets for business applications. The adapter configuration file provides two properties to enable result sets for business applications:

ResultsSetEnabled
: Set this property to true to enable result sets. The default value is false.

ResultsSetSize
: The WebSphere Business Integration wrapper sets the maximum size of the result set at run time as a property in the JMS message header. Do not change this value. The default value is 0.

Configure these properties in the adapter configuration file before you start the adapter.

Business object definitions

A business object definition is a template from which the WebSphere® Business Integration Adapter creates an instance of a business object. A business object definition represents a business application data entity.

A business object is an instance of a business object definition and can be either a flat or hierarchical structure. A flat business object contains only simple attributes. A hierarchical business object contains one or more complex attributes. A repository of business object definitions exists within each WebSphere Business Integration Adapter for the particular application that is supported.

The following example shows a flat business object:

```
Customer
     Gomez
     Juanita
     Apt 2C
     123 Main Street
     Big City
     California
     91234
     888
     1111111
```

The following example shows a hierarchical business object with some complex attributes:

```
Contact (Parent)
     ID
     Customer ID
     Date
     Text
     Authorization
     Line items (there are 0 or more Line item elements)
          (Child elements)
          Business object 1
          Business object 2
          Business object 3
```

Business object definitions must be generated by using the Object Discovery Agent tool that is packaged with each WebSphere Business Integration Adapter. The Object Discovery Agent tool generates an XML schema definition file for a business object definition. The Object Discovery Agent tool might generate multiple schema files if the business object that is being defined has a hierarchical structure.

The XML schema definition is a file with file type XSD in a directory that is specified in the WebSphere Business Integration configuration. You must generate the business object definition before you create the nicknames for the WebSphere Business Integration wrapper.

To create nicknames, use the XSD file that the Object Discovery Agent tool creates. Nicknames provide a relational schema representation of the business object definition. The WebSphere Business Integration wrapper maps a hierarchical business object into a hierarchy of relational nicknames. For example, each child business object of cardinality 'n' is mapped to a separate nickname that is linked to the nickname of the parent business object with a foreign key constraint.

The WebSphere Business Integration business objects that are accessible to the federated server map to the specific application entities in the following table:

Table 23. Business objects and the related application entities

Business objects	Application entities
Siebel	Business Component
PeopleSoft	Component Interface
SAP	BAPI

Configuration properties for business object definitions

There are standard configuration properties and application-specific configuration properties for business object definitions.

The following configuration properties must be in the configuration file when you define business object definitions:

Standard configuration properties

You must customize some property values to use the adapter with a federated system. Some specific properties to configure are included in the following list:

- Specify the value of the integration broker as WMQI.
- Specify the location of the metadata repository that is owned by the adapter. The XML schema definition files, which contain the business object definitions, are saved in this location.
- Specify the type of delivery transport as WMQI-MQ.
- Specify the name of the queue manager that manages the queues that are used by the adapter.
- Specify the names of the queues that are required to run the adapter.

Application-specific configuration properties

These properties specify values for a particular application-specific component. The values that you provide help to establish a session with the application. The properties also direct the processing behavior for the application-specific components.

WebSphere MQ message queues for the WebSphere Business Integration wrapper and the adapter

WebSphere MQ is the messaging and transport layer between the adapter and the wrappers. You must define the queues that are required for the WebSphere Business Integration wrapper and adapter.

You must define eight queues to use the WebSphere Business Integration wrapper. The wrapper uses three of these static queues to exchange messages, including data objects and error messages, between the adapter and the wrapper:

request_queue
Delivers request messages from the federated server to the adapter.

response_queue
Delivers response messages from the adapter to the federated server.

fault_queue
Delivers fault messages from the adapter to the federated server. The adapter places a message on this queue when it is unable to place the message on the reply-to queue.

The following diagram shows the flow of messages on message queues between Siebel applications and the federated server.

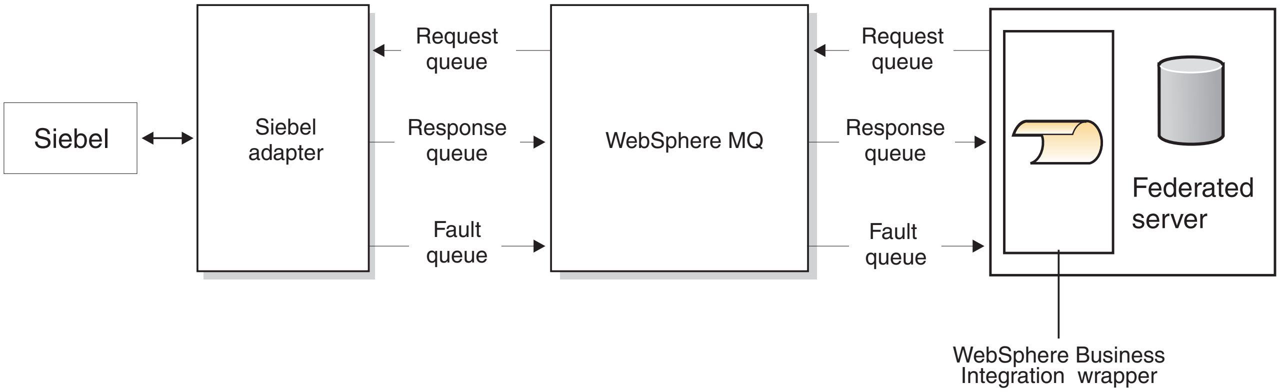


Figure 8. The topology of the WebSphere message queues that transport information between the Siebel business applications and the federated server

The WebSphere Business Integration adapters require five additional queues that are used when the adapter is used with a WebSphere MQ Integrator broker. You must create and configure these additional message queues so that the adapter can be started:

- AdminInQueue
- AdminOutQueue
- SynchronousRequestQueue
- SynchronousResponseQueue
- DeliveryQueue

If result sets are enabled for the WebSphere Business Integration wrapper, WebSphere MQ log files can be very large. Configure more log files or larger log files if necessary.

Adding business application data sources to a federated system

Registering the WebSphere Business Integration wrapper (DB2 command line)

You must register a wrapper to access WebSphere Business Integration data sources. Wrappers are used by federated servers to communicate with and retrieve data from data sources.

Before you begin

1. Install and configure the appropriate adapter.
2. Install and configure WebSphere MQ, version 5.3 (CSD level 5) or later.
3. Create the WebSphere MQ message queues.
4. If the WebSphere MQ manager is not installed on the same system as the federated server, install the WebSphere MQ, version 5.3 (CSD level 5) client on the same system on which you installed a federated server instance.

About this task

You can register the wrapper by using the DB2 Control Center or the DB2 command line. The DB2 Control Center includes a wizard to guide you through the steps that are required to register the wrapper.

Procedure

To register a wrapper:

Issue the CREATE WRAPPER statement with the name of the wrapper and the name of the wrapper library file. For example, to register a wrapper with the name wbi_wrapper on the federated server that uses Windows, issue the following statement:

```
CREATE WRAPPER wbi_wrapper LIBRARY 'db2wbi.dll';
```

The name of the wrapper library file that you specify depends on the operating system of the federated server.

WebSphere Business Integration wrapper library files

The WebSphere Business Integration wrapper library files are added to the federated server when you install WebSphere Federation Server.

When you install WebSphere Federation Server, library files are added to the directory path listed in the table. For example, if the federated server is running on AIX, the wrapper library files added to the directory path are libdb2wbi.a, libdb2wbiF.a, and libdb2wbiU.a.

You must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name. When you register a wrapper, specify only the library file name that is listed in the table.

Table 24. WebSphere Business Integration wrapper library locations and file names

Operating system	Default directory path	Default wrapper library file
AIX	/usr/opt/*install_path*/lib/	libdb2wbi.a
Linux	/opt/IBM/db2/*install_path*/lib	libdb2wbi.so
Solaris	/opt/IBM/db2/*install_path*/lib	libdb2wbi.so
Windows	%DB2PATH%\bin	db2wbi.dll

- *install_path* is the directory path where WebSphere Federation Server is installed on UNIX or Linux.
- %DB2PATH% is the is the environment variable that is used to specify the directory path where WebSphere Federation Server is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

Registering the server definition for business application data sources (DB2 command line)

You can register a server definition from the DB2 command line.

Restrictions

The server options must include the WebSphere MQ Series queue definitions:

- Specify the MQ_SVRCONN_CHANNELNAME option only if you specify the MQ_CONN_NAME option.
- Do not drop the MQ_CONN_NAME option until you drop the MQ_SVRCONN_CHANNELNAME option.
- If MQ_CONN_NAME is not specified, the federated system uses the value of the MQSERVER environment variable.

The WebSphere Business Integration wrapper requires the WebSphere Business Integration Adapters and Framework, version 2.6 for result set support. The wrapper supports the value 2.6 for the VERSION parameter that is specified in the CREATE SERVER statement. To enable result sets, you must specify version 2.6 for the VERSION parameter in the CREATE SERVER statement.

About this task

After you register the wrapper, you must register a corresponding server.

A server option enables result sets. Set the RESULTSET_ENABLED option to Yes to enable result sets. Set the RESULTSET_ENABLED option to No to disable result sets. The default value for this server option is No. The VERSION parameter value must be set to 2.6 or later to enable result sets. Before you set the RESULTSET_ENABLED option to Yes, ensure that the ResultsSetEnabled property in the adapter configuration file is set to true. If the values do not match, an error results.

Procedure

To register the server definition for a business application:

Issue the CREATE SERVER statement. For example, to register a server definition for the Siebel business applications:

```
CREATE SERVER siebel_server
  VERSION 2.6
  WRAPPER wbi_wrapper
  OPTIONS ( App_Type 'siebel',
    Request_Queue 'myqueue3',
    Response_Queue 'myqueue4',
    Fault_Queue 'myqueue5',
    MQ_Manager 'mymq'
    MQ_REPONSE_TIMEOUT '55000',
    MQ_CONN_NAME '9.30.76.151(1420)',
    MQ_SVRCONN_CHANNELNAME 'SYSTEM.DEF.SVRCONN'
    RESULTSET_ENABLED 'Y'
)
```

In the example, the business application is a Siebel application, which is identified with the APP_TYPE option. The valid values are SIEBEL, PSOFT, and SAP. The VERSION option represents the version of the WebSphere Business Integration Adapters that you are using. Valid values are 2.3, 2.4, and 2.6, but you must specify version 2.6 to enable result sets. The default value for MQ_RESPONSE_TIMEOUT is set to 50000 milliseconds. A value of -1 specifies that there is no timeout limit.

Registering nicknames for business application data sources

You can register the nickname by using the DB2 Control Center or the DB2 command line. The DB2 Control Center includes a wizard to guide you through the steps that are required to register the nickname.

You must create nicknames that correspond to the structural hierarchy of your business object definition. Parent nicknames contain at least one child nickname. Child nicknames correspond to the elements that contain a cardinality greater than 1 that are nested within the element for the parent nickname. The DB2 Control Center generates unique nicknames by grouping part names or element names with the column name from the XML schema document.

Procedure

To register nicknames for business application data sources:

Select one of the following methods:
- "Registering nicknames for business application data sources (DB2 Control Center)"
- "Registering nicknames for business application data sources (DB2 command line)" on page 107

Registering nicknames for business application data sources (DB2 Control Center)

For each WebSphere Business Integration server definition that you register, you must register a nickname for each data source that you want to access. Use these nicknames instead of the names of the data sources when you query the Web services data sources.

Before you begin

If result sets are not enabled, identify the required input columns before you generate nickname definitions by using the DB2 Control Center.

Procedure

To register nicknames for business application data sources from the DB2 control center:
1. Expand the **Federated Database Objects** folder.
2. Expand the wrapper folder for which you want to register nicknames.
3. Expand the **Server Definitions** folder.
4. Expand the server folder for which you want to register nicknames.
5. Right-click the **Nicknames** folder and select **Create**.
6. In the Create Nicknames window, click **Discover** to define search criteria to help you select objects from the data source.
7. Specify the XML schema definition file that contains the definition of the business objects that you want federated system users to access.

8. Click **OK** to create the nickname according to the selected XML schema definition file.

The DB2 Control Center extracts the schema file into multiple CREATE NICKNAME statements, with the appropriate parent-child relationship definitions. The nicknames that are created represent the business object hierarchy that is defined in the XML schema definition file.

Identifying the required input columns in the business object before generating nickname definitions in the DB2 control center

If you want to retrieve only one business object from a query, you need to ensure that the wrapper sends all the required predicates. If result sets are enabled, you can retrieve result sets of multiple business objects.

Before you begin

If you are using a WebSphere Business Integration Adapter, version 2.4 or earlier, you must identify the required input columns in the business object.

If you are using a WebSphere Business Integration Adapter, version 2.6 or later and result sets are not enabled, you can identify the required input columns.

Because the WebSphere Business Integration wrapper supports result sets of multiple business objects, you do not need to identify the required input columns in the business objects for PeopleSoft and Siebel applications. However, the WebSphere Business Integration Adapters support only equality predicates in all queries on input columns. You can deselect any column that should not be an input column so that you can query by using a nonequality predicate.

Optionally, you can identify required input columns to limit the response to a query to only a single business object.

Procedure

To identify the required input columns in the business object:

1. Identify the columns in the SAP, Siebel, or PeopleSoft application repository that represent a unique key for the application entity being mapped.

Application	Tool
SAP	Use the SAP Business Object Repository to identify the required input parameters for the BAPI that is being mapped to a WebSphere Business Integration business object definition by the WebSphere Business Integration Object Discovery Agent tool.

Application	Tool
Siebel	Use one of the following methods: • The Siebel application has a unique identifier column that is associated with each business component and generates hexadecimal values for this column for each instance of the business component. This identifier column exists at the highest level of the business component hierarchy and is already flagged by the isKey=″true″ specification in the appSpecificInfo section of the XML annotation for the element in the generated XSD file. If result sets are enabled, every column in the root level can be marked as an input column. • You can use Siebel tools to identify the database columns that represent a composite unique key for the business component that is being mapped. These columns must all be at the highest or root level of the business object hierarchy.
PeopleSoft	Use the Application Designer tool to identify the getKey columns in the component interface for the highest level of the hierarchy that is being mapped to a WebSphere Business Integration business object definition. If result sets are enabled, findKey columns can be used as input columns.

2. Edit the XSD files that are generated for the business object definition by the WebSphere Business Integration Object Discovery Agent tool to flag the required input columns.
3. Generate the nickname DDL statement for the business object definition from the DB2 Control Center.

Registering nicknames for business application data sources (DB2 command line)

For each WebSphere Business Integration server definition that you register, you must register a nickname for each data source that you want to access. Use these nicknames instead of the names of the data sources when you query the Web services data sources.

Procedure

To register nicknames for business application data sources from the DB2 command line, issue a CREATE NICKNAME statement:

For example, to register a nickname for a Siebel business object definition that is called sieb_ssa_Contact_Contact, issue the following statement:

```
CREATE NICKNAME sieb_ssa_Contact_Contact_NN(
 Id VARCHAR(15)  OPTIONS(XPATH './ns1:Id/text()',
                          TEMPLATE '<ns1:Id>&column</ns1:Id>'),
 FirstName VARCHAR(50) OPTIONS(XPATH './ns1:FirstName/text()',
             TEMPLATE '<ns1:FirstName>&column</ns1:FirstName>'),
 LastName VARCHAR(50)  OPTIONS(XPATH './ns1:LastName/text()',
             TEMPLATE '<ns1:LastName>&column</ns1:LastName>'),
 AccountId VARCHAR(255)  OPTIONS(XPATH './ns1:AccountId/text()'),
 PrimaryAccountName VARCHAR(100)
      OPTIONS(XPATH './ns1:PrimaryAccountName/text()'),
 PrimaryPostalCode VARCHAR(30)
      OPTIONS(XPATH './ns1:PrimaryPostalCode/text()'),
 PrimaryStreetAddress VARCHAR(200)
```

```
     OPTIONS(XPATH './ns1:PrimaryStreetAddress/text()'),
SalesRep VARCHAR(255)  OPTIONS(XPATH './ns1:SalesRep/text()'),
State VARCHAR(255)  OPTIONS(XPATH './ns1:State/text()'))
 FOR SERVER siebel_server
 OPTIONS(XPATH '//ns1:sieb_ssa_Contact_Contact',
 TEMPLATE '<ns1:sieb_ssa_Contact_Contact>
                  &Id[1,1] &FirstName[0,1] &LastName[0,1]
           </ns1:sieb_ssa_Contact_Contact>',
 BUSOBJ_NAME 'sieb_ssa_Contact_Contact',
 NAMESPACES  'ns1="http://www.ibm.com/websphere/
               crossworlds/2002/BOSchema/
                 sieb_ssa_Contact_Contact"'
 REQUIRE_PREDICATE 'N',
 RESULTSET_SIZE '50');
```

The BUSOBJ_NAME nickname option is the name of the XML schema definition (XSD) file that represents the business object definition.

CREATE NICKNAME statement – examples for the WebSphere Business Integration wrapper

The examples in this section show how to use the CREATE NICKNAME statement.

Example 1: Flat business object

Figure 9 is a portion of an XSD file that represents a WebSphere Business Integration business object definition for a Siebel Business Component. The business object definition hierarchy consists of a single level, which contains only the root business object. The DB2 Control Center creates a single relational nickname to represent this business object definition.

In the XSD file, the ID element is flagged as a required input column by adding the isRequired=″true″ flag in the annotation section for the element. The FirstName and LastName columns are designated as optional input columns by adding the isRequired=″false″ flag.

Figure 9. XML schema file for a flat business object

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<xsd:schema elementFormDefault="qualified"
  targetNamespace="http://www.ibm.com/websphere/
         crossworlds/2002/BOSchema/sieb_ssa_Contact_Contact"
  ...
 <xsd:element name="sieb_ssa_Contact_Contact">
  <xsd:annotation>
    <xsd:appinfo>
     <bx:boDefinition version="1.0.0">
     <bx:appSpecificInfo>ON=Contact;CN=Contact</bx:appSpecificInfo>
     </bx:boDefinition>
    </xsd:appinfo>
   </xsd:annotation>
<xsd:complexType>
 <xsd:sequence>
   <xsd:element name="Id" minOccurs="0">
  <xsd:annotation>
   <xsd:appinfo>
    <bx:boAttribute>
     <bx:appSpecificInfo>FN=Id</bx:appSpecificInfo>
     <bx:attributeInfo isForeignKey="false"
                       isKey="true" isRequired="true" />
    </bx:boAttribute>
   </xsd:appinfo>
  </xsd:annotation>
 ...
 </xsd:simpleType>
</xsd:element>

<xsd:element name="FirstName" minOccurs="1">
  <xsd:annotation>
  ...
  <bx:appSpecificInfo>FN=First Name</bx:appSpecificInfo>
  <bx:attributeInfo isForeignKey="false"
      isKey="false" isRequired="false" />
 ...
  </xsd:annotation>
  ...
</xsd:element>
<xsd:element name="LastName" minOccurs="1">
  <xsd:annotation>
 ...
  <bx:appSpecificInfo>FN=Last Name</bx:appSpecificInfo>
  <bx:attributeInfo isForeignKey="false"
      isKey="false" isRequired="false" />
  ...
 </xsd:annotation>
...
</xsd:element>

<xsd:element name="AccountId" minOccurs="0">
 <xsd:annotation>
 ...
 <bx:appSpecificInfo>FN=Account Id</bx:appSpecificInfo>
 <bx:attributeInfo isForeignKey="false"
        isKey="false" />
 ...
  </xsd:annotation>
...
</xsd:element>
<xsd:element name="PrimaryAccountName" minOccurs="0">
 <xsd:annotation>
  ...
    <bx:appSpecificInfo>FN=Primary Account Name</bx:appSpecificInfo>
  <bx:attributeInfo isForeignKey="false" isKey="false" />
  ...
 </xsd:annotation>
 ...
</xsd:element>

<xsd:element name="PrimaryPostalCode" minOccurs="0">
 <xsd:annotation>
 ...
  <bx:appSpecificInfo>FN=Primary Postal Code</bx:appSpecificInfo>
  <bx:attributeInfo isForeignKey="false" isKey="false" />
 ...
 </xsd:annotation>
</xsd:element>
<xsd:element name="PrimaryStreetAddress" minOccurs="0">
 <xsd:annotation>
  ...
 <bx:appSpecificInfo>FN=Primary Street Address</bx:appSpecificInfo>
   <bx:attributeInfo isForeignKey="false" isKey="false" />
 ...
 </xsd:annotation>
...
</xsd:element>

<xsd:element name="SalesRep" minOccurs="0">
<xsd:annotation>
...
 <bx:boAttribute>
```

```
<bx:appSpecificInfo>FN=Sales Rep</bx:appSpecificInfo>
<bx:attributeInfo isForeignKey="false" isKey="false" />
 ...
 </xsd:annotation>
 ...
</xsd:element>
<xsd:element name="State" minOccurs="0">
 <xsd:annotation>
 ...
 <bx:boAttribute>
 <bx:appSpecificInfo>FN=State</bx:appSpecificInfo>
 <bx:attributeInfo isForeignKey="false" isKey="false" />
 ...
 </xsd:annotation>
...
</xsd:element>
<xsd:element name="ObjectEventId"
    type="xsd:string" minOccurs="0" />
</xsd:sequence>
...
</xsd:schema>
```

The DB2 Control Center generates the following CREATE NICKNAME statement from the XSD file that is shown in Figure 9 on page 108. A TEMPLATE option value is specified for each input column. The column option templates are associated with the nickname option template. The input columns are also referenced in the option value of the nickname level TEMPLATE. The nickname option template provides the structure for the input business object. The value of the minOccurs attribute for each of the input column references in the nickname template value determines whether the input column is a required or optional column. The reference for the ID column is specified as &Id[1,1]. The references for the FirstName and LastName columns are specified as &FirstName [0,1] and &LastName [0,1]. All output columns include an XPATH column option value. The nickname is for a flat business object that does not contain any children (elements of cardinality ′n′):

```
CREATE NICKNAME sieb_ssa_Contact_Contact_NN(
 Id VARCHAR(15)  OPTIONS(XPATH './ns1:Id/text()',
     TEMPLATE '<ns1:Id>&column</ns1:Id>'),
 FirstName VARCHAR(50)  OPTIONS(XPATH './ns1:FirstName/text()',
     TEMPLATE '<ns1:FirstName>&column</ns1:FirstName>'),
 LastName VARCHAR(50)  OPTIONS(XPATH './ns1:LastName/text()',
     TEMPLATE '<ns1:LastName>&column</ns1:LastName>'),
 AccountId VARCHAR(255)
     OPTIONS(XPATH './ns1:AccountId/text()'),
 PrimaryAccountName VARCHAR(100)
     OPTIONS(XPATH './ns1:PrimaryAccountName/text()'),
 PrimaryPostalCode VARCHAR(30)
     OPTIONS(XPATH './ns1:PrimaryPostalCode/text()'),
 PrimaryStreetAddress VARCHAR(200)
     OPTIONS(XPATH './ns1:PrimaryStreetAddress/text()'),
 SalesRep VARCHAR(255)  OPTIONS(XPATH './ns1:SalesRep/text()'),
 State VARCHAR(255)  OPTIONS(XPATH './ns1:State/text()'))
  FOR SERVER siebel_server
  OPTIONS(XPATH '//ns1:sieb_ssa_Contact_Contact',
  TEMPLATE '<ns1:sieb_ssa_Contact_Contact>
                 &Id[1,1] &FirstName[0,1] &LastName[0,1]
             </ns1:sieb_ssa_Contact_Contact>',
  BUSOBJ_NAME 'sieb_ssa_Contact_Contact',
  NAMESPACES  'ns1="http://www.ibm.com/websphere/
                 crossworlds/2002/BOSchema/sieb_ssa_Contact_Contact"');
```

Example 2: Hierarchical business object

In this example, the business object definition consists of a two-level hierarchy comprising the root business object and two child business objects, or three XSD files. Only two nicknames are generated to represent the business object definition hierarchy. The sap_customeraddress child business object has a cardinality of 1, which is indicated by the absence of a maxOccurs attribute specification in the element definition. All of the columns of sap_customeraddress are included in the root nickname, sap_bapi_customer_getdetail2_NN. The sap_customerbankdetail child business object has a cardinality of n, which is indicated by the maxOccurs=″unbounded″ specification in the element definition. The sap_customerbankdetail child business object is mapped to a separate child nickname, sap_bapi_customer_getdetail2_sap_customerbankdetail_NN. The child nickname is associated with the root nickname by a special primary key-foreign key relationship.

Figure 10. SAP hierarchical business object: customer_getdetail2

```
...
<xsd:element name="sap_bapi_customer_getdetail2">
  <xsd:annotation>
   <xsd:appinfo>
    <bx:boDefinition version="3.0.0" />
   </xsd:appinfo>
  </xsd:annotation>

...
<xsd:element name="COMPANYCODE" minOccurs="0">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boAttribute>
    <bx:appSpecificInfo>ICOMPANYCODE:</bx:appSpecificInfo>
    <bx:attributeInfo isForeignKey="false" isKey="true" />
   </bx:boAttribute>
  </xsd:appinfo>
</xsd:annotation>

...
</xsd:element>
<xsd:element name="CUSTOMERNO" minOccurs="1">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boAttribute>
    <bx:appSpecificInfo>ICUSTOMERNO:</bx:appSpecificInfo>
    <bx:attributeInfo isForeignKey="false" isKey="false" />
   </bx:boAttribute>
  </xsd:appinfo>
 </xsd:annotation>
...
</xsd:element>

<xsd:element name="sap_customeraddress" minOccurs="0">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boAttribute>
    <bx:appSpecificInfo>:ECUSTOMERADDRESS</bx:appSpecificInfo>
    <bx:attributeInfo isForeignKey="false" isKey="false" />
    <bx:childObjectInfo relationship="Containment" version="3.0.0" />
   </bx:boAttribute>
  </xsd:appinfo>
 </xsd:annotation>
...
```

```
<xsd:element ref="sap_customeraddress:sap_customeraddress" />
</xsd:sequence>

</xsd:complexType>
</xsd:element>
<xsd:element name="sap_customerbankdetail" minOccurs="0">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boAttribute>
    <bx:appSpecificInfo>
        ICUSTOMERBANKDETAIL:ECUSTOMERBANKDETAIL
    </bx:appSpecificInfo>
    <bx:attributeInfo isForeignKey="false" isKey="false" />
    <bx:childObjectInfo relationship="Containment" version="3.0.0" />
   </bx:boAttribute>
  </xsd:appinfo>
 </xsd:annotation>
<xsd:complexType>
<xsd:sequence>
 <xsd:element ref="sap_customerbankdetail:sap_customerbankdetail"
     maxOccurs="unbounded" />
</xsd:sequence>
<xsd:attribute name="size" type="xsd:positiveInteger"
   default="1" />
</xsd:complexType>
</xsd:element>
<xsd:element name="ObjectEventId" type="xsd:string"
   minOccurs="0" />
</xsd:sequence>
...
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boVerb>
    <bx:appSpecificInfo>
        bapi.client.Bapi_customer_getdetail2
    </bx:appSpecificInfo>
   </bx:boVerb>
  </xsd:appinfo>
 </xsd:annotation>
</xsd:enumeration>
<xsd:enumeration value="Update" />
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
...
```

Figure 11. SAP hierarchical business object: customer_address

```
<xsd:element name="sap_customeraddress">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boDefinition version="3.0.0">
    <bx:appSpecificInfo>:ECUSTOMERADDRESS</bx:appSpecificInfo>
   </bx:boDefinition>
  </xsd:appinfo>
 </xsd:annotation>
<xsd:complexType>
<xsd:sequence>
<xsd:element name="CUSTOMER" minOccurs="0">
 <xsd:annotation>
  <xsd:appinfo>
```

```
   <bx:boAttribute>
    <bx:appSpecificInfo>:ECUSTOMER</bx:appSpecificInfo>
    <bx:attributeInfo isForeignKey="false" isKey="true" />
   </bx:boAttribute>
  </xsd:appinfo>
 </xsd:annotation>
...
</xsd:element>
<xsd:element name="NAME" minOccurs="0">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boAttribute>
    <bx:appSpecificInfo>:ENAME</bx:appSpecificInfo>
    <bx:attributeInfo isForeignKey="false" isKey="false" />
   </bx:boAttribute>
  </xsd:appinfo>
 </xsd:annotation>
...
</xsd:element>
<xsd:element name="CITY" minOccurs="0">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boAttribute>
    <bx:appSpecificInfo>:ECITY</bx:appSpecificInfo>
    <bx:attributeInfo isForeignKey="false" isKey="false" />
   </bx:boAttribute>
  </xsd:appinfo>
 </xsd:annotation>
...
</xsd:element>
<xsd:element name="POSTL_CODE" minOccurs="0">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boAttribute>
    <bx:appSpecificInfo>:EPOSTL_CODE</bx:appSpecificInfo>
    <bx:attributeInfo isForeignKey="false" isKey="false" />
   </bx:boAttribute>
  </xsd:appinfo>
 </xsd:annotation>
...
</xsd:element>
<xsd:element name="STREET" minOccurs="0">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boAttribute>
    <bx:appSpecificInfo>:ESTREET</bx:appSpecificInfo>
    <bx:attributeInfo isForeignKey="false" isKey="false" />
   </bx:boAttribute>
</xsd:appinfo>
</xsd:annotation>
...
</xsd:element>
<xsd:element name="REGION" minOccurs="0">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boAttribute>
    <bx:appSpecificInfo>:EREGION</bx:appSpecificInfo>
    <bx:attributeInfo isForeignKey="false" isKey="false" />
   </bx:boAttribute>
  </xsd:appinfo>
 </xsd:annotation>
...
</xsd:element>
</xsd:sequence>
...
```

Figure 12. SAP hierarchical business object: bank_detail

```
...
<xsd:element name= "sap_customerbankdetail ">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boDefinition version= "3.0.0 ">
    <bx:appSpecificInfo>
       ICUSTOMERBANKDETAIL:ECUSTOMERBANKDETAIL
    </bx:appSpecificInfo>
   </bx:boDefinition>
   </xsd:appinfo>
  </xsd:annotation>

<xsd:complexType>
<xsd:sequence>
<xsd:element name= "CUSTOMER " minOccurs= "0 ">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boAttribute>
    <bx:appSpecificInfo>ICUSTOMER:ECUSTOMER</bx:appSpecificInfo>
    <bx:attributeInfo isForeignKey= "false " isKey= "true " />
   </bx:boAttribute>
  </xsd:appinfo>
</xsd:annotation>
...
</xsd:element>

<xsd:element name= "BANK_KEY " minOccurs= "0 ">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boAttribute>
    <bx:appSpecificInfo>IBANK_KEY:EBANK_KEY</bx:appSpecificInfo>
    <bx:attributeInfo isForeignKey= "false " isKey= "false " />
   </bx:boAttribute>
  </xsd:appinfo>
 </xsd:annotation>
...
</xsd:element>

<xsd:element name= "BANK_ACCT " minOccurs= "0 ">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boAttribute>
    <bx:appSpecificInfo>IBANK_ACCT:EBANK_ACCT</bx:appSpecificInfo>
    <bx:attributeInfo isForeignKey= "false " isKey= "false " />
   </bx:boAttribute>
  </xsd:appinfo>
 </xsd:annotation>
...
</xsd:element>

<xsd:element name= "CTRL_KEY " minOccurs= "0 ">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boAttribute>
   <bx:appSpecificInfo>ICTRL_KEY:ECTRL_KEY</bx:appSpecificInfo>
   <bx:attributeInfo isForeignKey= "false " isKey= "false " />
   </bx:boAttribute>
  </xsd:appinfo>
 </xsd:annotation>
...
</xsd:element>

<xsd:element name= "BANK_REF " minOccurs= "0 ">
 <xsd:annotation>
  <xsd:appinfo>
```

```
   <bx:boAttribute>
    <bx:appSpecificInfo>IBANK_REF:EBANK_REF</bx:appSpecificInfo>
    <bx:attributeInfo isForeignKey= "false " isKey= "false " />
   </bx:boAttribute>
  </xsd:appinfo>
 </xsd:annotation>
...
</xsd:element>
...
```

The DB2 Control Center generates two nicknames from the three SAP XSD files, as shown in Figure 13 and Figure 14 on page 116. Column customerno is marked as a required input column in the nickname level template of the sap_bapi_customer_getdetail2_NN nickname because of the XSD file specifications for the customerno element. Customerno is flagged with an ″I″ prefix and a minOccurs=1 attribute value.

Figure 13. getdetail2 nickname

```
CREATE NICKNAME sap_bapi_customer_getdetail2_NN(
 CUSTOMER VARCHAR(10)
     OPTIONS(XPATH './ns3:sap_customeraddress/
           ns1:sap_customeraddress/ns1:CUSTOMER/text()'),
 NAME VARCHAR(35)
     OPTIONS(XPATH './ns3:sap_customeraddress/
           ns1:sap_customeraddress/ns1:NAME/text()'),
 CITY VARCHAR(35)
     OPTIONS(XPATH './ns3:sap_customeraddress/
           ns1:sap_customeraddress/ns1:CITY/text()'),
 POSTL_CODE VARCHAR(10)
     OPTIONS(XPATH './ns3:sap_customeraddress/
           ns1:sap_customeraddress/ns1:POSTL_CODE/text()'),
 STREET VARCHAR(35)
     OPTIONS(XPATH './ns3:sap_customeraddress/
           ns1:sap_customeraddress/ns1:STREET/text()'),
 REGION VARCHAR(3)
     OPTIONS(XPATH './ns3:sap_customeraddress/
           ns1:sap_customeraddress/ns1:REGION/text()'),
 NN__PKEY VARCHAR(16)  FOR BIT DATA NOT NULL OPTIONS(PRIMARY_KEY 'YES'),
 COMPANYCODE VARCHAR(4)  OPTIONS(XPATH './ns3:COMPANYCODE/text()',
       TEMPLATE '<ns3:COMPANYCODE>&column</ns3:COMPANYCODE>'),
 CUSTOMERNO VARCHAR(10)  OPTIONS(XPATH './ns3:CUSTOMERNO/text()',
       TEMPLATE '<ns3:CUSTOMERNO>&column</ns3:CUSTOMERNO>'),
 ObjectEventId VARCHAR(48)  OPTIONS(XPATH './ns3:ObjectEventId/text()'))
  FOR SERVER sap_server
  OPTIONS(XPATH '//ns3:sap_bapi_customer_getdetail2',
  TEMPLATE '<ns3:sap_bapi_customer_getdetail2>
                   &sap_bapi_customer_getdetail2_sap_customerbankdetail_NN[0,1]
                   &COMPANYCODE[0,1]
                   &CUSTOMERNO[1,1]
              </ns3:sap_bapi_customer_getdetail2>',
  BUSOBJ_NAME 'sap_bapi_customer_getdetail2',
  NAMESPACES  'ns1="http://www.ibm.com/websphere/
                  crossworlds/2002/BOSchema/sap_customeraddress",
                 ns2="http://www.ibm.com/websphere/
                  crossworlds/2002/BOSchema/sap_customerbankdetail",
                 ns3="http://www.ibm.com/websphere/
                  crossworlds/2002/BOSchema/sap_bapi_customer_getdetail2"');
```

Figure 14. Customer bank detail nickname

```
CREATE NICKNAME sap_bapi_customer_getdetail2_sap_customerbankdetail_NN(
 CUSTOMER VARCHAR(10)  OPTIONS(XPATH './ns2:CUSTOMER/text()',
           TEMPLATE '<ns2:CUSTOMER>&column</ns2:CUSTOMER>'),
 BANK_KEY VARCHAR(15)  OPTIONS(XPATH './ns2:BANK_KEY/text()',
           TEMPLATE '<ns2:BANK_KEY>&column</ns2:BANK_KEY>'),
 BANK_ACCT VARCHAR(18)  OPTIONS(XPATH './ns2:BANK_ACCT/text()',
           TEMPLATE '<ns2:BANK_ACCT>&column</ns2:BANK_ACCT>'),
 CTRL_KEY VARCHAR(2)  OPTIONS(XPATH './ns2:CTRL_KEY/text()',
           TEMPLATE '<ns2:CTRL_KEY>&column</ns2:CTRL_KEY>'),
 BANK_REF VARCHAR(20)  OPTIONS(XPATH './ns2:BANK_REF/text()',
           TEMPLATE '<ns2:BANK_REF>&column</ns2:BANK_REF>'),
 NN__FKEY VARCHAR(16)  FOR BIT DATA NOT NULL
       OPTIONS(FOREIGN_KEY 'SAP_BAPI_CUSTOMER_GETDETAIL2_NN'))
  FOR SERVER sap_server
  OPTIONS(XPATH './ns3:sap_customerbankdetail/ns2:sap_customerbankdetail',
  TEMPLATE '<ns3:sap_customerbankdetail>
                <ns2:sap_customerbankdetail>
                  &CUSTOMER[0,1]
                  &BANK_KEY[0,1]
                  &BANK_ACCT[0,1]
                  &CTRL_KEY[0,1]
                  &BANK_REF[0,1]
                </ns2:sap_customerbankdetail>
              </ns3:sap_customerbankdetail>',
  NAMESPACES  'ns1="http://www.ibm.com/websphere/
                    crossworlds/2002/BOSchema/sap_customeraddress",
                 ns2="http://www.ibm.com/websphere/
                    crossworlds/2002/BOSchema/sap_customerbankdetail",
                 ns3="http://www.ibm.com/websphere/
                    crossworlds/2002/BOSchema/sap_bapi_customer_getdetail2"');
```

Example 3: Primary and foreign keys

The PRIMARY_KEY and FOREIGN_KEY columns are used to define relationships between the parent and child nicknames. Each parent nickname must have a primary key column option. You define the children of a parent nickname with the foreign key column option that references the primary key column of a parent nickname. A nickname can have multiple children, but a nickname can have only one parent.

Primary and foreign key values for the WebSphere Business Integration wrapper are only valid and unique within a single query. A primary and foreign key cannot be used to retrieve a row with a second query. The values cannot persist into another table, because the uniqueness of the value is not guaranteed if that table is populated with multiple queries.

The following CREATE NICKNAME statements are derived from the XML schema definition files that are shown in Figure 10 on page 111, Figure 11 on page 112, and Figure 12 on page 114. The foreign key, nn_fkey, uniquely associates the child nickname, sap_bapi_customer_getdetail2_sap_customerbankdetail_nn, with the parent nickname sap_bapi_customer_getdetail2_nn. The parent nickname also uses a reference to the child nickname in the nickname options template structure.

```
CREATE NICKNAME sap_bapi_customer_getdetail2_NN(
 CUSTOMER VARCHAR(10)
     OPTIONS(XPATH './ns3:sap_customeraddress/
         ns1:sap_customeraddress/ns1:CUSTOMER/text()'),
 NAME VARCHAR(35)
     OPTIONS(XPATH './ns3:sap_customeraddress/
         ns1:sap_customeraddress/ns1:NAME/text()'),
 ...
 NN__PKEY VARCHAR(16)  FOR BIT DATA NOT NULL
            OPTIONS(PRIMARY_KEY 'YES'),
...
TEMPLATE '<ns3:sap_bapi_customer_getdetail2>
            &sap_bapi_customer_getdetail2_sap_customerbankdetail_NN[0,1]
            &COMPANYCODE[0,1]
            &CUSTOMERNO[1,1]
          </ns3:sap_bapi_customer_getdetail2>',

...
CREATE NICKNAME sap_bapi_customer_getdetail2_sap_customerbankdetail_NN(
 CUSTOMER VARCHAR(10)  OPTIONS(XPATH './ns2:CUSTOMER/text()',
           TEMPLATE '<ns2:CUSTOMER>&column</ns2:CUSTOMER>'),
 BANK_KEY VARCHAR(15)  OPTIONS(XPATH './ns2:BANK_KEY/text()',
           TEMPLATE '<ns2:BANK_KEY>&column</ns2:BANK_KEY>'),
 BANK_ACCT VARCHAR(18)  OPTIONS(XPATH './ns2:BANK_ACCT/text()',
           TEMPLATE '<ns2:BANK_ACCT>&column</ns2:BANK_ACCT>'),
 CTRL_KEY VARCHAR(2)  OPTIONS(XPATH './ns2:CTRL_KEY/text()',
           TEMPLATE '<ns2:CTRL_KEY>&column</ns2:CTRL_KEY>'),
 BANK_REF VARCHAR(20)  OPTIONS(XPATH './ns2:BANK_REF/text()',
           TEMPLATE '<ns2:BANK_REF>&column</ns2:BANK_REF>'),
 NN__FKEY VARCHAR(16) FOR BIT DATA NOT NULL
         OPTIONS(FOREIGN_KEY 'SAP_BAPI_CUSTOMER_GETDETAIL2_NN'))
  FOR SERVER sap_server
  OPTIONS(XPATH './ns3:sap_customerbankdetail/ns2:sap_customerbankdetail',
  TEMPLATE '<ns3:sap_customerbankdetail>
                <ns2:sap_customerbankdetail>
                  &CUSTOMER[0,1]
                  &BANK_KEY[0,1]
                  &BANK_ACCT[0,1]
                  &CTRL_KEY[0,1]
                  &BANK_REF[0,1]
                </ns2:sap_customerbankdetail>
              </ns3:sap_customerbankdetail>',

...
```

Example 4: Namespaces to resolve XPath expression prefixes

The NAMESPACES option is a comma separated list of name-value pairs. It resolves the prefixes that are used in XPath expressions with the namespace URIs that are defined in the XML schemas. These XPath expressions are applied on the business objects (the XML document) that are returned from WebSphere Business Integration Adapter. The following example includes namespace prefixes and the definitions of those prefixes:

```
CREATE NICKNAME sap_customer
  (
sap_customeraddress_CUSTOMER VARCHAR(10)
  OPTIONS(XPATH './ns5:sap_customeraddress/
         ns2:sap_customeraddress/ns2:CUSTOMER/text()'),
sap_customeraddress_NAME VARCHAR(35)
  OPTIONS(XPATH './ns5:sap_customeraddress/
         ns2:sap_customeraddress/ns2:NAME/text()'),

...

sap_bapi_customer_getdet1_PKEY VARCHAR(16) FOR BIT DATA NOT NULL
  OPTIONS(PRIMARY_KEY 'YES'),
COMPANYCODE VARCHAR(4)
  OPTIONS(XPATH './ns5:COMPANYCODE/text()',
          TEMPLATE '<ns5:COMPANYCODE>&column</ns5:COMPANYCODE>'),
CUSTOMERNO VARCHAR(10)
  OPTIONS(XPATH './ns5:CUSTOMERNO/text()',
  TEMPLATE '<ns5:CUSTOMERNO>&column</ns5:CUSTOMERNO>'),
ObjectEventId VARCHAR(48)
  OPTIONS(XPATH './ns5:ObjectEventId/text()')
 )
FOR SERVER SAP_SOURCE
  OPTIONS (
  XPATH '//ns5:sap_bapi_customer_getdetail2',
  TEMPLATE
         '<ns5:sap_bapi_customer_getdetail2>
             &customerbankdetail_NN[0,1] &COMPANYCODE[0,1] &CUSTOMERNO[1,1]
          </ns5:sap_bapi_customer_getdetail2>',
  BUSOBJ_NAME 'sap_bapi_customer_getdetail2',
  NAMESPACES  '
ns2="http://www.ibm.com/websphere/
          crossworlds/2002/BOSchema/sap_customeraddress",
...
ns5="http://www.ibm.com/websphere/
          crossworlds/2002/BOSchema/sap_bapi_customer_getdetail2",
ns6="http://www.ibm.com/websphere/
           crossworlds/2002/BOSchema/sap_return"'
);
```

TEMPLATE option for the WebSphere Business Integration wrapper

The WebSphere® Business Integration wrapper needs the TEMPLATE option on the CREATE NICKNAME statement at the time that the nickname is created.

The required and optional columns vary according to the application and the associated adapter. Identify the required and optional input columns by specifying the appropriate template option values for those columns. Before you use the DB2® Control Center to create the nicknames, you must modify the XML schema definition file to flag the required and optional input columns.

SAP BAPI

The DB2 Control Center determines the required and optional input columns based on the value of specific flags in the XML schema definition (XSD) files that represent the business object definition.

In the annotation section of an element at any level of the business object hierarchy (parent or child business objects), an "I" prefix in the appSpecificInfo value indicates an import parameter for the SAP BAPI to which the business object definition maps. An "E" prefix indicates an export parameter for the SAP BAPI. Some elements can be both import and export parameters for a BAPI. The following example shows an element which is both an import and an export parameter:

```
<bx:appSpecificInfo>ICOMPANYCODE:ECOMPANYCODE</bx:appSpecificInfo>
```

The prefixes are generated automatically by the WebSphere Business Integration Object Discovery Agent tool based on information that is extracted from the SAP business object repository.

If an element that represents an import parameter (an ″I″ prefix in the appSpecificInfo value) is specified with the attribute minOccurs=1, the DB2 Control Center identifies the element as a required input parameter and flags the elements as a required input column in the nickname definition. The WebSphere Business Integration Object Discovery Agent tool does not automatically set the value of minOccurs to 1 for the required input parameters of the SAP BAPI. You must reference the SAP Business Object

Repository to determine all the required input parameters for the BAPI that you want to access. Then, you must edit the corresponding elements in the XML schema file by manually setting the attribute to minOccurs=1. If the minOccurs attribute value for an input parameter remains as the default value of 0, then the DB2 Control Center specifies the column as an optional input column in the nickname hierarchy that is generated.

The following example shows an optional input column:

```
<xsd:element name="Company_code" minOccurs="0">
  <xsd:annotation>
   <xsd:appinfo>
     <bx:boAttribute>
       <bx:appSpecificInfo>ICOMPANYCODE:</bx:appSpecificInfo>
       <bx:attributeInfo isForeignKey="false" isKey="true" />
     </bx:boAttribute>
   </xsd:appinfo>
  </xsd:annotation>
  <xsd:simpleType>
   <xsd:restriction base="xsd:string">
     <xsd:maxLength value="4" />
   </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

The following example shows a required input column:

```
<xsd:element name="Company_id" minOccurs="1">
  <xsd:annotation>
   <xsd:appinfo>
     <bx:boAttribute>
       <bx:appSpecificInfo>ICOMPANYID:</bx:appSpecificInfo>
       <bx:attributeInfo isForeignKey="true" isKey="true" />
     </bx:boAttribute>
   </xsd:appinfo>
  </xsd:annotation>
  <xsd:simpleType>
   <xsd:restriction base="xsd:string">
     <xsd:maxLength value="4" />
   </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

The required and optional input columns for SAP business applications are designated by the syntax shown in the following table:

Table 25. Flagging schema for SAP input column information

Flags used in SAP XSD files	Required input column	Column reference in nickname template
Any element anywhere in the hierarchy with the prefix = 'I' and minOccurs=1	Yes	&columnname[1,1]
Any element anywhere in the hierarchy with the prefix = 'I' and minOccurs=0	No	&columnname[0,1]

Siebel and PeopleSoft

The DB2 Control Center determines the required and optional input columns based on the existence and the value of the isRequired flag in the attributeInfo section of the annotation for the element. If there is no isRequired flag, then the column is not an input column. The WebSphere Business Integration Object Discovery Agent tool does not automatically generate these flags in the XSD file. You must identify the required and optional input columns, and flag them appropriately in the XSD file before you use the DB2 Control Center to generate the nickname DDL statement.

The following example shows the flags for a required input column and optional input columns in the XSD file for a Siebel or PeopleSoft business object definition.

```
<xsd:element name="sieb_ssa_Contact_Contact">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boDefinition version="1.0.0">
   <bx:appSpecificInfo>ON=Contact;CN=Contact</bx:appSpecificInfo>
   </bx:boDefinition>
  </xsd:appinfo>
 </xsd:annotation>
<xsd:complexType>
<xsd:sequence>
<xsd:element name="Id" minOccurs="0">
 <xsd:annotation>
  <xsd:appinfo>
   <bx:boAttribute>
    <bx:appSpecificInfo>FN=Id</bx:appSpecificInfo>
    <bx:attributeInfo isForeignKey="false"
       isKey="true" isRequired="true" />
   </bx:boAttribute>
  </xsd:appinfo>
 </xsd:annotation>
...
</xsd:element>

...

...
<xsd:element name="FirstName" minOccurs="1">
<xsd:annotation>
 <xsd:appinfo>
  <bx:boAttribute>
  <bx:appSpecificInfo>FN=First Name</bx:appSpecificInfo>
  <bx:attributeInfo isForeignKey="false" isKey="false"
        isRequired="false" />
  </bx:boAttribute>
 </xsd:appinfo>
</xsd:annotation>
<xsd:simpleType>
 <xsd:restriction base="xsd:string">
  <xsd:maxLength value="50" />
 </xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="LastName" minOccurs="1">
<xsd:annotation>
 <xsd:appinfo>
  <bx:boAttribute>
   <bx:appSpecificInfo>FN=Last Name</bx:appSpecificInfo>
   <bx:attributeInfo isForeignKey="false" isKey="false"
         isRequired="false" />
  </bx:boAttribute>
 </xsd:appinfo>
</xsd:annotation>
...
```

Figure 15. Portion of a Siebel business object definition

The required and optional input columns for Siebel and PeopleSoft business applications are designated by the syntax shown in the following table:

Table 26. Flagging schema for Siebel and PeopleSoft input column information

Flags used in Siebel and PeopleSoft XSD files	Required input column	Column reference in nickname template
isRequired=″true″	Yes	&columnname[1,1]
isRequired=″false″	No	&columnname[0,1]

The following example shows the DDL that the DB2 Control Center creates that is based on the XSD file in the figure Figure 15 on page 120. The XSD file in that figure includes a value of false for the isRequired attribute.

```
CREATE NICKNAME sieb_ssa_Contact_Contact_NN(
 Id VARCHAR(15)  OPTIONS(XPATH './ns1:Id/text()',
           TEMPLATE '<ns1:Id>&column</ns1:Id>'),
 FirstName VARCHAR(50)  OPTIONS(XPATH './ns1:FirstName/text()',
           TEMPLATE '<ns1:FirstName>&column</ns1:FirstName>'),
 LastName VARCHAR(50)  OPTIONS(XPATH './ns1:LastName/text()',
           TEMPLATE '<ns1:LastName>&column</ns1:LastName>'),
 AccountId VARCHAR(255)  OPTIONS(XPATH './ns1:AccountId/text()'),
 PrimaryAccountName VARCHAR(100)
           OPTIONS(XPATH './ns1:PrimaryAccountName/text()'),
 PrimaryPostalCode VARCHAR(30)
           OPTIONS(XPATH './ns1:PrimaryPostalCode/text()'),
 PrimaryStreetAddress VARCHAR(200)
           OPTIONS(XPATH './ns1:PrimaryStreetAddress/text()'),
 SalesRep VARCHAR(255)  OPTIONS(XPATH './ns1:SalesRep/text()'),
 State VARCHAR(255)  OPTIONS(XPATH './ns1:State/text()'))
  FOR SERVER siebel_server
  OPTIONS(XPATH '//ns1:sieb_ssa_Contact_Contact',
  TEMPLATE '<ns1:sieb_ssa_Contact_Contact>
                &Id[1,1] &FirstName[0,1] &LastName[0,1]
            </ns1:sieb_ssa_Contact_Contact>',
  BUSOBJ_NAME 'sieb_ssa_Contact_Contact',
  NAMESPACES  'ns1="http://www.ibm.com/websphere/
                crossworlds/2002/BOSchema/sieb_ssa_Contact_Contact"');
```

Nickname requirements for result sets

The WebSphere Business Integration wrapper supports results sets that include multiple business objects. You must create nicknames according to restrictions for each business application.

Settings to enable result sets from mySAP.com

The SAP business API (BAPI) does not provide support for result sets. However, you can create a wrapper business object definition that sends multiple requests to SAP to return a result set.

The wrapper business object works only with an SAP business object that provides a getlist*() and a getdetail*() BAPI, and only if the getdetail*() BAPI has only one required input parameter.

For a wrapper business object that is generated by using the Object Discovery Agent (ODA) tool, you do not need to specify the verb value. The ODA tool automatically assigns a RetrieveByContent verb value to the wrapper business object.

To enable result sets for mySAP.com, use the ODA Business Object wizard that is packaged with the adapter to create a wrapper business object definition by using the following settings:

- Set the value of the ResultSet property to true.
- In the Select Source window, select the getlist*() and getdetail*() BAPIs that you want to access.
- Specify a name for the ResultSet Name property and set the UseFieldName property to Yes.
- Select the getlist*() BAPI for the Query Object. The remaining BAPI is automatically selected as a Result Object.
- Select the key value or the field on which to create the ResultSet relationship. If the key field is in a child business object of the query object, select the table structure that represents the child business object.
- Select the input parameter in the result object (the getdetail*() BAPI) to map to the key value.
- Select **Required** for all the input parameters and fields of the Query Object that map to required parameters of the getlist*() BAPI in the SAP application.
- Use only the top level XSD file for the wrapper business object for nickname generation.

For example, this query retrieves the name and bank account number for all customers in zip codes:

```
SELECT a.NAME, c.BANK_ACCT
FROM Customer_resultset a, Idrange b, CustomerBankDetail c
WHERE b.SIGN='I', b.OPTION='BT', b.LOW='000000000', b.HIGH='999999999' and
a.POSTL_CODE IN ('95141', '95123', '95136');
```

Settings to enable result sets from Siebel business applications

If you have enabled result sets, you do not need to manually specify elements in the business object XSD files to identify required columns. However, columns that are flagged as input columns (whether required or optional) support only equality predicates in all queries. You can use non-equality predicates on columns that are not declared as input columns in the nickname hierarchy.

The Discover Nicknames wizard in the DB2 Control Center by default marks all elements at the root business object level as optional input columns. To convert an input column to an output column, remove the TEMPLATE column option specification and the reference for the column in the nickname level TEMPLATE option.

To support result sets, you must specify a verb value of RetrieveByContent for the business object in the Object Discovery Agent.

Example

The following query returns all employees in the specified zip code with salaries greater than $100,000. All columns in the following query map to elements in the root business object, and all these columns were automatically declared as optional input columns. The salary column must be flagged as an output only column in the nickname definition or the wrapper rejects the query.

```
SELECT Lastname, Address FROM siebel_nickname WHERE zipcode=95141 AND salary>100000
```

Settings to enable result sets from PeopleSoft business applications

If result sets are enabled, you do not need to specify elements in the business object XSD files to identify required columns. However, columns that are flagged as input columns (whether required or optional) support only equality predicates in all queries. You can use non-equality predicates on columns that are not declared as input columns in the nickname hierarchy.

The Discover Nicknames wizard in the DB2 Control Center by default marks all Find Key columns in the associated PeopleSoft component interface as optional input columns in the nickname hierarchy. To convert an input column to an output column, remove the TEMPLATE column option specification and the reference for the column in the nickname level TEMPLATE option.

When you use the PeopleSoft adapter, a WebSphere Business Integration business object definition maps to a PeopleSoft component interface, which is the API that is associated with a PeopleSoft component.

To support result sets, you must specify a verb value of RetrieveByContent for the business object in the Object Discovery Agent.

In PeopleSoft component interfaces, a Find Key column does not uniquely identify a row, so a query on a Find Key column can return multiple results. Do not declare as input columns any PeopleSoft component interface columns that are not Find Key columns. In addition, queries on any Find Key columns must include only equality predicates if the column was declared as an input column in the nickname hierarchy.

In PeopleSoft component interfaces, a Get Key column uniquely identifies a row, so a query on a Get Key column returns a unique result. Typically, Get Key columns are also identified as Find Key columns; however, Get Key columns are not always identified as Find Key columns. To successfully query a Get Key column that is not also identified as a Find Key column, the adapter requires the verb property to be Retrieve, thus retrieving a single business object based on a unique identifier.

If you want to use Get Key columns that are not also Find Key columns in a query, do not enable result sets for PeopleSoft component interfaces. To use result sets, be sure that all Get Key columns are also identified as Find Key columns.

Memory limitations in business application adapters

WebSphere Business Integration adapters run in a Java™ virtual machine (JVM) that has an upper memory limit of 2 GB. If this upper limit is reached, the JVM stops with a memory exception, and the adapter must be restarted.

By default, WebSphere Business Integration adapters are configured with a memory limit of less that 2 GB, which limits the maximum number of results that can be returned by queries. This limit can be changed up to the JVM memory limit.

If your query results approach this upper memory limit, change the parameters of the query to reduce the number of results that are returned.

Nickname options for business application data sources

You can use nickname options to support result sets.

Nickname options

To support result sets, the WebSphere Business Integration wrapper includes these nickname options:

REQUIRE_PREDICATE
: Specify Y to require an equality predicate on at least one input column in all queries on the nickname hierarchy, which can limit the size of the result set. If you know that the size of the result set that returns from a query with no predicate does not exceed the JVM memory limit, you can set the value of the REQUIRE_PREDICATE nickname option to N.

RESULTSET_SIZE
: Use this option to specify how many business objects the adapter returns to the wrapper. The default value is 0, which specifies that all business objects that match the query are returned. Specify any other value to have the adapter return the specified number of business objects. You must enable result sets in the wrapper (RESULTSET_ENABLED set to Yes) for the RESULTSET_SIZE option to work. If you specify a nonzero value for RESULTSET_SIZE, an incomplete result might be returned. Any rows that exceed the specified number are discarded, and the wrapper issues a warning message that indicates that an incomplete result set was returned to the application.

Creating federated views for business application nicknames

Create federated views for WebSphere Business Integration nicknames to ensure that queries that join pieces of a WebSphere Business Integration nickname hierarchy run properly.

Procedure

To create federated views for business application nicknames:

1. Define a view for each business application nickname as a join of all the nicknames on the path to the parent nickname.
2. In the WHERE clause of the view, define the PRIMARY_KEY and FOREIGN_KEY columns as the join predicates.
3. In the SELECT list of the view, include all of the columns of the business application nickname except the column that is designated with the FOREIGN_KEY nickname column option. Do not include the columns in the SELECT list that are designated as PRIMARY_KEY in the parent nicknames along the hierarchy. Include the required input columns for the hierarchy in the select list. These columns might belong to some other nickname in the hierarchy.

The following example shows a view that is based on nicknames that are generated from a business object. The WHERE clause contains the primary and foreign keys that are defined in a CREATE NICKNAME statement.

```
CREATE VIEW view1 (
   customer, bankkey, bankact, customerno )
 AS (SELECT b.customer, b.bank_key, b.bank_acct,
     a.customerno
  FROM sap_bapi_customer_getdetail2_NN a,
       sap_bapi_customer_getdetail2_sap_customerbankdetail_NN b
  WHERE a.NN__PKEY=b.NN__FKEY);
```

Queries that use the view1 view must include predicate values for the required column, such as in the following example:

```
SELECT * FROM view1
  WHERE customerno='1234567890';
```

Business application data sources – example queries

You can join parent and child nicknames, create views, and query using required input columns.

Example 1: Joining parent and child nicknames

If a select statement contains child nickname columns, then you must specify a join predicate with the parent nickname. The join conditions are necessary to maintain the parent-child relationships along the nickname hierarchy. Specify the primary and foreign key join conditions for each parent-child nickname pair in the hierarchy by including the hierarchy association from the child nickname that is referenced to the parent nickname for the hierarchy.

The following queries are nonvalid because they do not contain all of the elements that are necessary to maintain the nickname hierarchy:

```
SELECT * FROM <child_nickname>;
SELECT b.col1
  FROM <parent_nickname> a,<child_nickname> b
  WHERE a.required_column=<value>;
```

The following is an example of a valid query that maintains the nickname hierarchy:

```
SELECT b.col1, a.cola
  FROM <parent_nickname> a,<child_nickname> b
  WHERE a.primary_key_column=b.foreign_key_column
   AND a.required_column=<value>;
```

In the following example, all of the required parent and child input columns are included in the predicates in the WHERE clause. The WHERE clause includes join predicates that specify an equality between the parent primary key column and a child foreign key column:

```
SELECT a.customer, a.name, b.bank_key, b.bank_acct
  FROM sap_bapi_customer_getdetail2_NN a,
       sap_bapi_customer_getdetail2_sap_customerbankdetail_NN b
   WHERE a.nn_pkey=b.nn_fkey
   AND a.customer = 'ABC'
```

Example 2: Views

You can create these types of views:

- A view that is derived from columns in a child nickname so that you can issue queries directly on the child nickname without including the parent-child join conditions in the queries.

  ```
  CREATE VIEW view1 (
     customer, bankkey, bankact, customerno )
   AS (SELECT b.customer, b.bank_key, b.bank_acct,
  ```

```
     a.customerno
  FROM sap_bapi_customer_getdetail2_NN a,
       sap_bapi_customer_getdetail2_sap_customerbankdetail_NN b
  WHERE a.NN__PKEY=b.NN__FKEY);
```

The view definition includes the required input column, customerno, for the nickname hierarchy. Queries that use view view1 must include predicate values for the required column, such as in the following example:

```
SELECT * FROM view1
  WHERE customerno='1234567890';
```

- A global view that includes all columns of the child and parent nicknames except for the primary and foreign key columns.

Example 3: Required input columns

Queries must include predicate values for all required input columns. Required input columns are those columns with TEMPLATE column options definitions and a reference value of [1,1] in the nickname TEMPLATE option value. The wrapper returns an error for any query that does not include the required input columns in the predicates.

The following example shows an nonvalid query. The customers nickname contains a required input column customer_id.

```
SELECT * FROM customers;
```

The following example shows a valid query.

```
SELECT * FROM customers WHERE customer_id = 123;
```

The following example shows a local table in a database that contains customer IDs in the custid column of table local_table. This example is an inner join between the WebSphere Business Integration nickname and the local table.

```
SELECT a.name, a.address
  FROM customers a, local_table b
  WHERE a.customer_id=b.custid;
```

Chapter 7. DB2 family data sources

Configuring access to DB2 family data sources

To configure a federated server to access DB2 family data sources, you must provide the federated server with information about the data sources and objects that you want to access.

Before you begin

- Check the setup of the federated server.
- Check the federated parameter to ensure that federation is enabled.

Restrictions

You can create nicknames only for supported DB2 family data sources. See the topic, Supported data sources, for a list of DB2 family data sources that you can configure a federated server to access.

About this task

You can configure a federated server to access data that is stored in DB2 family data sources by using the DB2 Control Center or by issuing SQL statements on the DB2 command line. The DB2 Control Center includes a wizard to guide you through the steps that are necessary to configure the required federated objects.

Procedure

To add DB2 data sources to a federated server:

1. Catalog the node.
2. Catalog the remote database.
3. Register the wrapper.
4. Register the server definition.
5. Create the user mappings.
6. Test the connection to the DB2 server.
7. Register nicknames for DB2 tables and views.

Cataloging a node entry in the federated node directory

To point to the location of the DB2 data source, you must catalog an entry in the node directory of the federated server.

About this task

The federated server uses the node entry to determine the proper access method to connect to a DB2 data source.

Procedure

To catalog a node entry in the federated node directory:

Issue the appropriate command to catalog the node entry using TCP/IP. For example,

Issue the CATALOG TCPIP NODE command.

For example:

```
CATALOG TCPIP NODE DB2NODE REMOTE SYSTEM42 SERVER DB2TCP42
```

The *DB2NODE* value is the name that you assign to the node that you are cataloging. REMOTE *SYSTEM42* is the host name of the system where the data source resides. SERVER *DB2TCP42* is the service name or primary port number of the server database manager instance. If a service name is used, it is case sensitive.

Cataloging the remote database in the federated server system database directory

You must specify the DB2 database that you want the federated server to connect to. You specify the database by cataloging the remote database in the federated server system database directory.

Procedure

To catalog the remote database in the federated server system database directory:

Use the Client Configuration Assistant (CCA) to catalog the database.

You can use the CATALOG DATABASE command from a command prompt.

For example:

```
CATALOG DATABASE DB2DB390 AS CLIENTS390 AT NODE DB2NODE AUTHENTICATION DCS
```

The value *DB2DB390* is the name of the remote database that you are cataloging in the federated server system database directory. AS *CLIENTS390* is the alias for the database being cataloged. If you do not specify an alias, the database manager uses the database name (for example DB2DB390) as the alias. The AT® NODE *DB2NODE* is the name of the node that you specified when cataloging the node entry in the node directory. AUTHENTICATION SERVER specifies that authentication takes place on the DB2 data source node.

Attention: If the name of the remote database is more than eight characters, you must create a DCS directory entry by issuing the CATALOG DCS DATABASE command.

For example:

```
CATALOG DCS DATABASE SALES400 AS SALES_DB2DB400
```

The value *SALES400* is the alias of the remote database to catalog. This name should match the name of an entry in the federated server system database directory that is associated with the remote node. It is the same name you entered in the CATALOG DATABASE command. AS *SALES_DB2DB400* is the name of the target host database that you want to catalog.

Registering the DB2 wrapper

You must register a wrapper to access DB2 family data sources. Wrappers are used by federated servers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.

Procedure

To register a wrapper:

Issue the CREATE WRAPPER statement and specify the name for the wrapper. The default wrapper name for the DB2 family data sources is DRDA®. If you do not use the default name, you must specify the library name for the wrapper.

For example:

```
CREATE WRAPPER DRDA
```

When you register the wrapper by using the default name, you do not need to specify the library name because the federated server automatically uses the default library name that is associated with the wrapper. If the wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name that you choose. If you do not use the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement.

For example, to register a wrapper with the name db2_wrapper on a federated server that uses the AIX operating system, issue the following statement:

```
CREATE WRAPPER db2_wrapper LIBRARY 'libdb2drda.a'
```

The name of the wrapper library file that you specify depends on the operating system of the federated server. See the list of DB2 wrapper library files for the correct name to specify in the CREATE WRAPPER statement.

DB2 wrapper library files

The DB2 wrapper library files are added to the federated server when you install WebSphere Federation Server.

When you install WebSphere Federation Server, 3 library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2drda.a, libdb2drdaF.a, and libdb2drdaU.a. The default wrapper library file is libdb2drda.a. The other wrapper library files are used with specific wrapper options.

If you decide not to use the default wrapper name when you register a wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

The default directory paths and default wrapper library file names are listed in the following table.

Table 27. DB2 wrapper library locations and file names

Operating system	Directory path	Library file name
AIX	/usr/opt/*install_path*/lib32/ /usr/opt/*install_path*/lib64/	libdb2drda.a

Table 27. DB2 wrapper library locations and file names (continued)

Operating system	Directory path	Library file name
HP-UX	/opt/IBM/db2/*install_path*/lib32 /opt/IBM/db2/*install_path*/lib64	libdb2drda.sl
Linux	/opt/IBM/db2/*install_path*/lib32 /opt/IBM/db2/*install_path*/lib64	libdb2drda.so
Solaris	/opt/IBM/db2/*install_path*/lib32 /opt/IBM/db2/*install_path*/lib64	libdb2drda.so
Windows	%DB2PATH%\bin	db2drda.dll

- *install_path* is the directory path where WebSphere Federation Server is installed on UNIX or Linux.
- %DB2PATH% is the environment variable that is used to specify the directory path where WebSphere Federation Server is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

Registering the server definitions for a DB2 data source

You must register each DB2 server that you want to access in the federated database. When you register the server definition, the federated server connects to the DB2 server and binds packages to the database.

About this task

The federated server needs the authorization and password information to connect to DB2 server. Because the authorization and password information are not stored in the federated global catalog, you must include them in the server definition.

Procedure

To register a server definition for a DB2 data source:

Use one of the following methods to register the server definition.

- Use the Federated Objects wizard in the DB2 Control Center. To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.
- Issue the CREATE SERVER statement.

For example:

```
CREATE SERVER server_definition_name TYPE DB2/ZOS
VERSION version_number WRAPPER DRDA
      AUTHORIZATION "userid" PASSWORD "password"
      OPTIONS (DBNAME 'database_name')
```

Although the *'database_name'* variable is specified as an option in the CREATE SERVER statement, this option is required for DB2 data sources.

When you issue the CREATE SERVER statement, the federated server will test the connection to the DB2 data source server.

After you register the server definition, you can add or drop server options by issuing the ALTER SERVER statement.

CREATE SERVER statement - Examples for the DB2 wrapper

Use the CREATE SERVER statement to register server definitions for the DRDA wrapper. This topic includes a complete example with the required parameters, and an example with additional server options.

Complete example

The following example shows you how to register a server definition for a DRDA wrapper by using the CREATE SERVER statement:

```
CREATE SERVER DB2SERVER TYPE DB2/ZOS VERSION 7 WRAPPER DRDA
      AUTHORIZATION "spalten" PASSWORD "db2guru"
      OPTIONS (DBNAME 'CLIENTS390')
```

DB2SERVER
A name that you assign to the DB2 database server. Duplicate server definition names are not allowed.

TYPE *DB2/ZOS*
Specifies the type of data source server to which you are configuring access.

VERSION *7*
The version of the DB2 database server that you want to access.

WRAPPER *DRDA*
The name that you specified in the CREATE WRAPPER statement.

AUTHORIZATION *"spalten"*
The authorization ID at the data source. This ID must have BINDADD authority at the data source. This value is case sensitive.

PASSWORD *"db2guru"*
The password that is associated with the authorization ID at the data source. This value is case sensitive.

DBNAME *'CLIENTS390'*
The alias for the DB2 database that you want to access. You defined this alias when you cataloged the database using the CATALOG DATABASE command. This value is case sensitive.

Although the database name variable is specified as an option in the CREATE SERVER statement, it is required for DB2 data sources.

Server option example

When you register the server definition, you can specify additional server options in the CREATE SERVER statement. These options include general server options and DB2 data source-specific server options.

The CPU_RATIO option indicates how much faster or slower the data source CPU runs than the federated CPU. If you set the CPU_RATIO option to '0.001', this indicates that the CPU at the remote data source has 1000 times more available capacity than the federated server CPU.

For example:

```
CREATE SERVER DB2SERVER TYPE DB2/ZOS VERSION 7 WRAPPER DRDA
      AUTHORIZATION "spalten" PASSWORD "db2guru"
      OPTIONS (DBNAME 'CLIENTS390', CPU_RATIO '0.001')
```

Creating the user mappings for a DB2 data source

When you attempt to access a remote DB2 server, the federated server establishes a connection to the remote DB2 server by using a user ID and password that are valid for that data source. You must define an association (a user mapping) between the federated server user ID and password and the corresponding data source user ID and password.

About this task

Create a user mapping for each user ID that will access the federated system to send distributed requests to the DB2 data source.

Procedure

To map the local user ID to the DB2 server user ID and password:

Issue a CREATE USER MAPPING statement. For example:

For example:

```
CREATE USER MAPPING FOR local_userID SERVER server_definition_name
      OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password')
```

Although the REMOTE_AUTHID and REMOTE_PASSWORD variables are specified as options in the CREATE USER MAPPING statement, these options are required to access DB2 family data sources.

The REMOTE_AUTHID is the connect authorization ID, not the bind authorization ID.

CREATE USER MAPPING statement - Examples for the DB2 wrapper

Use the CREATE USER MAPPING statement to map a federated server user ID to an DB2 server user ID and password. This topic includes a complete example with the required parameters, and an example that shows you how to use the DB2 special register USER with the CREATE USER MAPPING statement.

Complete example

The following example shows how to map a federated server user ID to a DB2 server user ID and password:

```
CREATE USER MAPPING FOR ALONZO SERVER DB2SERVER
      OPTIONS (REMOTE_AUTHID 'al', REMOTE_PASSWORD 'day2night')
```

ALONZO
Specifies the local user ID that you are mapping to a user ID that is defined at a DB2 family data source server.

SERVER *DB2SERVER*
Specifies the name of the DB2 family data source server that you defined in the CREATE SERVER statement.

REMOTE_AUTHID *'al'*
Specifies the connect authorization user ID at the DB2 family data source server to which you are mapping *ALONZO*. Use single quotation marks to preserve the case of this value unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

REMOTE_PASSWORD *'day2night'*

Specifies the password that is associated with *'al'*. Use single quotation marks to preserve the case of this value unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

Special register example

The following is an example of the CREATE USER MAPPING statement which includes the special register USER:

```
CREATE USER MAPPING FOR USER SERVER DB2SERVER
      OPTIONS (REMOTE_AUTHID 'al', REMOTE_PASSWORD 'day2night')
```

You can use the DB2 special register USER to map the authorization ID of the person issuing the CREATE USER MAPPING statement to the data source authorization ID specified in the REMOTE_AUTHID user option.

Testing the connection to the DB2 data source server

Test the connection to the DB2 data source server to determine if the federated server is properly configured to access DB2 data source.

About this task

You can test the connection to the DB2 server by using the server definition and user mappings that you defined.

Procedure

To test the connection to the DB2 server:

Open a pass-through session and issue an SQL SELECT statement on the DB2 system tables. If the SQL SELECT statement returns a count, your server definition and your user mapping are set up properly. If the SELECT statement returns an error, you should troubleshoot the connection errors.

DB2 family data source	Example

For DB2 for z/OS and OS/390	SET PASSTHRU *server_definition_name*
	SELECT count(*) FROM sysibm.systables
	SET PASSTHRU RESET

For DB2 for iSeries	SET PASSTHRU *server_definition_name*
	SELECT count(*) FROM qsys2.systables
	SET PASSTHRU RESET

Registering nicknames for DB2 tables and views

For each DB2 server definition that you register, you must register a nickname for each table or view that you want to access. Use these nicknames, instead of the names of the data source objects, when you query the DB2 servers.

Before you begin

Update the statistics at the DB2 data source before you register a nickname. The federated database relies on the data source catalog statistics to optimize query processing. Use the data source command that is equivalent to the DB2 RUNSTATS command to update the data source statistics.

Restrictions

You cannot create a nickname on a DB2 alias.

Procedure

To register a nickname for a DB2 table or view:

Issue the CREATE NICKNAME statement. Nicknames can be up to 128 characters in length.

For example:

```
CREATE NICKNAME nickname FOR server_definition_name."remote_schema"."remote.table"
```

When you create the nickname, the federated server queries the data source catalog using the nickname. This query tests the connection to the data source. If the connection does not work, you will receive an error message.

Repeat this step for each DB2 table or view that you want to create a nickname for.

CREATE NICKNAME statement - Examples for the DB2 wrapper

Use the CREATE NICKNAME statement to register a nickname for an DB2 table or view that you want to access. This topic includes a complete example with the required parameters.

Complete example

```
CREATE NICKNAME DB2SALES FOR DB2SERVER.VINNIE.EUROPE
```

DB2SALES

A unique nickname that is used to identify the DB2 table or view. The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authorization ID of the user who registers the nickname.

DB2SERVER.VINNIE.EUROPE

A three-part identifier for the remote object:

- *DB2SERVER* is the name that you assigned to the DB2 database server in the CREATE SERVER statement.
- *VINNIE* is the user ID of the owner to which the table or view belongs. This value is case sensitive.
- *EUROPE* is the name of the remote table or view that you want to access.

Chapter 8. Entrez data sources

Configuring access to Entrez data sources

You can integrate the data that is in the Entrez databases with information from other sources by using a federated system.

To configure a federated server to access Entrez data sources, you must provide the federated server with information about the data sources and objects that you want to access. After you configure the federated server, you can create queries and use the custom functions to access the Entrez data sources.

Entrez Wrapper

Entrez is a query and retrieval system developed by the National Center for Biotechnology Information (NCBI). You can use the Entrez wrapper to access several linked databases hosted by the NCBI.

You can access all of the Entrez databases through a uniform set of Web-based tools. The Entrez wrapper uses these tools to federate the Entrez databases.

Although the Entrez interface supports many databases, the Entrez wrapper supports access only to the following databases:

- Nucleotide. This is a sequence database also called GenBank.
- OMIM. The Online Mendelian Inheritance in Man database from John Hopkins University.
- PubMed. A database containing biomedical literature.

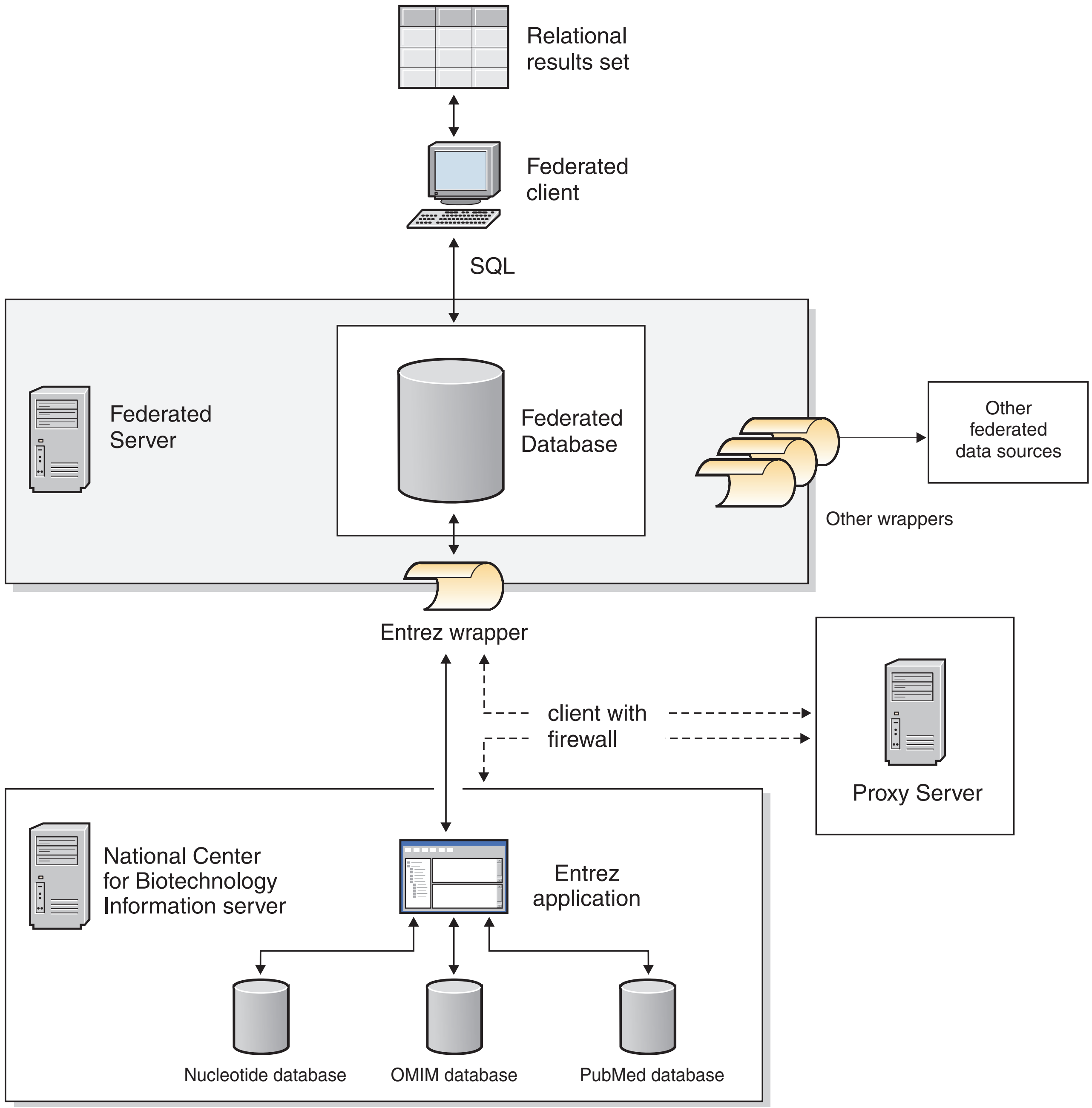


Figure 16. How the Entrez wrapper works

Many elements of the Entrez wrapper are common to all of the databases. These elements include:

- Connectivity with NCBI through the Web and the Entrez ESearch and EFetch utilities
- Mapping of hierarchical XML data into relational tables
- Joins between related tables through the XML wrapper technology

Adding Entrez data sources to a federated server

To configure a federated server to access Entrez data sources, you must register custom functions, a wrapper, a server definition, user mappings, and nicknames for the Entrez databases.

Before you begin

- IBM WebSphere Federation Server must be installed on a server that will act as the federated server
- A federated database must exist on the federated server

About this task

If your network uses a firewall, the Entrez wrapper can access the Nucleotide, OMIM, and PubMed databases if the network includes a proxy server. The Entrez wrapper supports the HTTP and SOCKS proxies.

You can configure a federated server to access data that is stored in Entrez data sources by using the Control Center or by issuing SQL statements on the command line. The Control Center includes a wizard to guide you through the steps that are necessary to configure the required federated objects.

Procedure

To add Entrez data sources to a federated server:

1. Register custom functions for the Entrez wrapper.
2. Register the Entrez wrapper.
3. Register the server definitions.
4. Register the user mappings (optional).
5. Register the nicknames.

Registering the custom functions for the Entrez wrapper

You must register the Entrez custom functions before you register the Entrez wrapper. The Entrez custom functions are used with the Entrez wrapper to send predicates to the Entrez query engine.

About this task

You must register all of the custom functions on each federated database instance where the Entrez wrapper is installed.

All of the custom functions for the Entrez wrapper must be registered with the schema name ENTREZ.

Tip: If you use the Federated Objects wizard in the Control Center to register the Entrez wrapper, server definitions, and nicknames, the Entrez custom functions are automatically created for you. Otherwise, you must register the custom functions from the command line.

Procedure

To register the Entrez custom functions, run the create_function_mappings.ddl file on each federated database instance where the Entrez wrapper is installed.

The create_function_mappings.ddl file is in the sqllib/samples/lifesci/entrez directory on the federated server. This file contains the CREATE FUNCTION statements for each of the custom functions for the Entrez wrapper.

The AS TEMPLATE, DETERMINISTIC, and NO EXTERNAL ACTION keywords are included in the CREATE FUNCTION statement in the create_function_mappings.ddl file.

Custom functions for the Entrez wrapper

You must include specific keywords when you register the Entrez custom functions

The federated environment uses two query engines. For the Entrez wrapper, these query engines are the federated database query engine and the Entrez query engine. You can specify that predicates should be sent to the Entrez engine by using the Entrez custom functions in the WHERE clause of your SELECT statement.

The custom functions for the Entrez wrapper are:

- ENTREZ.CONTAINS
- ENTREZ.SEARCH_TERM

A separate ENTREZ custom function must be registered for each valid data type for the column_name argument. The data types that you can specify for the ENTREZ custom functions are listed in the following table:

Table 28. Valid data types for the ENTREZ custom functions

ENTREZ.CONTAINS custom function	ENTREZ.SEARCH_TERM custom function
ENTREZ.CONTAINS (VARCHAR(), VARCHAR())	ENTREZ.SEARCH_TERM (CHAR(), VARCHAR())
ENTREZ.CONTAINS (INTEGER, VARCHAR())	
ENTREZ.CONTAINS (SMALLINT, VARCHAR())	
ENTREZ.CONTAINS (REAL, VARCHAR())	
ENTREZ.CONTAINS (DOUBLE, VARCHAR())	
ENTREZ.CONTAINS (DATE, VARCHAR())	
ENTREZ.CONTAINS (TIME, VARCHAR())	
ENTREZ.CONTAINS (CHAR(), VARCHAR())	
ENTREZ.CONTAINS (TIMESTAMP(), VARCHAR())	

ENTREZ.CONTAINS custom function

The following example shows the syntax for the ENTREZ.CONTAINS function:

```
CREATE FUNCTION ENTREZ.CONTAINS
(column_name_data_type, search_value_data_type)
    RETURNS INTEGER AS TEMPLATE
    DETERMINISTIC NO EXTERNAL ACTION;
```

ENTREZ.SEARCH_TERM custom function

The following example shows the syntax for the ENTREZ.SEARCH_TERM custom function:

```
CREATE FUNCTION ENTREZ.SEARCH_TERM
(parent_nickname_primary_key_data_type, search_value_data_type)
    RETURNS INTEGER AS TEMPLATE
    DETERMINISTIC NO EXTERNAL ACTION;
```

Registering the Entrez wrapper

You must register a wrapper to access the Entrez databases. Wrappers are used by federated servers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.

About this task

You can register a wrapper by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps that are necessary to register the wrapper.

If you use a proxy server and a keystore to access the Entrez files, you can specify the keystore information as options when you register the wrapper or server definition. If you specify the keystore information when you register the wrapper, the settings are used when you query any Entrez file. The wrapper settings are not used if you specify different settings when you register the server definitions.

Procedure

To register the Entrez wrapper:

Procedure

Choose the method that you want to use to register the Entrez wrapper:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.

From the command line	Issue the CREATE WRAPPER statement.
	`CREATE WRAPPER wrapper_name`
	`LIBRARY library_name`
	`OPTIONS(EMAIL 'email_address');`
	If you use a proxy server to access Entrez documents, the statement that you issue is:
	`CREATE WRAPPER wrapper_name`
	`LIBRARY library_name`
	`OPTIONS (EMAIL 'email_address',`
	`PROXY_TYPE 'type',`
	`PROXY_SERVER_NAME 'server_name',`
	`PROXY_SERVER_PORT 'port_number');`

You must specify the LIBRARY parameter in the CREATE WRAPPER statement. The name of the wrapper library file that you specify depends on the operating system of the federated server. See the list of Entrez wrapper library files for the correct library name to specify in the CREATE WRAPPER statement.

If you use a proxy server or a keystore to access Entrez files, you must specify several wrapper options when you register the Entrez wrapper.

You must specify an email address when you register the wrapper. The e-mail address that you specify is included with all queries and allows NCBI to contact you if there are problems, such as too many queries overloading the NCBI servers.

Entrez wrapper library files

The Entrez wrapper library files are added to the federated server when you install IBM WebSphere Federation Server.

When you install IBM WebSphere Federation Server, three library files are added to the default directory path for the Entrez wrapper. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2lsentrez.a, libdb2lsentrezF.a, and libdb2lsentrezU.a.

The default wrapper library file is libdb2lsentrez.a. The other two wrapper library files are used with specific wrapper options.

When you register the Entrez wrapper from the command line, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

The default directory paths and default wrapper library file names are listed in the following table.

Table 29. Locations and names for the Entrez wrapper library files

Operating system	Default directory path	Default wrapper library file
AIX	/usr/opt/`<install_path>`/lib32/ /usr/opt/`<install_path>`/lib64/	libdb2lsentrez.a
Linux	/opt/IBM/db2/`<install_path>`/lib32 /opt/IBM/db2/`<install_path>`/lib64	libdb2lsentrez.so
Solaris	/opt/IBM/db2/`<install_path>`/lib32 /opt/IBM/db2/`<install_path>`/lib64	libdb2lsentrez.so
Windows	%DB2PATH%\bin	db2lsentrez.dll

`<install_path>` is the directory path where IBM WebSphere Federation Server is installed on UNIX or Linux.

%DB2PATH% is the environment variable that is used to specify the directory path where IBM WebSphere Federation Server is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

CREATE WRAPPER statement - examples for the Entrez wrapper

Use the CREATE WRAPPER statement to register the Entrez wrapper. The examples show the parameters that are required to access Entrez documents with and without a proxy server.

Registering a wrapper

If you are not using a proxy server to access Entrez documents, the statement that you issue to register the wrapper is:

```
CREATE WRAPPER entrez_wrapper LIBRARY 'libdb2lsentrez.a';
```

entrez_wrapper
: A name that you assign to the Entrez wrapper. Duplicate wrapper names are not allowed.

LIBRARY *'libdb2lsentrez.a'*
: The name of the wrapper library file for federated servers that use AIX operating systems.

Registering a wrapper for an HTTP proxy server

To register a wrapper and specify an HTTP proxy server, use the following statement:

```
CREATE WRAPPER entrez_wrapper LIBRARY 'libdb2lsentrez.a'
    OPTIONS (PROXY_TYPE 'HTTP', PROXY_SERVER_NAME 'proxy_http',
        PROXY_SERVER_PORT '8080');
```

PROXY_TYPE *'HTTP'*
: Specifies the proxy type that is used to access the Internet when the federated server is behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy_http'*
: Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'8080'*
: Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Registering a wrapper for a SOCKS proxy server

To register a wrapper and specify a SOCKS proxy server without authentication information, use the following statement:

```
CREATE WRAPPER entrez_wrapper LIBRARY 'libdb2lsentrez.so'
    OPTIONS (PROXY_TYPE 'SOCKS', PROXY_SERVER_NAME 'proxy_socks',
        PROXY_SERVER_PORT '1081');
```

LIBRARY *'libdb2lsentrez.so'*
: The name of the wrapper library file for federated servers that use Linux and Solaris operating systems.

PROXY_TYPE *'SOCKS'*
: Specifies the proxy type that is used to access the Internet when the federated server behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy_socks'*
: Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'1081'*
: Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

CREATE SERVER statement - examples for the Entrez wrapper

Use the CREATE SERVER statement to register server definitions for the Entrez wrapper. This topic includes an example with the required parameters, and an example with the server options required for proxy servers.

Required parameters

The following example shows you how to register a server definition for a Entrez wrapper by issuing the CREATE SERVER statement:

```
CREATE SERVER nucleotide_server1
    TYPE Nucleotide
    WRAPPER entrez_wrapper;
```

nucleotide_server1
: A name that you assign to the Entrez server. Duplicate server definition names are not allowed.

TYPE *Nucleotide*
: Specifies the type of the data source. The valid values for server type are `Nucleotide`, `OMIM`, and `PubMed`. These values are case-insensitive.

WRAPPER *entrez_wrapper*
: Specifies the name of the wrapper that you registered in the CREATE WRAPPER statement.

Sample server definition for OMIM data sources

The following example shows you how to register a server definition for a Entrez wrapper by issuing the CREATE SERVER statement:

```
CREATE SERVER omim_server1
    TYPE OMIM
    WRAPPER entrez_wrapper;
```

omim_server1
: A name that you assign to the Entrez server. Duplicate server definition names are not allowed.

TYPE *OMIM*
: Specifies the type of the data source. The valid values for server type are `Nucleotide`, `OMIM`, and `PubMed`. These values are case-insensitive.

WRAPPER *entrez_wrapper*
: Specifies the name of the wrapper that you registered in the CREATE WRAPPER statement.

Proxy server example

You can access Entrez data sources through a proxy server. The following example registers a server definition for a proxy server that uses the SOCKS protocol with authentication information.

```
CREATE SERVER pubmed_server_s5
    TYPE PUBMED
    VERSION 1.0
    WRAPPER entrez_wrapper
    OPTIONS (PROXY_TYPE 'SOCKS', PROXY_SERVER_NAME 'proxy_5',
        PROXY_SERVER_PORT '1081', PROXY_AUTHID 'Khalid',
        PROXY_PASSWORD 'good2go');
```

pubmed_server_s5
A name that you assign to the Entrez server. Duplicate server definition names are not allowed.

TYPE *PUBMED*
Specifies the type of the data source. The valid values for the server type are `Nucleotide`, `OMIM`, and `PubMed`. These values are case-insensitive.

VERSION *1.0*
Specifies the version of the NCBI XML schema that you are using. This argument is optional. If you do not specify the version of the server, the default value is 1.0.

WRAPPER *entrez_wrapper*
Specifies the name of the wrapper that you registered in the CREATE WRAPPER statement.

PROXY_TYPE *'SOCKS'*
Specifies the type of proxy server. This value must be enclosed in single quotation marks.

PROXY_SERVER_NAME *'proxy_5'*
Specifies the name of the proxy server. This value must be enclosed in single quotation marks.

PROXY_SERVER_PORT *'1081'*
Specifies the number of the port for the proxy server. This value must be enclosed in single quotation marks.

PROXY_AUTHID *'Khalid'*
Specifies the user ID on the proxy server. This value must be enclosed in single quotation marks. This value is case sensitive.

PROXY_PASSWORD *'good2go'*
Specifies the password on the proxy server that is associated with *'Khalid'*. This value must be enclosed in single quotation marks. This value is case sensitive.

Access Entrez using a proxy server

If your network uses a proxy server, you must specify information about the proxy server when you register the wrapper or server definition for Entrez data sources.

The options that you specify depend on the type of proxy server that you want to access, and whether you are using Secure Socket Layer (SSL) or Transport Layer Security (TLS) protocols.

You can specify the proxy and SSL options when you register a wrapper or a server definition:

- If you specify these options when you register the wrapper, the nicknames that are associated with that wrapper will use the options that are set for the wrapper.
- If you specify these options when you register a server definition, the nicknames that are associated with that server definition will use the options that are set for the server definition.
- If you specify different values for these options when you register the wrapper and the server definition, the values that are set for the server definition take precedence over the values that are set for the wrapper.

Proxy server options

The following table lists the options that must you specify for each type of proxy server:

Table 30. Options that are required with proxy servers

Type of proxy server	Required wrapper or server options
HTTP or SOCKS without authentication	PROXY_TYPE PROXY_SERVER_NAME PROXY_SERVER_PORT
HTTP or SOCKS with authentication	PROXY_TYPE PROXY_SERVER_NAME PROXY_SERVER_PORT PROXY_AUTHID PROXY_PASSWORD

In addition to the required server options for proxy servers, you can also specify the SOCKET_TIMEOUT server option when you register a server definition. The SOCKET_TIMEOUT server option specifies the maximum amount of time, in minutes, that the federated server waits for the results from the proxy server. If you do not specify the SOCKET_TIMEOUT server option, there is no time limit. The federated server will wait indefinitely for the results from the proxy server.

Nicknames for the Entrez data sources

To access the data in the Entrez databases, you must register a fixed list of nicknames for each database that you want to access.

Each Entrez database has a fixed set of tables that are organized in a hierarchy. Each table has a fixed set of fields. One table, which is the parent of all of the other tables in the database, is called the *root* or parent table. All of the other tables in the database have a parent-child relationship that leads back to the root table.

The nicknames for the Entrez databases use a similar hierarchical structure and each nickname contains a fixed list of columns.

You can use the predefined set of names for the nicknames or you can specify alternative names for the nicknames.

Nicknames for the Nucleotide database

The fields in the Nucleotide database are mapped to a set of nicknames that are stored on the federated server.

When the Entrez wrapper sends your query to the Nucleotide database, the data that is returned from the Nucleotide database is in the XML format. The Entrez wrapper maps the hierarchical structure of the XML data to the Nucleotide nicknames.

You can use the default names for the nicknames

Each Nucleotide nickname represents a group of fields in the Nucleotide database. Each nickname contains a fixed list of columns and the nicknames are organized in a hierarchical relationship.

The top of the hierarchy of nicknames is the GBSeq nickname. The GBSeq nickname is the parent of all of the other nicknames and is called the *root nickname*. All of the other nicknames have parent-child relationships that lead back to the root nickname.

The following table shows the relationship between the GBSeq nickname and the other nicknames in the Nucleotide hierarchy of nicknames.

Table 31. The Nucleotide hierarchy of nicknames

Nickname	Parent
GBSeq	None. GBSeq is the root nickname.
GBFeatures	GBSeq
GBIntervals	GBFeatures
GBQualifiers	GBFeatures
GBReference	GBSeq

If you want to query a child nickname, you must create joins between each child nickname and the parent nickname.

Nicknames for the OMIM database

The fields in the OMIM database are mapped to a set of nicknames that are stored on the federated server.

When the Entrez wrapper sends your query to the OMIM database, the data that is returned from the OMIM database is in the XML format. The Entrez wrapper maps the hierarchical structure of the XML data to the OMIM nicknames.

Each OMIM nickname represents a group of fields in the OMIM database. Each nickname contains a fixed list of columns and the nicknames are organized in a hierarchical relationship.

The top of the hierarchy of nicknames is the OMIM nickname. The OMIM nickname is the parent of all of the other nicknames and is called the *root nickname*. All of the other nicknames have parent-child relationships that lead back to the root nickname.

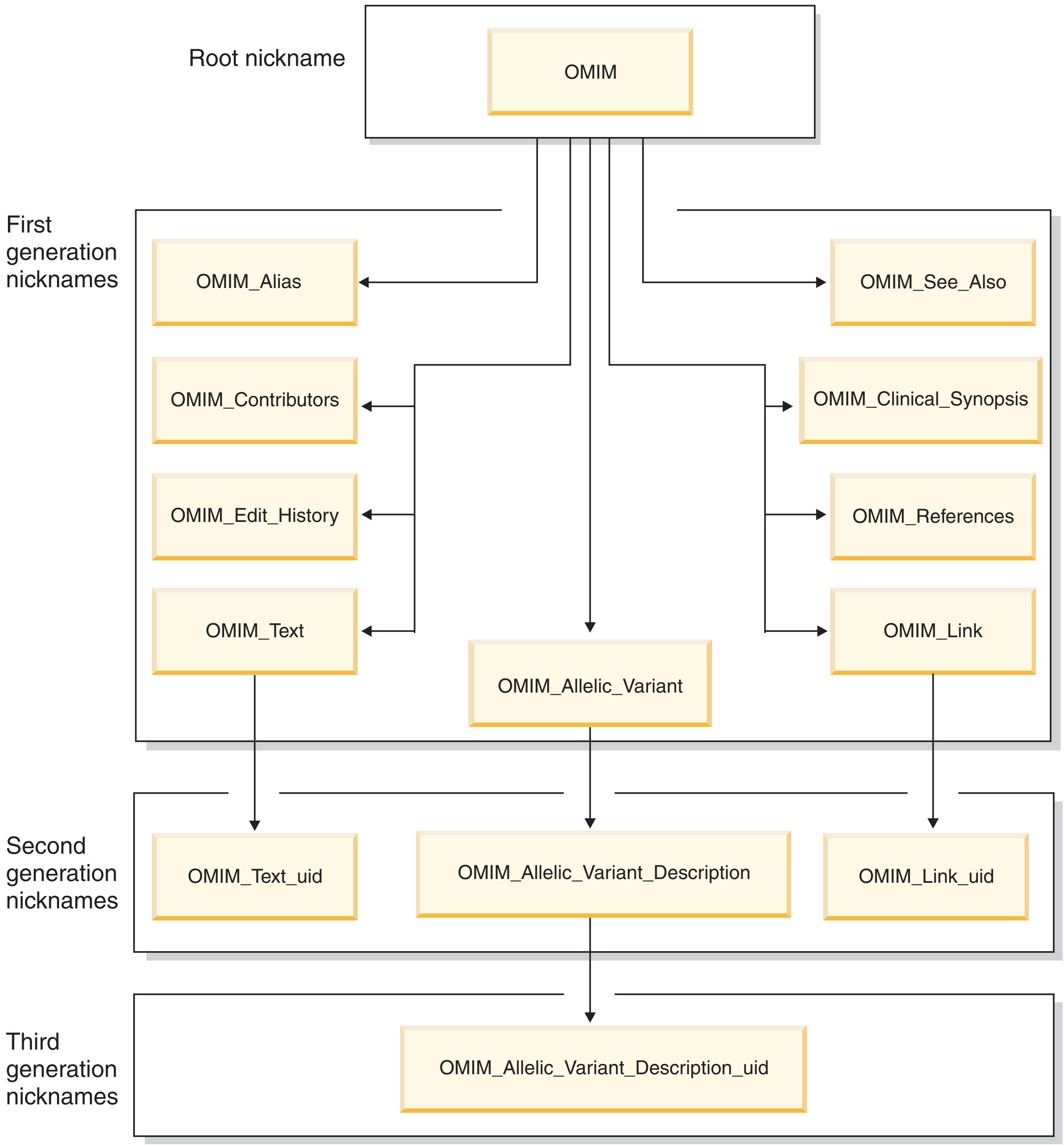


Figure 17. Hierarchy of OMIM nicknames

The following table shows the relationship between the OMIM nickname and the other nicknames in the OMIM hierarchy of nicknames.

Table 32. The OMIM hierarchy of nicknames

Nickname	Parent
OMIM	None. OMIM is the root nickname.
OMIM_Alias	OMIM
OMIM_Allelic_Variant	OMIM

Table 32. The OMIM hierarchy of nicknames (continued)

Nickname	Parent
OMIM_Allelic_Variant_Description	OMIM_Allelic_Variant
OMIM_Allelic_Variant_Description_UID	OMIM_Allelic_Variant_Description
OMIM_Clinical_Synopsis	OMIM
OMIM_Contributors	OMIM
OMIM_Edit_History	OMIM
OMIM_Links	OMIM
OMIM_Links_UID	OMIM_Links
OMIM_References	OMIM
OMIM_See_Also	OMIM
OMIM_Text	OMIM
OMIM_Text_UID	OMIM_Text

If you want to query a child nickname, you must create joins between each child nickname and the corresponding parent nickname, up to the root nickname.

For example, if you want to query the OMIM_Allelic_Variant_Description nickname, you must join the OMIM_Allelic_Variant_Description nickname (child) to the OMIM_Allelic_Variant nickname (parent) and join the OMIM_Allelic_Variant nickname (child) to the OMIM nickname (parent).

Nicknames for the PubMed database

The fields in the PubMed database are mapped to a set of nicknames that are stored on the federated server.

When the Entrez wrapper sends your query to the PubMed database, the data that is returned from the PubMed database is in the XML format. The Entrez wrapper maps the hierarchical structure of the XML data to the PubMed nicknames.

Each PubMed nickname represents a group of fields in the PubMed database. Each nickname contains a fixed list of columns and the nicknames are organized in a hierarchical relationship.

The top of the hierarchy of nicknames is the PMARTICLES nickname. The PMARTICLES nickname is the parent of all of the other nicknames and is called the *root nickname*. All of the other nicknames have parent-child relationships that lead back to the root nickname.

The following table shows the relationship between the PMARTICLES nickname and the other nicknames in the PubMed hierarchy of nicknames.

```
CREATE NICKNAME PMARTICLES FOR SERVER pubmed_server;
CREATE NICKNAME PMACCESSION FOR SERVER pubmed_server;
CREATE NICKNAME PMCHEMICAL FOR SERVER pubmed_server;
CREATE NICKNAME PMMESH FOR SERVER pubmed_server;
CREATE NICKNAME PMCOMMENTS FOR SERVER pubmed_server;
CREATE NICKNAME PMARTICLEID FOR SERVER pubmed_server;
```

Table 33. The PubMed hierarchy of nicknames

Nickname	Parent
PMARTICLES	None. PMARTICLES is the root nickname.

Table 33. The PubMed hierarchy of nicknames (continued)

Nickname	Parent
PMACCESSION	PMARTICLES
PMCHEMICAL	PMARTICLES
PMMESH	PMARTICLES
PMCOMMENTS	PMARTICLES
PMARTICLEID	PMARTICLES

If you want to query a child nickname, you must create joins between each child nickname and the parent nickname.

Alternative names for the Entrez nicknames

If the tables on your federated server use one or more of the predefined set of names for the Entrez nicknames, you must specify alternative names for the Entrez nicknames.

When you specify names for the Entrez nicknames other than the predefined set of names, you must include nickname options in the CREATE NICKNAME statements. These nickname options identify the predefined nickname that is associated with the nickname that are registering.

The REMOTE_OBJECT nickname option specifies the predefined name for the Entrez nickname. You must always include the REMOTE_OBJECT nickname option whenever you specify other names for the Entrez nicknames.

For the root nickname, you must specify only the REMOTE_OBJECT nickname option. For all other nicknames, you must specify both the REMOTE_OBJECT and PARENT nickname options.

Specifying an alternative name for the root nickname

For example, if a table on your federated server has the name OMIM, you need to register the root nickname OMIM with another name such as omim_nn:

```
CREATE NICKNAME omim_nn FOR SERVER omim_server1
   OPTIONS (REMOTE_OBJECT 'omim');
```

Specifying an alternative name for a child nickname

If a table on your federated server has the name OMIM_ALLELIC_VARIANT, you need to register the OMIM_ALLELIC_VARIANT nickname with another name such as omim_av_nn. You must identify the predefined name for the nickname in the REMOTE_OBJECT nickname option. You must also identify the parent nickname in the PARENT nickname option, which is the root nickname. For example:

```
CREATE NICKNAME omim_av_nn FOR SERVER omim_server1
   OPTIONS (REMOTE_OBJECT 'omim_allelic_variant', PARENT 'OMIM_NN');
```

You must always include the PARENT nickname option when the nickname that you are registering is a child nickname. The federated server changes the parent nickname to uppercase when you register the nickname, unless you enclose the name of the nickname in double quotation marks.

For example:

- You specify the name *omim_nn* when you register the parent nickname. You must specify *'OMIM_NN'* for the value of the PARENT nickname option when you register the child nickname.
- You specify the name *"omim_nn"* when you register the parent nickname. You must specify *'omim_nn'* for the value of the PARENT nickname option when you register the child nickname.

Specifying an alternative name for a child nickname that has a predefined parent nickname

If you used the predefined name when you registered a parent nickname, you specify the predefined name in the PARENT nickname option:

```
CREATE NICKNAME omim_av_nn FOR SERVER omim_server1
   OPTIONS (REMOTE_OBJECT 'omim_allelic_variant', PARENT 'OMIM');
```

Creating the user mappings for a Entrez data source (optional)

When you attempt to access a Entrez server, the federated server establishes a connection to the Entrez server. If your federated system uses a proxy server to access Entrez data sources, you must create user mappings.

A user mapping is an association between each federated server user ID and password, and the corresponding data source user ID and password.

About this task

There are two methods for specifying user mappings with federated systems. You can use an external repository, such as LDAP, to store the user mappings or you can create the user mappings in the federated database catalog.

If you have an external repository, such as LDAP, to store the user mappings, you do not need to create user mappings. You must specify the DB2_UM_PLUGIN option on the Entrez wrapper. You can specify this option when you register or alter the wrapper.

Procedure

To map a local user ID to the Entrez server user ID and password:

Choose the method that you want to use to register the user mappings:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.
Using the command line	Issue the CREATE USER MAPPING statement. For example: `CREATE USER MAPPING FOR local_userID SERVER server_definition_name OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password') PROXY_AUTHID 'proxy_server_userID', PROXY_PASSWORD 'proxy_server_password';`

If you specify authentication information for the proxy server when you register a server definition and a user mapping, the values that you specify when you register the user mapping take precedence.

For example, you have ten people in your organization and you specify authentication information when you register the server definition. You create user mappings for three of the ten people. When the three people access the federated system, the authentication information that you specified when you created the user mappings is used. For the remaining seven people, the authentication information that you specified when you registered the server definition is used.

CREATE USER MAPPING statement - examples for the Entrez wrapper

Use the CREATE USER MAPPING statement to map a federated server user ID to a Entrez server user ID and password.

You can create a user mapping by specifying a proxy server. The following example shows how to map a federated server user ID to a Entrez proxy server user ID and password:

```
CREATE USER MAPPING FOR lucille SERVER blast_proxy
      OPTIONS (PROXY_AUTHID 'lball'
      PROXY_PASSWORD 'them2us');
```

lucille Specifies the local user ID that you are mapping to a user ID that is defined at the Entrez proxy server.

SERVER *blast_proxy*
Specifies the server definition name that you registered in the CREATE SERVER statement for the Entrez server.

PROXY_AUTHID *'lball'*
Specifies the user ID on the proxy server. This user mapping option is required when the proxy server requires authentication.

PROXY_PASSWORD *'them2us'*
Specifies the password on the proxy server that is associated with the user name *'lball'*. This user mapping option is required when the proxy server requires authentication.

Registering nicknames for Entrez data sources

For each Entrez server definition that you register, you must register a nickname for each database that you want to access. Use these nicknames, instead of the names of the databases, when you query the Entrez servers.

About this task

The names that you give the nicknames can be up to 128 characters in length.

You can register a nickname by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps that are necessary to register the nickname.

Procedure

To register a nickname for an Entrez database:

Choose the method that you want to use to register the nickname:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard from the **Federated Database Objects** folder, or open the **Nicknames** folder. 1. Click **Discover** and use the Discover window to assist you in generating the nicknames for the Entrez database. 2. In the **Schema** field, specify the schema in which you want to register the nicknames. 3. You can use the predefined set of names for the nicknames or you can specify other names for the nicknames: • If you use the predefined set of names for the nicknames, a text string suffix is automatically added to the end of nicknames. You can replace the default suffix in the **Nickname suffix** field. • To specify other names for the nicknames, click the **Define nicknames** radio button and specify a nickname for each of the remote objects. You will also need to specify the required nickname options. For a root nickname, only the REMOTE_OBJECT nickname option is required. For all of the other nicknames, you must specify both the REMOTE_OBJECT and the PARENT nickname options. 4. Click **OK** to return to the Define Nicknames page in the Federated Objects wizard. The nicknames are listed in the **Nicknames** table. 5. Click **Finish** to return to the Create Nicknames window in the Federated Objects wizard. The nicknames are listed in the **Nicknames** table. If you did not already create the Entrez custom functions, the custom functions are created for you and appear on the Additional Objects page of the Federated Objects wizard. After you complete the wizard, the function templates are listed in the **User Defined Functions** folder that is under the **Application Objects** folder in the Control Center. 6. Click **Finish** to complete the wizard and create the nicknames.

Method	Procedure
Using the command line	Issue the CREATE NICKNAME statement. You can use the predefined set of names for the nicknames or you can specify alternative names for the Entrez nicknames: • To use the predefined set of names for the nicknames, copy the predefined CREATE NICKNAME statements for the Entrez database that you want to access to a command line prompt. Change the name of the server definition in each of the statements to quickly create the nicknames. • If the tables on your federated server use one or more of the predefined set of names for the Entrez nicknames, you must specify alternative names for the Entrez nicknames.

Repeat these steps for each Entrez database that you want to create nicknames for.

CREATE NICKNAME statement - examples for the Entrez wrapper

Use the CREATE NICKNAME statement to register the nicknames for each Entrez database that you want to access. There are a set of predefined nicknames that you create for the Entrez databases. You can use the default names for the nicknames or specify alternative names.

The predefined set of CREATE NICKNAME statements that are necessary to register the nicknames for each of the Entrez databases are specified in the following lists. You can edit these statements to specify the name of the server definition that you used in the CREATE SERVER statement. Issue these CREATE NICKNAME statements from the command line to register the nicknames.

If you use the Control Center to register the nicknames, the statements are automatically generated and issued for you.

Nucleotide nicknames

The following list contains the CREATE NICKNAME statements that are necessary to register the nicknames for the Nucleotide tables:

```
CREATE NICKNAME GBSeq FOR SERVER nuc1;
CREATE NICKNAME GBFeatures FOR SERVER nuc1;
CREATE NICKNAME GBIntervals FOR SERVER nuc1;
CREATE NICKNAME GBQualifiers FOR SERVER nuc1;
CREATE NICKNAME GBReference FOR SERVER nuc1;
```

OMIM nicknames

The following list contains the CREATE NICKNAME statements that are necessary to register the nicknames for the OMIM tables:

```
CREATE NICKNAME omim FOR SERVER omim_server1;
CREATE NICKNAME omim_alias FOR SERVER omim_server1;
CREATE NICKNAME omim_text FOR SERVER omim_server1;
CREATE NICKNAME omim_text_uid FOR SERVER omim_server1;
CREATE NICKNAME omim_see_also FOR SERVER omim_server1;
CREATE NICKNAME omim_allelic_variant FOR SERVER omim_server1;
CREATE NICKNAME omim_allelic_variant_desc FOR SERVER omim_server1;
CREATE NICKNAME omim_allelic_variant_desc_uid FOR SERVER omim_server1;
CREATE NICKNAME omim_clinical_synopsis FOR SERVER omim_server1;
CREATE NICKNAME omim_edit_history FOR SERVER omim_server1;
CREATE NICKNAME omim_references FOR SERVER omim_server1;
CREATE NICKNAME omim_contributors FOR SERVER omim_server1;
CREATE NICKNAME omim_links FOR SERVER omim_server1;
CREATE NICKNAME omim_links_uid FOR SERVER omim_server1;
```

PubMed nicknames

The following list contains the CREATE NICKNAME statements that are necessary to register the nicknames for the PubMed tables:

```
CREATE NICKNAME PMARTICLES FOR SERVER pubmed_server;
CREATE NICKNAME PMACCESSION FOR SERVER pubmed_server;
CREATE NICKNAME PMCHEMICAL FOR SERVER pubmed_server;
CREATE NICKNAME PMMESH FOR SERVER pubmed_server;
CREATE NICKNAME PMCOMMENTS FOR SERVER pubmed_server;
CREATE NICKNAME PMARTICLEID FOR SERVER pubmed_server;
```

Queries and custom functions for Entrez data sources

The Entrez wrapper includes custom functions that you can use in your queries.

Query the Entrez databases using the custom functions

The Entrez custom functions are used with the Entrez wrapper to send predicates to the Entrez query engine.

The custom functions that are provided with the Entrez wrapper must be registered on the federated server before you can use the functions in your queries.

ENTREZ.CONTAINS function

The ENTREZ.CONTAINS function requires two arguments. The first argument is the name of a fixed nickname column that can be searched. The second argument is the value that you want to search for.

The syntax for the ENTREZ.CONTAINS function is:

```
ENTREZ.CONTAINS (nickname_column,query_term)
```

In some situations, the predicates processed by the federated server and the predicates processed by the Entrez database server might be mixed in such a way that they cannot be processed. These cases generate the error message SQL0142N (″SQL statement not supported″).

For example, in the following query fragment you cannot separate the parts of the predicate that are processed by the wrapper (the ENTREZ.CONTAINS invocations) and the parts that must be processed by the federated server (the relational predicate on BaseCountA).

```
WHERE
   OMIM.CONTAINS (Organism, 'helicobacter pylori') = 1
   OR (BaseCountA > 10 AND OMIM.CONTAINS (Keywords, 'cellular vacuolation ') = 1)
```

The nicknames that you register for the Entrez databases have a set of fixed columns. The only column names that you can specify in the ENTREZ.CONTAINS functions are the column names that have a search tag qualifier. If you specify a column name that does not have a search tag qualifier, the error message SQL30090N (″Operation invalid for application execution environment″) is returned.

The value of the query term argument can be only a literal, a host variable, or a column reference. You cannot use arithmetic or string concatenation. The value of the query term argument cannot be NULL, even if the search term column that is used is defined as allowing null values. The query term argument is not case sensitive.

If you want to search multiple columns, you must include a separate ENTREZ.CONTAINS function in your query for each column that you want to search on. Use Boolean operators (OR and AND) and parentheses in your query to specify the relationship between each ENTREZ.CONTAINS function. Search term qualifiers, such as [pd], are not allowed.

The ENTREZ.CONTAINS function returns an integer result. When you use the ENTREZ.CONTAINS function, the return value must be compared to the value 1 by using the = or <> operators. For example:

```
SELECT * FROM NewGBSeq
    WHERE ENTREZ.CONTAINS (Organism, 'drosophila') = 1;
```

Pseudo columns

Some search fields do not have corresponding columns in the Entrez schema. For example, in the Nucleotide database, the term [ALL] searches all searchable fields, while [WORD] searches all of the free text associated with a record. The AllText and FreeText columns for the GBSeq nickname are examples of pseudo columns are provided for these search terms. If a pseudo column is referenced in a select list, a NULL value is returned.

The following example shows a query with a WHERE clause on a PubMed nickname. In this example, the individual predicates are `authors`, `title`, and `alltext`. The `alltext` pseudo column is used with the ENTREZ.CONTAINS function to search all searchable fields for the terms drosophila or fruit fly.

```
WHERE
    ENTREZ.CONTAINS (authors, 'kaufmann OR ito AND NOT rakesh')
    AND
  (ENTREZ.CONTAINS (title, 'drosophila')
    OR
    ENTREZ.CONTAINS (alltext, 'drosophila OR "fruit fly"'))
```

ENTREZ.SEARCH_TERM custom function

The ENTREZ.SEARCH_TERM function requires two arguments. The first argument is the name of a fixed nickname column that is the primary key column for the parent nickname. The second argument is the value that you want to search for.

The syntax for the ENTREZ.SEARCH_TERM function is:

```
ENTREZ.SEARCH_TERM (nickname_column,query_term)
```

You can run queries that might not otherwise be possible by issuing the ENTREZ.SEARCH_TERM custom function. When you use the ENTREZ.SEARCH_TERM function, it must be the only custom function in a query.

The nickname column argument must be the primary key column for the parent nickname. The query term argument is an ENTREZ-format search term that includes search field qualifiers. This text is passed unmodified, except for URI escapes as required by the URI syntax, to ENTREZ.

The ENTREZ.SEARCH_TERM function returns an integer result. When you use the ENTREZ.SEARCH_TERM custom function, the return value must be compared to the value 1 by using the equal (=) or not equal (<>) operators. For example:

```
SELECT mim_id, SUBSTR(title,1,40) FROM omim
   WHERE entrez.search_term(mim_id, 'CYSTIC FIBROSIS') = 1;
```

Relational predicates for the Entrez wrapper

The Entrez wrapper supports relational predicates in queries. Some of the predicates are processed by the Entrez search engine and some are processed by the federated server.

The Entrez search engine processes equal and IN predicates on the ID column for each schema. These predicates allow the Entrez wrapper to bypass the search phase of a query process and execute the fetch phase of a query process. Examples of valid predicates are:

```
WHERE pmid = '1234567'
```

```
WHERE medlineid IN ('1234567', '9191919')
```

```
WHERE mim_id IN ('602421', '608583')
```

You can specify only the columns that have ID as the search tag for the nickname column in the WHERE clause of your query.

Invalid WHERE clauses for the Entrez wrapper

The Entrez wrapper rejects any query that will result in an unqualified scan of the NCBI database. A valid WHERE clause must contain either a custom function, or an equal or IN predicate on the primary ID for the schema. Queries that do not meet these criteria are rejected with error code SQL0142N or SQL30090N.

Records returned from Entrez queries

The MAX_ROWS server option can be used to limit the number of rows that are returned from a query on the Entrez data sources.

The MAX_ROWS server option value is always used as an upper (maximum) limit to the number of records for the root nickname that a query can return. If a query attempts to retrieve more records than what is specified in the MAX_ROWS server option, the result set is truncated. You are not notified that the result set is truncated.

For example, if you set the MAX_ROWS server option to 25, a maximum of 25 records for the root nickname and all of the records for the child-related nicknames are returned.

You can set the MAX_ROWS server option when you register a server definition with the CREATE SERVER statement, or you can use the ALTER SERVER statement to set or change the option value.

The MAX_ROWS server option is not required. If you do not set this option, a default value is used. The specific default value that is used depends on your operating system.

- For federated servers that run UNIX, the default is 5000 records.
- For federated servers that run Windows, the default is 2000 records.

You can specify only positive numbers and 0. When you set the MAX ROWS server option to 0, you enable queries to retrieve an unlimited number of records for the root nickname from the NCBI Web site. However, setting the MAX_ROWS server option to 0 or to a very high number might impact your query performance.

Entrez data sources - Example queries

These examples show you how to query the Nucleotide, OMIM, and PubMed databases.

Queries for Nucleotide nicknames

The following query uses multiple fetch keys on a Nucleotide nickname:

```
SELECT PrimaryAccession, LocusName, SeqLength FROM GBSEQ
   WHERE PrimaryAccession IN ('NM_000890', 'NC_003106');
```

The following query searches for all of the searchable fields on a Nucleotide nickname:

```
SELECT PrimaryAccession, substr(Definition,1,300), GI FROM GBSEQ
   WHERE ENTREZ.CONTAINS(AllText, 'abcde')=1;
```

The following query searches for all of the free text on a Nucleotide nickname:

```
SELECT * FROM GBSEQ WHERE ENTREZ.CONTAINS(FreeText, 'abcde')=1;
```

The following query searches for a definition on a Nucleotide nickname:

```
SELECT PrimaryAccession, substr(Definition,1,300), version, GI FROM GBSEQ
    WHERE ENTREZ.CONTAINS(Definition, 'Sulfolobus tokodaii
   AND complete genome') = 1;
```

The following query searches for a keyword on a Nucleotide nickname:

```
SELECT PrimaryAccession, substr(KeywordList,1,200), Segment FROM GBSEQ`
   WHERE ENTREZ.CONTAINS(KeywordList, 'nkcc1 gene') = 1;
```

Queries for OMIM nicknames

The following query returns the MIM identification number and the titles (up to 40 characters) of the OMIM records from the OMIM nickname, for a single MIM number:

```
SELECT mim_id, SUBSTR(title,1,40) FROM OMIM
   WHERE mim_id = '603855';
```

The following query returns the MIM identification numbers and the titles (up to 40 characters) of the OMIM records from the OMIM nickname, for multiple MIM identification numbers:

```
SELECT mim_id, SUBSTR(title,1,40) FROM OMIM
   WHERE mim_id IN ('603855', '605382', '600737', '605637', '219700');
```

The following query uses a CONTAINS function to look up all of the OMIM entries that contain CYSTIC FIBROSIS or CF in the title of the entry:

```
SELECT mim_id, SUBSTR(title,1,40) FROM OMIM
   WHERE OMIM.CONTAINS(title, 'CYSTIC FIBROSIS; CF') = 1;
```

The following query uses two CONTAINS functions to look up all of the OMIM entries that contain DIABETES in the title and that have a clinical synopsis:

```
SELECT mim_id, SUBSTR(title,1,75), has_synopsis FROM OMIM
   WHERE OMIM.CONTAINS(title, 'DIABETES') = 1
   AND OMIM.CONTAINS(has_synopsis, 'Clinical Synopsis') = 1;
```

The following query uses a CONTAINS function to look up all of the OMIM entries that were created on 5 May 1999:

```
SELECT mim_id, SUBSTR(title,1,40), creation_date FROM OMIM
   WHERE OMIM.CONTAINS(creation_date, '5/27/1999') = 1;
```

The following query uses a CONTAINS function to look up all of the OMIM entries that were created in 2005 and contain HEMERALOPIA-MYOPIA in the title:

```
SELECT mim_id, SUBSTR(title,1,40), creation_date FROM OMIM
   WHERE OMIM.CONTAINS(creation_date, '2005') = 1
   AND OMIM.CONTAINS(title, 'HEMERALOPIA-MYOPIA') = 1;
```

The following query uses a SEARCH_TERM function to look up all of the OMIM entries that contain HEMERALOPIA-MYOPIA in one of their fields:

```
SELECT mim_id, SUBSTR(title,1,40) FROM OMIM
   WHERE OMIM.SEARCH_TERM(mim_id, 'HEMERALOPIA-MYOPIA') = 1;
```

The following query uses a CONTAINS function to look up all of the OMIM entries that contain ″CYSTIC FIBROSIS in the title of the entry. This example shows how to SELECT values from nicknames in the OMIM hierarchy by joining the child nickname primary key and foreign key with the parent nickname:

```
SELECT o.mim_id, SUBSTR(o.title,1,20), SUBSTR(ot.text,1,40)
FROM omim o, omim_text ot
   WHERE OMIM.CONTAINS(o.title, 'CYSTIC FIBROSIS') = 1
   AND o.mim_id = ot.mim_id;
```

Queries for PubMed nicknames

The following query contains a single fetch key on a PubMed nickname:

```
SELECT PMID, ArticleTitle FROM pmarticles
    WHERE pmid = '12345';
```

The following query contains mixed fetch keys on a PubMed nickname:

```
SELECT PMID, ArticleTitle FROM pmarticles
    WHERE pmid = '12345' OR MedlineID = '12346';
```

The following query uses a CONTAINS function on a PubMed nickname:

```
SELECT PMID, ArticleTitle FROM pmarticles
   WHERE ENTREZ.CONTAINS (ArticleTitle, 'granulation') = 1
   AND ENTREZ.CONTAINS (PubDate, '1992') = 1;
```

The following query searches for the specified AuthorList and LanguageList on a PubMed nickname:

```
SELECT PMID, ArticleTitle FROM pmarticles
   WHERE ENTREZ.CONTAINS (AuthorList, 'Albarrak') = 1
   AND ENTREZ.CONTAINS (LanguageList, 'eng')=1;
```

The following query uses a complex predicate on a PubMed nickname:

```
SELECT PMID, ArticleTitle FROM pmarticles
   WHERE ENTREZ.CONTAINS (PublicationTypeList, 'Journal Article') = 1
   AND ENTREZ.CONTAINS (MedlineTA, 'sun')=1
   OR ENTREZ.CONTAINS (PersonalNameSubjectList, 'shine')=1;
```

Fixed columns for the Nucleotide nicknames

When you register the Nucleotide nicknames, information about the fields in the Nucleotide database is added to the federated database catalog. The information that is added to the catalog is represented as a set of fixed columns for each nickname.

List of the Nucleotide nicknames

The Nucleotide nicknames are:

- GBSeq
- GBReference
- GBFeatures
- GBIntervals
- GBQualifiers

The following tables provide information about the columns in each nickname:

- The name of the column.
- The data type for the column. You can override the default data type for a column when you create a nickname. For example, the default data type for the Sequence column in the GBSeq nickname is VARCHAR(32000). You can change this data type to CLOB(1 MB). The Entrez wrapper supports the CLOB data type, up to 5 megabytes in length.
- The description for the column.
- The search tag qualifier that is used by the wrapper to query the Nucleotide database. The only column names that you can specify in the ENTREZ.CONTAINS functions are the column names that have a search tag qualifier. The Qualifier column contains the valid search qualifiers for the column. For a list of valid search tags, see the following NCBI web site and locate the link to Search Field Descriptions and Qualifiers: www.ncbi.nlm.nih.gov/entrez/query/static/help/Summary_Matrices.html
- If the column can return a null value. Most of the columns that return a null value are input columns.

GBSeq nickname fixed columns

The columns in the GBSeq nickname are described in the following table. The F column indicates columns that are designated fetch keys. Using the fetch keys might expedite query processing.

Table 34. Nucleotide GBSeq nickname

Column name	Data type	Description	Qualifier	Fetch key
PrimaryAccession	VARCHAR(16) NOT NULL	Primary accession number	PACC	Yes

Table 34. Nucleotide GBSeq nickname (continued)

Column name	Data type	Description	Qualifier	Fetch key
SequenceKey	VARCHAR(32) NOT NULL	The primary key column used to join the GBSeq nickname with child nicknames.		No
LocusName	VARCHAR(16) NOT NULL	The name of the locus.	ACCN	No
SeqLength	INTEGER NOT NULL	The length of the sequence.	SLEN	No
Strandedness	VARCHAR(32)	not-set, single-stranded, double-stranded, mixed-stranded	None	No
MoleculeType	VARCHAR(16)	nucleic-acid, dna, rna, trna, rrna, mrna, urna, snrna, snorna, peptide	PROP	No
Topology	VARCHAR(16)	linear, circular	None	No
Division	CHAR(3) NOT NULL	The GenBank division.	PROP	No
UpdateDate	DATE NOT NULL	The date of the most recent update.	MDAT	No
CreateDate	DATE NOT NULL	The date that the record was created.	None	No
Definition	VARCHAR(7000) NOT NULL	The definition line of the sequence.	TITL	No
Version	INTEGER	The version ID of the sequence.	None	No
GI	VARCHAR(16)	The GenInfo (GI) sequence ID.	None	No
KeywordList	VARCHAR(7000)	A semicolon separated list	KYWD	No
Segment	VARCHAR(250)	The segment.	None	No
Source	VARCHAR(200) NOT NULL	The source.	ORGN	No
Organism	VARCHAR(7000) NOT NULL	The organism.	ORGN	No
Taxonomy	VARCHAR(7000) NOT NULL	The taxonomy.	None	No
Comment	VARCHAR(7000)	Comments	None	No
Primary	VARCHAR(7000)	The primary.	None	No
SourceDB	VARCHAR(250)	The source database.	None	No
Sequence	VARCHAR(32000)	The sequence.	None	No
AllText	VARCHAR(1)	A pseudo-column that always returns NULL	ALL	No
FreeText	VARCHAR(1)	A pseudo-column that always returns NULL	WORD	No

GBReference nickname fixed columns

The columns in the GBReference nickname are described in the following table.

Table 35. Nucleotide GBReference nickname

Column name	Data type	Description	Qualifier
SequenceKey	VARCHAR(32) NOT NULL	The key that is used to join a GBReference child nickname with its parent nickname.	None
ReferenceNum	INTEGER NOT NULL	Parsed from GBReference_reference	None
RangeLow	INTEGER NOT NULL	Low base for reference (parsed from GBReference_reference)	None
RangeHigh	INTEGER NOT NULL	High base for reference (parsed from GBReference_reference)	None
Authors	VARCHAR(3200)	A semicolon-separated list of names in GenBank form	AUTH
Consortium	VARCHAR(250)	The consortium.	None
Title	VARCHAR(250)	The GenBank reference title.	WORD
Journal_Title	VARCHAR(250) NOT NULL	The title of the journal.	JOUR
MedlineID	INTEGER	The Medline ID	None
PubMedID	INTEGER	The PubMed ID	None
Remarks	VARCHAR(3200)	Remarks	None

GBFeatures nickname fixed columns

The columns in the GBFeatures nickname are described in the following table.

Table 36. Nucleotide GBFeatures nickname

Column name	Data type	Description	Qualifier
SequenceKey	VARCHAR(32) NOT NULL	The key that is used to join a GBFeatures child nickname with its parent nickname.	None
FeatureJoinKey	VARCHAR(32) NOT NULL	The primary key column used to join the GBFeatures nickname with child nicknames.	None
FeatureKey	VARCHAR(20) NOT NULL		FKEY
FeatureLocation	VARCHAR(200) NOT NULL		None

GBIntervals nickname fixed columns

The columns in the GBIntervals nickname are described in the following table.

Table 37. Nucleotide GBIntervals nickname

Column Name	Data type	Description	Qualifier
FeatureJoinKey	VARCHAR(32) NOT NULL	The key that is used to join a GBIntervals child nickname with its parent nickname.	None
IntervalFrom	INTEGER		None
IntervalTo	INTEGER		None
IntervalPoint	INTEGER		None
IntervalAccession	VARCHAR(32) NOT NULL		None

GBQualifiers nickname fixed columns

The columns in the GBQualifiers nickname are described in the following table.

Table 38. Nucleotide GBQualifiers nickname

Column name	Data type	Description	Qualifier
FeatureJoinKey	VARCHAR(32) NOT NULL	The key that is used to join a GBQualifiers child nickname with its parent nickname.	None
QualifierName	VARCHAR(50)	The name of the qualifier	None
QualifierValue	VARCHAR(32000)	The value of the qualifier	None

Fixed columns for the OMIM nicknames

When you register the OMIM nicknames, information about the fields in the OMIM database is added to the federated database catalog. The information that is added to the catalog is represented as a set of fixed columns for each nickname.

You can retrieve data from any of the fixed columns by including the column name in your SELECT statement. However, the columns that have search tags are the only columns that can be specified in the WHERE clause of a query.

List of the OMIM nicknames

The OMIM nicknames are:

- OMIM
- OMIM_Alias
- OMIM_Text
- OMIM_Text_uid
- OMIM_See_Also
- OMIM_Allelic_Variant
- OMIM_Allelic_Variant_Description
- OMIM_Allelic_Variant_Description_uid
- OMIM_Clinical_Synopsis
- OMIM_Edit_History

- OMIM_References
- OMIM_Contributors
- OMIM_Links
- OMIM_Links_uid

The following tables provide information about the columns in each nickname:

- The name of the column.
- The data type for the column.
- Whether the column can return a null value. Most of the columns that return a null value are input columns.
- The search tag that is used by the wrapper to query the OMIM database. The column names that have a search tag are the only column names that you can specify in the OMIM.CONTAINS functions.
- The description for the column.

OMIM nickname fixed columns

The columns for the OMIM nickname are described in the following table.

Table 39. Columns for the OMIM nickname

Column name	Data type	Null	Search tag	Description
MIM_ID	VARCHAR(6)	N	ID	A six-digit number that uniquely identifies an OMIM database entry. A primary key column that is defined as a FOR BIT DATA column. The OMIM database entry numbering system is described at the following Web site: OMIM Frequently Asked Questions (FAQs)
All_Fields	VARCHAR (1)	Y	ALL	An input column that is used to query all of the searchable columns in the OMIM database.
Chromosome	VARCHAR (1)	Y	CH	An input column that is used to specify the chromosome to which a gene or disorder is mapped in the OMIM Gene Map or Morbid Map.
Copyright	VARCHAR(60)	Y		The copyright statement for an OMIM database entry.
Created_By	VARCHAR (50)	Y		The full name of the person who created the OMIM database entry.
EC_RN_Number	VARCHAR (1)	Y	EC	An input column that is used to specify the number that designates a particular enzyme or chemical. The numbers for enzymes are assigned by the Enzyme Commission. The numbers for chemicals are assigned by the Chemical Abstract Service (CAS).
Filter	VARCHAR (1)	Y	FI	An input column that is used to retrieve records that contain crosslinks to other Entrez databases and LinkOuts to external (non-Entrez) resources. For example, ″omim protein″, ″omim structure″, ″loprovlocuslink″.
Creation_Date	DATE	Y	CD	The date on which an OMIM database entry was created. The format must be in a format that is supported by the federated database, such as MM/DD/YYYY, DD.MM.YYYY, or YYYY-MM-DD.
Gene_Map	VARCHAR (1)	Y	GM	An input column that is used to search the cytogenetic map location that is represented in the OMIM Gene Map.
Gene_Map_Disorder	VARCHAR (1)	Y	DI	An input column that is used to search for text that appears in the Disorder column of the OMIM Gene Map.

Table 39. Columns for the OMIM nickname (continued)

Column name	Data type	Null	Search tag	Description
Gene_Map_Name	VARCHAR (1)	Y	GN	An input column that is used to search for the official gene symbol and alternate gene symbols that are associated with an OMIM database entry.
Has_Gm_Locus	VARCHAR (1)	Y	PR	An input column that is used to search for the OMIM database entries that have a gene map locus. For example: `OMIM.CONTAINS (has_gm_locus,'has locus')`
Has_Summary	CHAR(1)	Y		Specifies if an OMIM database entry has a summary. Valid values for this column are Y and N.
Has_Synopsis	CHAR(1)	Y	PR	A column that is used to search for the OMIM database entries that have a synopsis. For example: `OMIM.CONTAINS (has_synopsis,'clinical synopsis')`
Has_Variants	VARCHAR (1)	Y	PR	An input column that is used to search for OMIM database entries that have allelic variants. For example: `OMIM.CONTAINS (has_variants,'allelic variants')`
Locus	VARCHAR(10)	Y		The position that a given gene occupies on a chromosome.
MIM_ID_Display	VARCHAR(7)	N		Displays the six-digit MIM_ID with the symbol corresponding to the MIM_type_value. For example: *605491, %608583. See the following Web site for more information about the symbols that precede the MIM_ID numbers. OMIM Frequently Asked Questions (FAQs)
Number_Gene_Maps	INTEGER	Y		The number of gene maps.
Symbol	VARCHAR(10)	Y		The symbol for the OMIM database entry.
Text	VARCHAR(1000)	Y		A field in an OMIM database entry that contains information about the entry.
Title	VARCHAR(250)	Y	TI	The title for the OMIM database entry.

OMIM_Alias nickname fixed columns

The columns for the OMIM_Alias nickname are described in the following table.

Table 40. Columns for the OMIM_Alias nickname

Column name	Data type	Null	Search tag	Description
MIM_ID	VARCHAR(6)	N		A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.
Alias	VARCHAR (100)	Y		Alternative titles and symbols for the OMIM database entry that are separated by semicolons.

OMIM_Text nickname fixed columns

The columns for the OMIM_Text nickname are described in the following table.

Table 41. Columns for the OMIM_Text nickname

Column name	Data type	Null	Search tag	Description
MIM_ID	VARCHAR(6)	N	ID	A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.
Neighbor_Link_uids	VARCHAR (300)	Y		A list of unique IDs that are separated by commas.
Text	VARCHAR (3000)	N	TXT	The content of the text field.
Text_Join_Key	VARCHAR (32)	N		The primary key column for the OMIM_Text nickname. This column contains a unique ID that is generated by the wrapper. This column is defined as a FOR BIT DATA column.
Text_Label	VARCHAR (50)	N		The label for the text field. For example: Description, Clinical Features, or Cloning.

OMIM_Text_uid nickname fixed columns

The columns for the OMIM_Text_uid nickname are described in the following table.

Table 42. Columns for the OMIM_Text_uid nickname

Column name	Data type	Null	Search tag	Description
Text_Join_Key	VARCHAR (32)	N		A foreign key that is used in a join condition with the parent nickname, OMIM_Text. This column is defined as a FOR BIT DATA column.
Uid	VARCHAR (10)	N		The unique ID that is associated with the Allelic_Join_Key for the OMIM_Text_uid nickname.

OMIM_See_Also nickname fixed columns

The columns for the OMIM_See_Also nickname are described in the following table.

Table 43. Columns for the OMIM_See_Also nickname

Column name	Data type	Null	Search tag	Description
MIM_ID	VARCHAR(6)	N		A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.
Author	VARCHAR (50)	N		The last name of the primary author of the OMIM database entry.
Citation_Number	INTEGER	N		The citation number in the reference section of the OMIM database entry.
Others	VARCHAR (50)	Y		The last name of the second author if the OMIM database entry is authored by two authors. If there are more than two authors, the phrase *et al* is returned from the OMIM database entry. If there is only one author for the OMIM database entry, this column returns a null value.

Table 43. Columns for the OMIM_See_Also nickname (continued)

Column name	Data type	Null	Search tag	Description
Year	CHAR (4)	Y		The year that the OMIM database entry is published. The format for this value is YYYY.

OMIM_Allelic_Variant nickname fixed columns

The columns for the OMIM_Allelic_Variant nickname are described in the following table.

Table 44. Columns for the OMIM_Allelic_Variant nickname

Column name	Data type	Null	Search tag	Description
MIM_ID	VARCHAR(6)	N	ID	A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.
Allelic_Join_Key	VARCHAR (32)	N		The primary key column for the OMIM_Allelic_Variant nickname. This column contains a unique ID that is generated by the wrapper. This column is defined as a FOR BIT DATA column.
Aliases	VARCHAR (100)	Y		A list of Allelic Variant aliases. The entries in the list are separated by commas.
Mutation	VARCHAR (40)	N	AV	The mutation of the Allelic Variant.
Name	VARCHAR (100)	N	AV	The name of the Allelic Variant.
Number	CHAR (4)	N		The number of the Allelic Variant. The Allelic Variant numbers are four digit numbers that start with 0001.

OMIM_Allelic_Variant_Description nickname fixed columns

The columns for the OMIM_Allelic_Variant_Description nickname are described in the following table.

Table 45. Columns for the OMIM_Allelic_Variant_Description nickname

Column name	Data type	Null	Search tag	Description
Allelic_Join_Key	VARCHAR (32)	N		A foreign key that is used in a join condition with the parent nickname, OMIM_Allelic_Variant. This column is defined as a FOR BIT DATA column.
Allelic_Description_Join_Key	VARCHAR (32)	N		The primary key column for the OMIM_Allelic_Variant_Description nickname. This column contains a unique ID that is generated by the wrapper. This column is defined as a FOR BIT DATA column.
Description	VARCHAR (3000)	N	AV	The description of the OMIM Allelic Variant.
Neighbor_Link_uids	VARCHAR (300)	Y		A list of unique IDs that are separated by commas.

OMIM_Allelic_Variant_Description_uid nickname fixed columns

The columns for the OMIM_Allelic_Variant_Description_uid nickname are described in the following table.

Table 46. Columns for the OMIM_Allelic_Variant_Description_uid nickname

Column name	Data type	Null	Search tag	Description
Allelic_Description_Join_Key	VARCHAR (32)	N		A foreign key that is used in a join condition with the parent nickname, OMIM_Allelic_Variant_Description. This column is defined as a FOR BIT DATA column.
Uid	VARCHAR (10)	N		The unique ID that is associated with the Allelic_Join_Key for the OMIM_Allelic_Variant_Description_uid nickname.

OMIM_Clinical_Synopsis nickname fixed columns

The columns for the OMIM_Clinical_Synopsis nickname are described in the following table.

Table 47. Columns for the OMIM_Clinical_Synopsis nickname

Column name	Data type	Null	Search tag	Description
MIM_ID	VARCHAR(6)	N		A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.
Author	VARCHAR (40)	Y		The full name of the author of the synopsis.
Creation_Date	DATE	Y		The date that the synopsis was created.
Synopsis_Key	VARCHAR (100)	Y		The clinical synopsis key. For example: Inheritance, Cardiovascular, Respiratory, Growth.
Synopsis_Terms	VARCHAR (500)	Y	CS	A list of terms that are used in the clinical synopsis. The list is separated by semicolons.

OMIM_Edit_History nickname fixed columns

The columns for the OMIM_Edit_History nickname are described in the following table.

Table 48. Columns for the OMIM_Edit_History nickname

Column name	Data type	Null	Search tag	Description
MIM_ID	VARCHAR(6)	N		A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.
Editor_ID	VARCHAR (20)	N	ED	The ID for the editor of the OMIM database entry.
Date	DATE	N	MD	The date that the OMIM database entry was edited. You must specify the date in a format that is supported by the federated database, such as MM/DD/YYYY, DD.MM.YYYY, or YYYY-MM-DD.

OMIM_References nickname fixed columns

The columns for the OMIM_References nickname are described in the following table.

Table 49. Columns for the OMIM_References nickname

Column name	Data type	Null	Search tag	Description
MIM_ID	VARCHAR(6)	N		A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.
Authors_List	VARCHAR (1000)	Y	RE	The list of authors for the reference. The names are separated by a semicolon and use the format Last Name, First/Middle initials. For example: Stewart, L.K.; Wamhoff, J.
Ambiguous	CHAR(1)	Y		Specifies whether the reference is ambiguous. Valid values for this column are Y and N.
Citation_Title	VARCHAR (100)	Y	RE	The titles of the references that are cited in the OMIM database entry.
Citation_Type	VARCHAR(10)	Y		The type of citation.
Journal	VARCHAR(50)	Y		Journal
No_Link	CHAR(1)	Y		Specifies if an OMIM database entry has links to other references. Valid values for this column are Y and N.
Other_Authors	VARCHAR (20)	Y	RE	The names of other authors that are cited in the reference. The last name of the second author if the reference is authored by two authors. If there are more than two authors, the phrase *et al* is returned from the OMIM database entry. If there is only one author for the reference, this column returns a null value.
Page_From	INTEGER	Y		The first page number in the reference where the article begins.
Page_To	INTEGER	Y		The last page number in the reference where the article ends. If the article is contained on one page, this column returns a null value.
Publication_Date	DATE	Y		The date that the reference was published. If the day or the month are not present, the day or month is set to 1. For example, if the entry in the OMIM database is April 2004, then the date 2004/04/01 is returned.
Pubmed_ID	VARCHAR (10)	Y		The Pubmed ID (PMID). If no PMID is available, this column returns ’0’.
Primary_Author	VARCHAR (20)	Y	RE	The last name of the primary author for the reference that is cited.
Ref_Orig_Number	INTEGER	Y		The original reference number for the OMIM database entry.
Ref_Number	INTEGER	N		The current reference number for the OMIM database entry.
Type	VARCHAR(20)	Y		The type of reference. For example citation, book, personal communication.
Volume	VARCHAR (10)	Y		The volume for the reference that is cited.

OMIM_Contributors nickname fixed columns

The columns for the OMIM_Contributors nickname are described in the following table.

Table 50. Columns for the OMIM_Contributors nickname

Column name	Data type	Null	Search tag	Description
MIM_ID	VARCHAR(6)	N		A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.
Action	VARCHAR (10)	Y		The action that was performed on a record. For example: updated and recognized. The action is parsed from the Author and Action fields.
Author	VARCHAR (40)	N	AU	The name of the contributor name. This name is parsed from the Author and Action fields.
Date	DATE	Y		The date that the contribution was made.

OMIM_Links nickname fixed columns

The columns for the OMIM_Links nickname are described in the following table.

Table 51. Columns for the OMIM_Links nickname

Column name	Data type	Null	Search tag	Description
MIM_ID	VARCHAR(6)	N		A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.
Links_Join_Key	VARCHAR (32)	N		The primary key column for the OMIM_Links nickname. This column contains a unique ID that is generated by the wrapper. This column is defined as a FOR BIT DATA column.
Link_Number	INTEGER	Y		The link number in the OMIM database entry.
Link_Type	VARCHAR (10)	Y		The type of link in the OMIM database entry. The valid types are Medline, Protein, Nucleotide, or Structure.
Link_Uids	VARCHAR (300)	Y		A list of unique IDs that are associated with the Links_Join_Key for the OMIM_Links nickname. The unique IDs are separated by commas.

OMIM_Links_uid nickname fixed columns

The columns for the OMIM_Links_uid nickname are described in the following table.

Table 52. Columns for the OMIM_Links_uid nickname

Column name	Data type	Null	Search tag	Description
Links_Join_Key	VARCHAR (32)	N		A foreign key that is used in a join condition with the parent nickname, OMIM_Links. This column is defined as a FOR BIT DATA column.

Table 52. Columns for the OMIM_Links_uid nickname (continued)

Column name	Data type	Null	Search tag	Description
Uid	VARCHAR (10)	N		The unique ID that is associated with the Links_Join_Key for the OMIM_Links_uid nickname.

Fixed columns for PubMed nicknames

When you register the PubMed nicknames, information about the fields in the PubMed database is added to the federated database catalog. The information that is added to the catalog is represented as a set of fixed columns for each nickname.

List of the PubMed nicknames

The PubMed nicknames are:

- PMArticles
- PMAccession
- PMChemical
- PMMeSHHeading
- PMComments
- PMArticleID

The following tables provide information about the columns in each nickname:

- The name of the column.
- The data type for the column.
- The description for the column.
- The search tag that is used by the wrapper to query the PubMed database. The column names that have a search tag are the only column names that you can specify in the ENTREZ.CONTAINS functions.
- If the column can return a null value. Most of the columns that return a null value are input columns.

You can override the default length for a column when you create a nickname. For example, some columns can return a large amount of data, such as the Abstract column in the PMArticles nickname. The default length for this column is VARCHAR(32000). To return the first 100 bytes of the column, you can define the column with the data type VARCHAR(100). Only the first 100 bytes will be returned.

For a list of valid search tags, see the following NCBI web site and locate the link to Search Field Descriptions and Qualifiers: www.ncbi.nlm.nih.gov/entrez/query/static/help/pmhelp.html

The Entrez wrapper supports the CLOB data type, up to 5 megabytes in length.

PMArticles nickname fixed columns

The columns in the PMArticles nickname are described in the following table. The F column indicates columns that are designated fetch keys. Using the fetch keys might expedite query processing.

Table 53. PubMed PMArticles nickname

Column name	Data type	Description	Tags	Fetch key
PMID	VARCHAR(10) NOT NULL	Primary key column used to join the PMArticles nickname with child nicknames	UID	Yes
MedlineID	VARCHAR(10)	Medline ID	UID	Yes
Owner	VARCHAR(8) NOT NULL	Owner of the publication entry; values are defined by NCBI and might be NLM, NASA, PIP, KIE, HSR, HMD, SIS, NOTNLM. If not present, then the default is NLM.	None	No
Status	VARCHAR(32) NOT NULL	Publication status as defined by NCBI. Values might include: In-Process, Completed, Out-of-scope, PubMed-not_MEDLINE.	None	No
DateCreated	DATE NOT NULL	Date that the publication entry was created.	None	No
DateCompleted	DATE	Date that the publication entry was completed.	None	No
DateRevised	DATE	Date that the publication entry was revised.	None	No
ArticleTitle	VARCHAR(250) NOT NULL	Title of the article.	TI	No
Pagination	VARCHAR(32)	Full pagination of the article.	None	No
Abstract	VARCHAR(32000)	Abstract for the article.	TIAB	No
Affiliation	VARCHAR(250)	Affiliation and address of the first author	AD	No
AuthorList	VARCHAR(3200)	List of authors; canonized	AU	No
LanguageList	VARCHAR(250) NOT NULL	Semicolon-separated list	LA	No
PublicationTypeList	VARCHAR(250) NOT NULL	Semicolon-separated list	PT	No
VernacularTitle	VARCHAR(250)	Vernacular title for the article.	None	No
DateOfElectronic Publication	VARCHAR(32)	NCBI schema specifies no structure for this column	None	No
Country	VARCHAR(128)	Country or region of publication that is cited in the journal.	None	No
MedlineTA	VARCHAR(250) NOT NULL	Medline title abbreviation.	TA	No
NlmUniqueId	VARCHAR(32)	Contains MedlineCode if NlmUniqueID is not present	None	No
GeneSymbolList	VARCHAR(250)	A semicolon-separated list; not used since 1996	None	No
NumberOfReferences	INTEGER	Number of bibliographic references for the review article.	None	No
PersonalNameSubjectList	VARCHAR(250)	Canonized as a semicolon-separated list of names	PS	No
KeywordList	VARCHAR(3200)	A semicolon-separated list	None	No
SpaceFlightMissionList	VARCHAR(250)	A semicolon-separated list	None	No

Table 53. PubMed PMArticles nickname (continued)

Column name	Data type	Description	Tags	Fetch key
InvestigatorList	VARCHAR(250)	Canonized as a semicolon-separated list of names	None	No
PublicationStatus	VARCHAR(32)	Status of the publication.	None	No
ProviderID	VARCHAR(32)	Publication provider ID.	None	No
CitationSubsetList	VARCHAR(250)	A semicolon-separated list	SB	No
AllFields	VARCHAR(1)	Pseudo-column; always returns NULL	ALL	No
TextWords	VARCHAR(1)	Pseudo-column; always returns NULL	TW	No
PubDate	DATE	Includes the journal and book publication date + medline date	DP	No
PubDateString	VARCHAR(32)	Includes the journal and book publication date + medline date	DP	No
Title	VARCHAR(250)	Book or journal title	TA	No
Journal_ISSN	CHAR(9)	ISSN for the journal.	TA	No
Journal_Volume	VARCHAR(10)	Volume of the journal.	VI	No
Journal_Issue	VARCHAR(10)	Issue of the journal.	IP	No
Journal_Coden	VARCHAR(32)	Code number (coden) of the journal.	None	No
Journal_ISOAbbreviation	VARCHAR(32)	ISO abbreviation for the journal.	None	No
Book_Publisher	VARCHAR(128)	Publisher the book.	None	No
Book_Authors	VARCHAR(250)	Canonized as other author lists	None	No
Book_CollectionTitle	VARCHAR(128)	Collection title of the book.	None	No
Book_Volume	VARCHAR(10)	Volume of the book.	None	No

PMAccession nickname fixed columns

The columns in the PMAccession nickname are described in the following table.

Table 54. PubMed PMAccession nickname

Column name	Data type	Description	Tags
PMID	VARCHAR(10) NOT NULL	Key that is used to join a PMAccession child nickname with its parent nickname.	None
DataBankName	VARCHAR(250) NOT NULL	Name of the data bank.	SI
Accession	VARCHAR(32) NOT NULL	Accession number.	SI

PMChemical nickname fixed columns

The columns in the PMChemical nickname are described in the following table.

Table 55. PubMed PMChemical nickname

Column name	Data type	Description	Tags
PMID	VARCHAR(10) NOT NULL	Key that is used to join a PMChemical child nickname with its parent nickname.	None

Table 55. PubMed PMChemical nickname (continued)

Column name	Data type	Description	Tags
NameOfSubstance	VARCHAR(128) NOT NULL	Name of the substance.	NM
RegistryNumber	VARCHAR(32) NOT NULL	Might be a CAS or other registry number	RN
CASRegistry	CHAR	Y or N	None

PMMeSHHeading nickname fixed columns

The columns in the PMMeSHHeading nickname are described in the following table.

Table 56. PubMed PMMeSHHeading nickname

Column name	Data type	Description	Tags
PMID	VARCHAR(10) NOT NULL	Key that is used to join a PMMeSHHeading child nickname with its parent nickname.	ID
DescriptorOrName	VARCHAR(128) NOT NULL	Name or descriptor of the MeSH.	MH[1]
DescriptorIsMajor	CHAR NOT NULL	Y if the descriptor is major	None
QualifierOrSubhead	VARCHAR(128)	Qualifier or subheading of the MeSH.	SH
QSIsMajor	CHAR	Y if qualifier or subhead is major	None

Note:

1. If the predicate ″DescriptorIsMajor = Y″ is included in the query, then the search term is MAJR.

PMComments nickname fixed columns

The columns in the PMComments nickname are described in the following table.

Table 57. PubMed PMComments nickname

Column name	Data type	Description	Tags
PMID	VARCHAR(10) NOT NULL	Key that is used to join a PMComments child nickname with its parent nickname.	None
RefSource	VARCHAR(128) NOT NULL	Source of the reference.	None
Type	VARCHAR(32) NOT NULL	Examples of valid types are: CommentOn, CommentIn, ErratumIn, ErratumFor, RepublishedFrom, RepublishedIn, RetractionOf, RetractionIn, UpdateIn, UpdateOf, SummaryForPatents, OriginalReportIn	None
Note	VARCHAR(3200)	Notes	None

PMArticleID nickname fixed columns

The columns in the PMArticleID nickname are described in the following table.

Table 58. PubMed PMArticleID nickname

Column name	Data type	Description	Tags
PMID	VARCHAR(10) NOT NULL	Key that is used to join a PMArticleID child nickname with its parent nickname.	None
ArticleID	VARCHAR(32) NOT NULL	ID of the article.	None
IdType	VARCHAR(8) NOT NULL	Examples of valid article identification types are: doi, pii, pmcpid, pmpid, sici, pubmed, medline, pmcid	None

Chapter 9. Excel data sources

Configuring access to Excel data sources

You can integrate the data that is in Excel data sources with information from other sources by using a federated system.

Procedure

To configure a federated server to access Excel data sources, you must provide the federated server with information about the data sources and objects that you want to access. After you configure the federated server, you can create queries to access the Excel data sources.

Excel wrapper

An Excel workbook is a file that is created using the Microsoft Excel application and has a file extension of xls. The Excel wrapper is used to perform searches on the Excel files.

You use Excel files to store information that is best displayed in a table, with corresponding rows and columns. Excel workbooks consist of one or more spreadsheet pages, or *worksheets*. Worksheets are often used to perform calculations.

IBM WebSphere Federation Server supports files that are created in Excel 2000, Excel 2002, and Excel 2003. The following figure shows how the Excel wrapper connects your worksheets to the federated system.

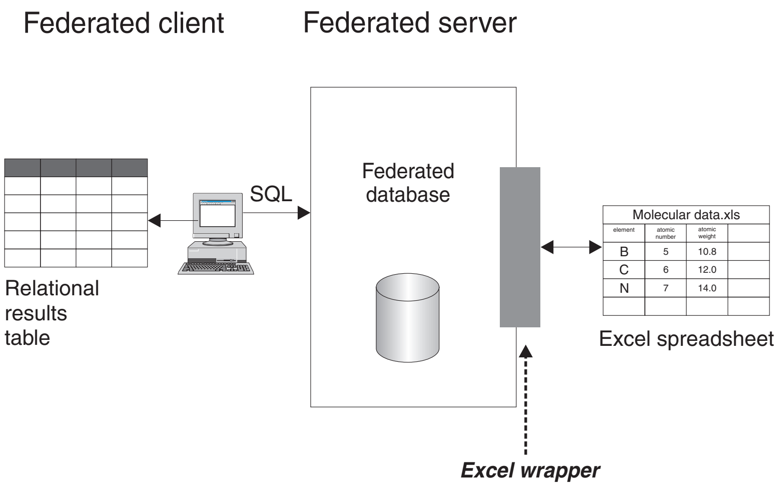


Figure 18. How the Excel wrapper works

The Excel wrapper uses the CREATE NICKNAME statement to map the columns in your Excel worksheets to columns in your federated system. The following table shows a sample of worksheet data that is stored in a file called `Compound_Master.xls`.

Table 59. Sample worksheet for Compound_Master.xls

	A	B	C	D
1	COMPOUND_NAME	WEIGHT	MOL_COUNT	WAS_TESTED
2	compound_A	1.23	367	tested
3	compound_G		210	
4	compound_F	0.000425536	174	tested
5	compound_Y	1.00256		tested
6	compound_Q		1024	
7	compound_B	33.5362		
8	compound_S	0.96723	67	tested
9	compound_O	1.2		tested

The information in an Excel worksheet is usually not available to you through standard SQL commands. When the Excel wrapper is installed and registered on your federated server, you can access this information as if it were a typical relational data source. For example, if you wanted to know all the compound data where the molecular count is greater than 100, you would run the following SQL query:

```
SELECT * FROM compound_master WHERE mol_count > 100
```

The results of the query are shown in the following table.

Table 60. Query results

COMPOUND_NAME	WEIGHT	MOL_COUNT	WAS_TESTED
compound_A	1.23	367	tested
compound_G		210	
compound_F	0.000425536	174	tested
compound_Q		1024	

Adding Excel data sources to a federated server

To configure a federated server to access Excel data sources, you must provide the federated server with information about the data sources and objects that you want to access.

Before you begin

- The data in Excel worksheets must be structured correctly so that the Excel wrapper can access the data.
- IBM WebSphere Federation Server must be installed on a server that will act as the federated server.
- A federated database must exist on the federated server

Restrictions

- The Excel wrapper is available only for versions of the Microsoft Windows operating system that are supported by the federated server.
- The Excel application must be installed on the federated server.
- The Excel workbook must be on the same computer as the federated server or on a network accessible drive.

- The data in only the first worksheet in the Excel workbook can be accessed by the Excel wrapper.
- The federated database codepage set must match the Excel file character set, otherwise you could get unexpected results from your queries.
- Pass-through sessions are not allowed.

About this task

You can configure a federated server to access data that is stored in Excel data sources by using the Control Center or by issuing SQL statements on the command line. The Control Center includes a wizard to guide you through the steps that are necessary to configure the required federated objects.

Procedure

To add the Excel data sources to a federated server:
1. "Registering the Excel wrapper."
2. "Registering the server definition for an Excel data source" on page 178.
3. "Registering nicknames for Excel data sources" on page 179.

Registering the Excel wrapper

You must register a wrapper to access Excel data sources.

About this task

Wrappers are used by federated servers to communicate with and retrieve data from the data sources. Wrappers are implemented as a set of library files.

You can register a wrapper by using the Control Center or by using the command line. The Control Center includes a wizard to guide you through the steps that are necessary to register the wrapper.

Procedure

To register the Excel wrapper:

Choose the method that you want to use to register the Excel wrapper:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.
Using the command line	Issue the CREATE WRAPPER statement. For example: `CREATE WRAPPER excel_wrapper LIBRARY 'db2lsxls.dll';` You must specify the LIBRARY parameter in the CREATE WRAPPER statement.

Excel wrapper library files

The Excel wrapper library files are added to the federated server when you install IBM WebSphere Federation Server.

When you install IBM WebSphere Federation Server, three library files are added to the default directory path. For example, if the federated server is running on Windows, the wrapper library files that are added to the directory path are db2lsxls.dll, db2lsxlsF.dll, and db2lsxlsU.dll. The default wrapper library file is db2lsxls.dll. The other wrapper library files are used with specific wrapper options.

When you register the Excel wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

The default directory path and default wrapper library file name are listed in the following table.

Table 61. Excel wrapper library location and file name

Operating system	Directory path	Wrapper library file name
Windows	%DB2PATH%\bin	db2lsxls.dll

%DB2PATH% is the environment variable that is used to specify the directory path where IBM WebSphere Federation Server is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

Registering the server definition for an Excel data source

You must register a server definition because the hierarchy of the federated objects requires that the Excel workbook files, which are identified by nicknames, are associated with a specific server definition object.

About this task

You can register a server definition by using the Control Center or by using the command line. The Control Center includes a wizard to guide you through the steps that are necessary to register the server definition.

Procedure

To register a server definition for an Excel data source:

Choose the method that you want to use to register the server definition:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.
Using the command line	Issue the CREATE SERVER statement. For example: `CREATE SERVER server_definition_name WRAPPER excel_wrapper;`

CREATE SERVER statement - Examples for the Excel wrapper

Use the CREATE SERVER statement to register server definitions for the Excel wrapper.

The following example shows you how to register a server definition called `biochem_lab` for a workbook that contains biochemical data. The CREATE SERVER statement that you issue is:

```
CREATE SERVER biochem_lab WRAPPER excel_wrapper;
```

biochem_lab
A name that you assign to the Excel server definition. Duplicate server definition names are not allowed.

WRAPPER *Excel_wrapper*
The wrapper name that you specified in the CREATE WRAPPER statement.

Registering nicknames for Excel data sources

For each Excel server definition that you register, you must register a nickname for each Excel worksheet that you want to access. Use these nicknames, instead of the names of the worksheets, when you query the Excel data sources.

About this task

When you create a nickname for an Excel worksheet, the information in the worksheet data is mapped to a relational table.

Blank cells in the worksheet are interpreted as NULL.

Up to 10 consecutive blank rows can exist in the worksheet and will be included in the data set. More than 10 consecutive blank rows are interpreted as the end of the data set.

Blank columns can exist in the worksheet. However, these columns must be registered and described as valid fields even if they are not used.

Procedure

To register a nickname for an Excel worksheet:

Choose the method that you want to use to register the nickname. Nicknames can be up to 128 characters in length.

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**. Follow the steps in the wizard.

Method	Procedure

From command line	Issue the CREATE NICKNAME statement. For example:
	`CREATE NICKNAME nickname`
	`(`
	`column_name data_type`
	`   OPTIONS (nickname_column_ options),`
	`column_name data_type`
	`   OPTIONS (nickname_column_ options),`
	`column_name data_type`
	`   OPTIONS (nickname_column_ options)`
	`)`
	`FOR SERVER server_definition_name`
	`OPTIONS (nickname_options);`

Repeat this step for each Excel worksheet that you want to create a nickname for.

CREATE NICKNAME statement - examples for the Excel wrapper

Use the CREATE NICKNAME statement to register a nickname for an Excel worksheet that you want to access. These examples show the required parameters and optional nickname options.

```
CREATE NICKNAME Compounds
    (
    Compound_ID  INTEGER,
    CompoundName VARCHAR(50),
    MolWeight  FLOAT
    )
    FOR SERVER biochem_lab
    OPTIONS (FILE_PATH 'C:\My Documents\CompoundMaster.xls',
        RANGE 'B2:D25');
```

Compounds
: A unique nickname that is used to identify the Excel worksheet.

Important: The nickname is a two-part name the consists of the schema and the name of the nickname. If you omit the schema when you register the nickname, the authorization ID of the user who registers the nickname is used for the nickname schema.

Compound_ID *INTEGER*
: The name and data type for a worksheet column that contains the compound identifiers.

CompoundNAME *VARCHAR(50)*
: The name and data type for a worksheet column that contains the compound names.

MolWeight *FLOAT*
: The name and data type for a worksheet column that contains the molecular weight of the compounds.

FOR SERVER *biochem_lab*
: The name that you assigned to the Excel server definition in the CREATE SERVER statement.

FILE_PATH *'C:\My Documents\CompoundMaster.xls'***)**
: Specifies the fully qualified directory path and file name for the Excel workbook that contains the data you want to access. The data must be in the first worksheet in the workbook.

OPTIONS (RANGE *'B2:D25'***)**

Specifies the range of cells that you want to access in the workbook that you specified in the FILE_PATH nickname option.

Any syntax or semantic error in the range option value results in an SQL1882E message. Errors might include:

- The range is not a valid range. For example, if the top-left cell specified in the range is either below or to the right of the bottom-right cell.
- The number of columns designated by the range value does not correspond to the number of columns specified in the CREATE NICKNAME statement.
- A nonvalid character or other syntax error has been found.

Excel data sources - example queries

To access Excel data, you use the nickname and the defined nickname columns in your SQL statements in the same manner as you would use a regular table name and table columns.

These examples show you how to structure the queries to access Excel data using the nickname compounds.

Selecting a specific column of information

The following query displays all compound_IDs where the molecular weight is greater than 2000:

```
SELECT compound_ID FROM compounds
   WHERE molweight > 200;
```

Using an OR condition in your SELECT statements

The following query displays all records where the compound name or molecular weight is null:

```
SELECT * FROM compounds
   WHERE compoundname IS NULL OR molweight IS NULL;
```

Using LIKE and AND conditions in your SELECT statements

The following query displays all records where the compound name contains the string `ase` and the molecular weight is greater than or equal to 300:

```
SELECT * FROM compounds
   WHERE compoundname LIKE '%ase% AND molweight >= 300;
```

Excel data source – sample scenario

This scenario shows the SQL statements that are necessary to register the federated objects that are used to access an Excel worksheet. Included in this scenario are several queries that you can run using the nickname that you create.

Excel worksheet information

This scenario starts with a worksheet that contains information about various compounds. The name of the workbook that the worksheet is in is Compound_Master.xls and the workbook was created in Excel. The fully-qualified path name to the workbook is C:\Data\Compound_Master.xls.

The first worksheet in the workbook contains four columns and nine rows of data. The columns list the names of the compounds, the weight of the compounds, the molecular count of the compound, and if the compound has been tested.

The contents of the worksheet are shown in the following table.

Table 62. Sample worksheet Compound_Master.xls

	A	B	C	D
1	COMPOUND_NAME	WEIGHT	MOL_COUNT	WAS_TESTED
2	compound_A	1.23	367	tested
3	compound_G		210	
4	compound_F	0.000425536	174	tested
5	compound_Y	1.00256		tested
6	compound_Q		1024	
7	compound_B	33.5362		
8	compound_S	0.96723	67	tested
9	compound_O	1.2		tested

Register the federated objects

To access the worksheet using the Excel wrapper you must register the objects on the federated server:

1. Register the Excel wrapper.

   For example:

   ```
   CREATE WRAPPER Excel LIBRARY 'db2lsxls.dll';
   ```

2. Register the server definition:

   For example:

   ```
   CREATE SERVER biochem_lab WRAPPER Excel;
   ```

3. Register a nickname that refers to the Excel worksheet:

   For example:

   ```
   CREATE NICKNAME Compound_Master
        (compound_name VARCHAR(40),
         weight        FLOAT,
         mol_count     INTEGER,
         was_tested    VARCHAR(20))
       FOR SERVER biochem_lab
       OPTIONS (FILE_PATH 'C:\Data\Compound_Master.xls');
   ```

The registration process is complete. The Excel worksheet is now part of the federated system, and can be used in SQL queries.

The following examples show the SQL queries and the results that arereturned from the *Compound_Master* nickname.

A query that returns all data that matches a specific WHERE clause condition

To return all of the data for the compounds that have a molecular count that is greater than 100, issue this query:

```
SELECT * FROM Compound_Master
    WHERE mol_count > 100;
```

A query that returns specific columns from the worksheet

To return the names and molecular counts for all of the compounds where the molecular count has not yet been determined, issue this query:

```
SELECT compound_name, mol_count FROM Compound_Master
    WHERE mol_count IS NULL;
```

All of the columns from rows 2, 3, 4, 6, and 8 are returned.

The *compound_name* and *mol_count* columns from rows 5, 7, and 10 are returned.

A query that counts the number of rows that match specific WHERE clause conditions

To return the number of compounds that have a weight that is greater than 1 and that have not been tested, issue this query:

```
SELECT count(*) FROM Compound_Master
    WHERE was_tested IS NULL AND weight > 1
```

The record count of 1 is returned. The compound in row 7 matches the query criteria.

A query that returns specific columns from the worksheet and includes a subselect statement

To return the names and molecular counts for all of the compound where the molecular count has been determined and the molecular count is less than the average molecular count, issue this query:

```
SELECT compound_name, mol_count FROM Compound_Master
    WHERE mol_count IS NOT NULL
    AND mol_count <
    (SELECT AVG(mol_count) FROM Compound_Master
         WHERE mol_count IS NOT NULL AND was_tested  IS NOT NULL);
```

The subquery returns 368 for the molecular count average. The main query uses the average to return the query results that are shown in the following table:

Table 63. Query results

COMPOUND_NAME	MOL_COUNT
compound_A	367
compound_G	210
compound_F	174
compound_S	67

File access control model for the Excel wrapper

To access an Excel file, the wrapper needs a user identity for security purposes. The Excel wrapper uses the user identity that is associated with the federated database service. The name of the federated database service depends on the name of the database instance. For example, if the database instance name is DB2, then the service name is DB2 - DB2. To determine the user identity that is associated with federated database service, use the Control Panel in Windows to display the services. Double-click the service name and display the Log On properties page.

Chapter 10. HMMER data sources

Configuring access to HMMER data sources

You can query and integrate the data that is in HMMER data sources with information from other sources by using a federated system.

Procedure

To configure a federated server to access HMMER data sources, you must provide the federated server with information about the data sources and objects that you want to access. After you configure the federated server, you can create queries to access the HMMER data sources.

HMMER wrapper

HMMER is a application package that you can use to search gene sequence databases that use statistical models or profile hidden Markov models (HMMs). The HMMER wrapper transforms the query statements into a format that the HMMER application package can interpret and starts the hmmpfam program or the hmmsearch program to run the query.

You can download the HMMER application package at no charge from http://hmmer.wustl.edu/. You can install the HMMER application package on a separate HMMER server or on the federated server.

An HMM is a statistical model of the primary structure consensus of a gene sequence family. An HMM is based upon probability models. You can train an HMM to recognize patterns from unaligned gene sequences if a trusted alignment is not yet known. You need less skill and manual intervention to train and use a successful HMM than to carefully construct a profile. You can use a trained HMM to access libraries of hundreds of profile HMMs and apply them on a very large scale to whole genome or Expressed Sequence Tag (EST) analyses.

PFAM (Protein Families Database of Alignments and HMMs) is a database of protein domain models. The HMMER application package is tightly tied to the construction and use of the PFAM database.

The HMMER application package contains the nine programs, but only two of these programs are supported by the IBM WebSphere Federation Server, the hmmpfam and the hmmsearch programs.

Table 64. The HMMER programs supported by the HMMER wrapper

HMMER program	Description
hmmpfam	Uses a specific gene sequence to search an HMM database and determine the family that the test gene sequence might belong to. Calculates how well each model matches a specified sequence and a database of models. The match is expressed in terms of statistical significance.
hmmsearch	Uses a specific HMM profile to search a sequence database for significantly similar sequence matches.

Users or applications issue SQL query statements with HMMER-specific predicates to the federated server. The predicates in these statements map to command-line options in the hmmpfam or hmmsearch programs.

A special daemon program runs on the server where the HMMER application package is installed. This daemon receives the query request from the federated server and sends it to the HMMER application package. The HMMER application package runs the query on a profile database, such as PFAM.

The following figure shows how HMMER works with your federated system.

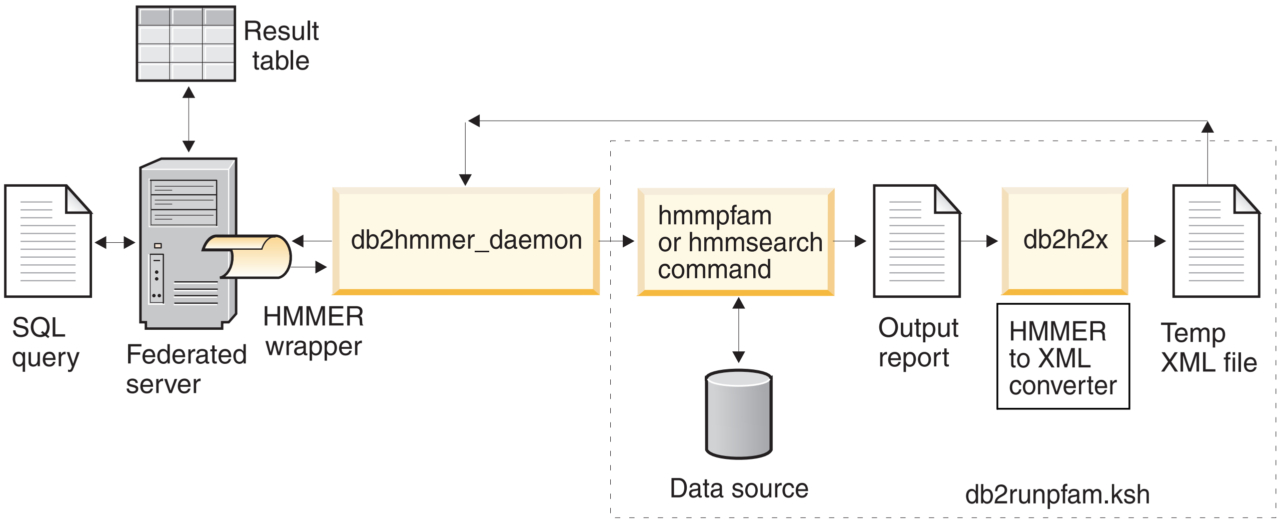


Figure 19. How the HMMER wrapper works

The daemon returns the results to the HMMER wrapper. The wrapper transforms the data into a relational table, and returns this table to the user or application.

The following example shows how information is extracted from profile databases, which are constructed by HMMER programs, and displayed as a relational table. The HMMER User's Guide http://hmmer.wustl.edu/ provides examples of creating profile databases and a HMMER tutorial.

The example shows a sample query that uses the 7LES_DROME gene sequence. You specify sequences in the WHERE clause of the query.

```
SELECT Model, ModelScore, DomainNumber, DomainScore
FROM myhmms
WHERE HmmQSeq = 'MTMFWQQNVDHQSDEQDKQAKGAAPTKRLNISFNVKIAVNVNTKMTTTH
INQQAPGTSSSSSNSQNASPSKIVVRQQSSSFDLRQQLARLGRQLASGQDGHGGISTILIINLLLL
ILLSICCDVCRSHNYTVHQSPEPVSKDQMRLLRPKLDSDVVEKVAIWHKHAAAAPPSIVEGIAISS
RPQSTMAHHPDDRDRDRDPSEEQHGVDERMVLERVTRDCVQRCIVEEDLFLDEFGIQCEKADNGEK
CYKTRCTKGCAQWYRALKELESCQEACLSLQFYPYDMPCIGACEMAQRDYWHLQRLAISHLVERTQ
PQLERAPRADGQSTPLTIRWAMHFPEHYLASRPFNIQYQFVDHHGEELDLEQEDQDASGETGSSAW
FNLADYDCDEYYMCEILEALIPYTQYRFRFELPFGENRDEVLYSPATPAYQTPPEGAPISAPVIEH
LMGLDDSHLAVHWHPGRFTNGPIEGYRLRLSSSEGNATSEQLVPAGRGSYIFSQLQAGTNYTLALS
MINKQGEGPVAKGFVQTHSARNEKPAKDLTESVLLVGRRAVMWQSLEPAGENSMIYQSQEELADIA
WSKREQQLWLLNVHGELRSLKFESGQMVSPAQQLKLDLGNISSGRWVPRRLSFDWLHHRLYFAMES
PERNQSSFQIISTDLLGESAQKVGESFDLPVEQLEVDALNGWIFWRNEESLWRQDLHGRMIHRLLR
IRQPGWFLVQPQHFIIHLMLPQEGKFLEISYDGGFKHPLPLPPPSNGAGNGPASSHWQSFALLGRS
LLLPDSGQLILVEQQGQAASPSASWPLKNLPDCWAVILLVPESQPLTSAGGKPHSLKALLGAQAAK
ISWKEPERNPYQSADAARSWSYELEVLDVASQSAFSIRNIRGPIFGLQRLQPDNLYQLRVRAINVD
GEPGEWTEPLAARTWPLGPHRLRWASRQGSVIHTNELGEGLEVQQEQLERLPGPMTMVNESVGYYV
TGDGLLHCINLVHSQWGCPISEPLQHVGSVTYDWRGGRVYWTDLARNCVVRMDPWSGSRELLPVFE
ANFLALDPRQGHLYYATSSQLSRHGSTPDEAVTYYRVNGLEGSIASFVLDTQQDQLFWLVKGSGAL
RLYRAPLTAGGDSLQMIQQIKGVFQAVPDSLQLLRPLGALLWLERSGRRARLVRLAAPLDVMELPT
PDQASPASALQLLDPQPLPPRDEGVIPMTVLPDSVRLDDGHWDDFHVRWQPSTSGGNHSVSYRLLL
EFGQRLQTLDLSTPFARLTQLPQAQLQLKISITPRTAWRSGDTTRVQLTTPPVAPSQPRRLRVFVE
RLATALQEANVSAVLRWDAPEQGQEAPMQALEYHISCWVGSELHEELRLNQSALEARVEHLQPDQT
YHFQVEARVAATGAAAGAASHALHVAPEVQAVPRVLYANAEFIGELDLDTRNRRRLVHTASPVEHL
VGIEGEQRLLWVNEHVELLTHVPGSAPAKLARMRAEVLALAVDWIQRIVYWAELDATAPQAAIIYR
LDLCNFEGKILQGERVWSTPRGRLLKDLVALPQAQSLIWLEYEQGSPRNGSLRGRNLTDGSELEWA
TVQPLIRLHAGSLEPGSETLNLVDNQGKLCVYDVARQLCTASALRAQLNLLGEDSIAGQLAQDSGY
LYAVKNWSIRAYGRRRQQLEYTVELEPEEVRLLQAHNYQAYPPKNCLLLPSSGGSLLKATDCEEQR
CLLNLPMITASEDCPLPIPGVRYQLNLTLARGPGSEEHDHGVEPLGQWLLGAGESLNLTDLLPFTR
YRVSGILSSFYQKKLALPTLVLAPLELLTASATPSPPRNFSVRVLSPRELEVSWLPPEQLRSESVY
YTLHWQQELDGENVQDRREWEAHERRLETAGTHRLTGIKPGSGYSLWVQAHATPTKSNSSERLHVR
SFAELPELQLLELGPYSLSLTWAGTPDPLGSLQLECRSSAEQLRRNVAGNHTKMVVEPLQPRTRYQ
CRLLLGYAATPGAPLYHGTAEVYETLGDAPSQPGKPQLEHIAEEVFRVTWTAARGNGAPIALYNLE
ALQARSDIRRRRRRRRRNSGGSLEQLPWAEEPVVVEDQWLDFCNTTELSCIVKSLHSSRLLLFRVR
ARSLEHGWGPYSEESERVAEPFVSPEKRGSLVLAIIAPAAIVSSCVLALVLVRKVQKRRLRAKKLL
QQSRPSIWSNLSTLQTQQQLMAVRNRAFSTTLSDADIALLPQINWSQLKLLRFLGSGAFGEVYEGQ
LKTEDSEEPQRVAIKSLRKGASEFAELLQEAQLMSNFKHENIVRLVGICFDTESISLIMEHMEAGD
LLSYLRAARATSTQEPQPTAGLSLSELLAMCIDVANGCSYLEDMHFVHRDLACRNCLVTESTGSTD
RRRTVKIGDFGLARDIYKSDYYRKEGEGLLPVRWMSPESLVDGLFTTQSDVWAFGVLCWEILTLGQ
QPYAARNNFEVLAHVKEGGRLQQPPMCTEKLYSLLLLCWRTDPWERPSFRRCYNTLHAISTDLRRT
QMASATADTVVSCSRPEFKVRFDGQPLEEHREHNERPEDENLTLREVPLKDKQLYANEGVSRL'
```

Figure 20. Sample query run on 7LES_DROME data

The HMMER wrapper transforms the results from the query into the relational table as shown in the following table.

Table 65. The HMMER results are transformed into a relational table

Model	ModelScore	DomainNumber	DomainScore
pkinase	+3.04100000000000E+002	1	+3.04100000000000E+002
fn3	+1.76300000000000E+002	1	+4.90000000000000E+001
fn3	+1.76300000000000E+002	2	+1.36000000000000E+001
fn3	+1.76300000000000E+002	3	+1.62000000000000E+001
fn3	+1.76300000000000E+002	4	+6.35000000000000E+001
fn3	+1.76300000000000E+002	5	+1.46000000000000E+001
fn3	+1.76300000000000E+002	6	+1.94000000000000E+001
rrm	-4.45000000000000E+001	1	-4.45000000000000E+001

The data is now in a relational format and can be joined with data from other data sources.

Adding HMMER data sources to a federated server

To configure a federated server to access HMMER data sources, you must provide the federated server with information about the data sources and objects that you want to access.

Before you begin

- IBM WebSphere Federation Server must be installed on a server that will act as the federated server
- A federated database must exist on the federated server

About this task

You can configure a federated server to access data that is stored in HMMER data sources by using the Control Center or by issuing SQL statements on the command line. The Control Center includes a wizard to guide you through the steps that are necessary to configure the required federated objects.

Procedure

To add HMMER data sources to a federated server:

1. Verify that the correct version of the HMMER program executable files are installed.
2. Configure the HMMER daemon.
3. Start the HMMER daemon.
4. Register the wrapper.
5. Register the server definitions.
6. Optional: Create the user mappings.
7. Register the nickname.

Verifying the version of the HMMER program executable

You must have a supported version of the hmmpfam and hmmsearch executable files installed on the server where the HMMER application program is installed.

About this task

The HMMER application program can be installed on the federated server or a separate HMMER server.

Procedure

To check the version level of the HMMER executable files:

1. Issue a command that returns the version number.
   - For the hmmpfam program, the command is `hmmpfam -h`.
   - For the hmmsearch program, the command is `hmmsearch -h`.
2. In the output file, check the version of the executable files. You must have HMMER version 2.2g, or later.

If you do not have the correct version, download the files from http://hmmer.wustl.edu/.

Configuring the HMMER daemon

A HMMER daemon is required by the HMMER wrapper to access HMMER data sources. The HMMER daemon must be configured before you register the HMMER wrapper.

Before you begin

The HMMER daemon must have:

- Execute access to the hmmpfam and hmmsearch executable files so that it can run HMMER searches.
- Write access to a directory in which it can write temporary files.
- Read access to at least one profile database on which you can run HMMER searches.

Restrictions

The HMMER daemon might not run properly if the executable filename or the database path contains spaces. For example, you should not install the HMMER executable file in C:\Program Files on Windows servers.

About this task

The HMMER wrapper requires a HMMER daemon. The HMMER daemon must be running on a server that you can access through TCP/IP from your federated system. This can be the same server that operates as the federated server, or a separate HMMER server.

The HMMER daemon runs separately from the HMMER wrapper and the federated database. The HMMER daemon listens for HMMER job requests from the HMMER wrapper.

During the installation of WebSphere Federation Server, a sample daemon configuration file is installed on the federated server. The name of the sample daemon configuration file is HMMER_DAEMON.config.

Procedure

To configure the HMMER daemon:

1. Ensure that the required files are on the server where HMMER is installed.

   During the installation of IBM WebSphere Federation Server, some of the required files are installed on the federated server. You must provide the other required files.

   On federated servers that run UNIX, the required files are:

   - The HMMER daemon executable file, $DB2PATH/bin/db2hmmer_daemon.
   - The HMMER daemon configuration file, %DB2PATH%/samples/lifesci/HMMER_DAEMON.config.
   - The conversion utility, %DB2PATH%/bin/db2h2x.
   - The shell script, %DB2PATH%/bin/db2runpfam.ksh.

- The HMMER program executable files, hmmpfam and hmmsearch. These files are not supplied by IBM.
- The HMMER database files. These files are not supplied by IBM.

On federated servers that run Windows, the required files are:

- The daemon executable files, %DB2PATH%\bin\db2hmmer_daemon.exe and %DB2PATH%\bin\db2hmmer_daemon_svc.exe.
- The HMMER daemon configuration file, %DB2PATH%\samples\lifesci\HMMER_DAEMON.config.
- The conversion utility, %DB2PATH%\bin\db2h2x.exe.
- The HMMER program executable files, hmmpfam.exe and hmmsearch.exe. These files are not supplied by IBM.
- The HMMER database files. These files are not supplied by IBM.

%DB2PATH% is the path where the federated database is installed.

By default, the daemon expects to find the configuration file in the working directory from which the daemon is started. You can copy the configuration file to another location. If you use a HMMER server, you must copy the daemon configuration file from the directory on the federated server to a directory on the HMMER server. You can copy the daemon configuration file to any directory on the HMMER server that the daemon can access.

2. On federated servers that run UNIX, ensure that the HMMER daemon executable file, conversion utility, and the shell script are executable.

   To make the files executable, run the following command:

   ```
   chmod a+x db2hmmer_daemon db2h2x db2runpfam.ksh
   ```

3. Edit the daemon configuration file to work with your data source.
   a. Optional: You can rename the configuration file.
   b. Ensure that the first line in the configuration file is an equal sign. If the equal sign is missing, the daemon will not start. An error message will indicate that the DAEMON_PORT was not specified.
   c. Ensure that the last line in the configuration file ends with a new line.

      The sample configuration file that is provided with WebSphere Federation Server ends with a new line. If the last line does not end with a new line, you will receive an error message when you attempt to run your first HMMER query using the data source that is listed on the last line.
   d. Specify the following options in the configuration file. For options that require paths, you can specify the relative paths. Relative paths are relative to the directory from which the daemon process was started.

      DAEMON_PORT
      This is the network port on which the daemon listens for HMMER job requests submitted by the wrapper.

      MAX_PENDING_REQUESTS
      This is the maximum number of HMMER job requests that can be blocking on the daemon at any one time. This number does not represent the number of HMMER jobs that run concurrently, only the number of job requests that can block at one time. It is recommended that you set this to a number greater than five. The HMMER daemon does not restrict the number of HMMER jobs that can run concurrently.

DAEMON_LOGFILE_DIR

This is the directory in which the daemon creates its log file. This file contains useful status and error information generated by the HMMER daemon.

Q_SEQ_DIR_PATH

This is the directory in which a temporary query sequence data file is created by the daemon. This temporary file is cleaned up once the HMMER job completes.

HMMER_OUT_DIR_PATH

This is the directory in which the daemon creates the temporary file to store the HMMER output data. Data is read from this file and passed back to the wrapper through the network connection. After the data is passed to the wrapper, the daemon cleans up the temporary file.

RUNPFAM_PATH

This is the fully-qualified name of the `db2runpfam.ksh` shell script provided with WebSphere Federation Server. This option is ignored if it is specified on Windows.

HMMERPFAM_PATH

This is the fully-qualified name of the HMMER executable file on the computer that is running the daemon. On UNIX, the name of the file is `hmmpfam`. On Windows, the name of the file is `hmmpfam.exe`.

HMMSEARCH_PATH

This is the fully-qualified name of the HMMER executable file on the computer that is running the daemon. On UNIX, the name of the file is `hmmsearch`. On Windows, the name of the file is `hmmsearch.exe`.

H2X_PATH

This is the fully-qualified name of the conversion program (HMMER to XML) provided with the daemon. On UNIX, the name of the program is `db2h2x`. On Windows, the name of the program is `db2h2x.exe`.

database specification entry

Specifies the location of a profile database or sequence file. Make note of the database *data_source_name* that you specify in the configuration file. For the daemon to function properly, you must specify the database *data_source_name* when you create the nickname for the data source. The name is case-sensitive. The database *data_source_name* is specified in:

- The DATASOURCE option of the CREATE NICKNAME statement (for hmmpfam)
- The MODEL predicate of the CREATE NICKNAME statement (for hmmsearch)

The configuration file must contain at least one database specification entry that specifies the fully qualified path for the profile or database. For example:

```
data_source_name=profile_or_sequence_database_name
```

On federated servers that run UNIX

For example, to specify the MYHMMS profile database you would add the following line to the daemon configuration file:

```
myhmms=/home/user_ID/myhmms
```

On federated servers that run Windows

For example, to specify MYHMMS profile database you would add the following line to the daemon configuration file:

```
myhmms=c:\hmmer\tutorial\myhmms
```

HMMER daemon configuration file - examples

The examples show the required settings and specifications for PFAM and SEARCH data sources. There are daemon configuration file examples for both UNIX and Windows servers.

The following examples show the contents of a sample HMMER daemon configuration file. The HMMER configuration file can run on the federated server or on a separate HMMER server. The HMMER configuration file must be on the same server as the HMMER executable files.

HMMER_DAEMON.config file for UNIX

This example shows the required options and profile database specification for a server that runs UNIX.

```
=
DAEMON_PORT=4098
MAX_PENDING_REQUESTS=10
DAEMON_LOGFILE_DIR=./
Q_SEQ_DIR_PATH=./
HMMER_OUT_DIR_PATH=./
RUNPFAM_PATH=./db2runpfam.ksh
HMMPFAM_PATH= ./db2runpfam.ksh /home/user_id/hmmer/bin/hmmpfam
HMMSEARCH_PATH= ./db2runpfam.ksh /home/user_id/hmmer/bin/hmmsearch
H2X_PATH=/home/user_id/sqllib/bin/db2h2x
myhmms=/home/user_id/hmmer/tutorial/myhmms
globin=/home/user_id/hmmer/tutorial/globin.hmm
pfamls=/home/user_id/hmmer/pfam/Pfam_ls
```

HMMER_DAEMON.config file for Windows

This example shows the required options and profile database specification for a server that runs Windows.

```
=
DAEMON_PORT=4098
MAX_PENDING_REQUESTS=10
DAEMON_LOGFILE_DIR=.\
Q_SEQ_DIR_PATH=.\
HMMER_OUT_DIR_PATH=.\
HMMPFAM_PATH=c:\hmmer\bin\hmmpfam.exe
HMMSEARCH_PATH=c:\hmmer\bin\hmmsearch.exe
H2X_PATH=.\db2h2x.exe
myhmms=c:\hmmer\tutorial\myhmms
globin=c:\hmmer\tutorial\globin.hmm
pfamseq=c:\hmmer\pfam\pfamseq
```

Starting the HMMER daemon

Before you can access HMMER data sources, you must start the HMMER daemon.

Before you begin

Before you start the HMMER daemon, you must have write access to all paths listed under the `DAEMON_LOGFILE_DIR`, `HMMER_OUT_DIR_PATH`, and `Q_SEQ_DIR_PATH` entries in the configuration file.

About this task

The executable file starts a new process in which the HMMER daemon runs.

Procedure

To start the HMMER daemon:

1. Open the directory where the daemon executable file is located.
2. Issue the `db2hmmer_daemon` command to run the executable files with the parameters that you need.

   There are options that you can specify when you issue the command.

db2hmmer_daemon command - options and examples

The `db2hmmer_daemon` command can be used on UNIX and Windows servers. Some of the options listed in the syntax can be used only on Windows servers.

Options - db2hmmer_daemon command

The options for the `db2hmmer_daemon` command are:

```
db2hmmer_daemon -a action -c config_file -d debug_level
    -u user_id -p password
```

-a *action*

Performs the specified activity. Valid actions are *status*, *install*, *start*, *stop*, and *remove*.

You can specify this option only on Windows servers.

-c *config_file*

Instructs the daemon to use the specified configuration file. If you do not specify the configuration file, the daemon searches for the `HMMER_DAEMON.config` file in the directory where the daemon executable files are installed.

You can specify this option on UNIX and Windows servers. On Windows servers, this option can be specified only with the *install* or *start* actions.

-d *debug_level*

Sets the daemon debug level to the specified value. The valid values are 1, 2, or 3. You can use this option with the *install* and *start* actions.

You can specify this option on UNIX and Windows servers.

-u *user_id*

Sets the daemon to run under the specified user ID. You can use this option with the *install* action.

You can specify this option only on Windows servers.

-p *password*

Specifies the password for the specified user ID. The password is valid and required only when you specify the -u option. If the -p option is not specified when you set the -u option, the program prompts you for the password. You can use this option with the *install* action.

You can specify this option only on Windows servers.

The options that are specified with the *start* action apply only until the daemon is stopped, and override the values that are specified with the *install* action. If the daemon is stopped and then started without the option, then the value of the option is whatever was specified with the install action, or the built-in default for the daemon.

For example, if the commands are issued in the following order, the HMMER1.config file is specified with the *install* action. The HMMER2.config file is specified with the *start* action and used until the daemon is stopped. When the daemon is started again, the HMMER1.config file is used.

```
[1]db2hmmer_daemon -a install -c HMMER1.config
[2]db2hmmer_daemon -a start -c HMMER2.config
[3]db2hmmer_daemon -a stop
[4]db2hmmer_daemon -a start
```

Examples - db2hmmer_daemon command

The following examples show you how to use the options with the db2hmmer_daemon command.

Start the daemon

To start the daemon, issue the following command:

```
db2hmmer_daemon -a start
```

This command assumes that the daemon configuration file is in the same directory as the executable file.

Daemon configuration file

If you changed the name of the daemon configuration file or if the configuration file is not in the same directory as the daemon executable file, you must use the `-c` option when you run the executable file. This option specifies the directory path and name for the daemon configuration file.

For example, if the daemon configuration information in a file called HMMER_D.config in the subdirectory cfg on a UNIX server, issue the following command:

```
dbhmmer_daemon -c cfg/HMMER_D.config
```

Debugging

If you want to start the daemon with debugging turned on with a debug level of 2, issue the following commands:

```
db2hmmer_daemon -a install -d 2

db2hmmer_daemon -a start
```

Status of the daemon (Windows)

To check the status of the daemon on a Windows server, issue the following command:

```
db2hmmer_daemon -a status
```

This command assumes that the daemon configuration file is in the same directory as the executable file.

Stop the daemon

To stop the daemon on Windows, issue the following command:

```
db2hmmer_daemon -a stop
```

This command assumes that the daemon configuration file is in the same directory as the executable file.

To stop the HMMER daemon on a UNIX server, determine the process ID of the db2hmmer_daemon by issuing the `ps -ef | grep db2hmmer` command. Issue the command `kill` *nnnn*, where *nnnn* is the process ID of the db2hmmer_daemon.

Remove the daemon

To remove the daemon, issue the following command:

```
db2hmmer_daemon -a remove
```

Registering the HMMER wrapper

You must register a wrapper to access HMMER data sources. Wrappers are used by federated servers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.

About this task

You can register a wrapper by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps that are necessary to register the wrapper.

If you use a proxy server and a keystore to access HMMER files, you can specify the keystore information as options when you register the wrapper or server definition. If you specify the keystore information when you register the wrapper, the settings are used when you query any HMMER file unless you specify different settings when you register the server definitions.

Procedure

To register the HMMER wrapper:

Choose the method that you want to use to register the HMMER wrapper:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.

Method	Procedure

From the command line	Issue the CREATE WRAPPER statement.
	`CREATE WRAPPER wrapper_name`
	`LIBRARY library_name;`
	If you use a proxy server to access HMMER data sources, the statement that you issue is:
	`CREATE WRAPPER wrapper_name`
	`LIBRARY library_name`
	`OPTIONS (PROXY_TYPE 'type',`
	`PROXY_SERVER_NAME 'server_name',`
	`PROXY_SERVER_PORT 'port_number');`

You must specify the LIBRARY parameter in the CREATE WRAPPER statement. The name of the wrapper library file that you specify depends on the operating system of the federated server. See the list of HMMER wrapper library files for the correct library name to specify in the CREATE WRAPPER statement.

If you use a proxy server or a keystore to access HMMER files, you must specify several wrapper options when you register the HMMER wrapper.

HMMER wrapper library files

The HMMER wrapper library files are added to the federated server when you install IBM WebSphere Federation Server.

When you install IBM WebSphere Federation Server, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2lshmmer.a, libdb2lshmmerF.a, and libdb2lshmmerU.a. The default wrapper library file is libdb2lshmmer.a. The other wrapper library files are used with specific wrapper options.

You must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

The default directory paths and default wrapper library file names are listed in the following table.

Table 66. HMMER wrapper library locations and file names

Operating system	Directory path	Wrapper library file name
AIX	/usr/opt/<install_path>/lib32/ /usr/opt/<install_path>/lib64/	libdb2lshmmer.a
Linux	/opt/IBM/db2/<install_path>/lib32 /opt/IBM/db2/<install_path>/lib64	libdb2lshmmer.so
Solaris	/opt/IBM/db2/<install_path>/lib32 /opt/IBM/db2/<install_path>/lib64	libdb2lshmmer.so
Windows	%DB2PATH%\bin	db2lshmmer.dll

`<install_path>` is the directory path where IBM WebSphere Federation Server is installed on UNIX or Linux.

%DB2PATH% is the environment variable that is used to specify the directory path where IBM WebSphere Federation Server is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

CREATE WRAPPER statement - examples for the HMMER wrapper

Use the CREATE WRAPPER statement to register the HMMER wrapper. The examples show the parameters that are required to access HMMER documents with and without a proxy server.

Example of registering a wrapper

If you are not using a proxy server to access HMMER documents, the statement that you issue to register the wrapper is:

```
CREATE WRAPPER hmmer_wrapper LIBRARY 'libdb2lshmmer.a';
```

hmmer_wrapper
: A name that you assign to the HMMER wrapper. Duplicate wrapper names are not allowed.

LIBRARY *'libdb2lshmmer.a'*
: The name of the wrapper library file for federated servers that use AIX operating systems.

Example of registering a wrapper for an HTTP proxy server

To register a wrapper and specify an HTTP proxy server, use the following statement:

```
CREATE WRAPPER hmmer_proxy LIBRARY 'libdb2lshmmer.a'
    OPTIONS (PROXY_TYPE 'HTTP',
        PROXY_SERVER_NAME 'proxy.mysite.com',
        PROXY_SERVER_PORT '81');
```

PROXY_TYPE *'HTTP'*
: Specifies the proxy type that is used to access the Internet when behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy.mysite.com'*
: Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'81'*
: Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Example of registering a wrapper for a SOCKS proxy server

To register a wrapper and specify a SOCKS proxy server without authentication information, use the following statement:

```
CREATE WRAPPER hmmer_wrapper LIBRARY 'libdb2lshmmer.so'
    OPTIONS (PROXY_TYPE 'SOCKS',
        PROXY_SERVER_NAME 'proxy_socks',
        PROXY_SERVER_PORT '1081');
```

LIBRARY *'libdb2lshmmer.so'*
: The name of the wrapper library file for federated servers that use Linux and Solaris operating systems.

PROXY_TYPE *'SOCKS'*
Specifies the proxy type that is used to access the Internet when behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy_socks'*
Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'1081'*
Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Registering the server definitions for a HMMER data source

You must register each HMMER server that you want to access in the federated database.

About this task

You can register a server definition by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps that are necessary to register the server definition.

Procedure

To register a server definition for a HMMER data source:

1. Choose the method that you want to use to register the server definition:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.

From the command line	Issue the CREATE SERVER statement. For example:
	`CREATE SERVER server_definition_name`
	`TYPE HMMER_search_type`
	`VERSION version_number`
	`WRAPPER wrapper_name`
	`OPTIONS (NODE 'node_name',`
	`DAEMON_PORT 'port_number');`
	If you use a proxy server to access HMMER data sources, the statement that you issue is:
	`CREATE SERVER server_definition_name`
	`VERSION version_number`
	`WRAPPER wrapper_name`
	`OPTIONS (PROXY_TYPE 'type',`
	`PROXY_SERVER_NAME 'server_name',`
	`PROXY_SERVER_PORT 'port_number');`

   When you register the server definition, you specify server options in the CREATE SERVER statement. There are required server options and optional server options. The NODE and DAEMON_PORT server options are required for HMMER data sources.

2. If you have more than one computer on which the HMMER daemon is installed, you must register the server definitions on each computer.

After the server definition is registered, use the ALTER SERVER statement to add or drop server options.

CREATE SERVER statement - examples for HMMER wrapper

Use the CREATE SERVER statement to register server definitions for the HMMER wrapper. This topic includes an example with the required parameters and an example with additional server options.

Required parameters

The following example shows you how to register a server definition for the HMMER wrapper by issuing the CREATE SERVER statement:

```
CREATE SERVER hmmpfam_server TYPE PFAM
    VERSION 2.3 WRAPPER hmmer_wrapper
    OPTIONS(NODE 'someserver.someschool.edu', DAEMON_PORT '4422');
```

hmmpfam_server
A name that you assign to the HMMER server. Duplicate server definition names are not allowed.

TYPE *PFAM*
The type of HMMER search that this associated with the *hmmpfam_server* server definition. The valid values for TYPE are PFAM (for hmmpfam) or SEARCH (for hmmsearch).

VERSION *2.3*
The version of the hmmpfam or hmmsearch program executable file that you are using. The supported versions are HMMER 2.2g (or later).

WRAPPER *hmmer_wrapper*
The wrapper name that you specified in the CREATE WRAPPER statement.

NODE *'someserver.someschool.edu'*
Specifies the host name or IP address of the server on which the HMMER daemon process is running. This value is case sensitive.

Although the name of the node is specified as an option in the CREATE SERVER statement, it is required for HMMER data sources.

DAEMON_PORT *'4422'*
The port number on which the daemon listens for HMMER job requests. The port number must be the same number that you specified in the DAEMON_PORT option for the HMMER daemon configuration file. The default is port number is 4098.

A server definition for the hmmpfam program with optional parameters

This example shows a server definition for a hmmpfam program:

```
CREATE SERVER hmmpfam_server TYPE PFAM
    VERSION 2.2 WRAPPER hmmer_wrapper
    OPTIONS(NODE 'someserver.someschool.edu', DAEMON_PORT '4422',
        PROCESSORS '2', HMMPFAM_OPTIONS '--null2 --pvm');
```

PROCESSORS *'2'*
Specifies the number of processors that the HMMER program uses. This option is equivalent to the *--cpu* option for the hmmpfam and hmmsearch programs.

HMMPFAM_OPTIONS *′--null2 --pvm′*

Use this server option to specify options that are used by the hmmpfam program. These options cannot be specified in a predicate. In this example, the *--null2* and *--pvm* options will be used whenever a query is run that uses the *hmmpfam_server* server definition. The HMMPFAM_OPTIONS server option is only valid with server definition that specify TYPE *PFAM*.

A server definition for the hmmsearch program with optional parameters

This example shows a server definition for a hmmsearch program:

```
CREATE SERVER hmmsearch_server TYPE SEARCH
    VERSION 2.2 WRAPPER hmmer_wrapper
    OPTIONS(NODE 'someserver.someschool.edu', DAEMON_PORT '4422',
        PROCESSORS '2', HMMSEARCH_OPTIONS '--null2 --pvm');
```

PROCESSORS *′2′*

Specifies the number of processors that the HMMER program uses. This option is equivalent to the *--cpu* option for the hmmpfam and hmmsearch programs.

HMMSEARCH_OPTIONS *′--null2 --pvm′*

Use this server option to specify options that are used by the hmmsearch program. These options cannot be specified in a predicate. In this example, the *--null2* and *--pvm* options will be used whenever a query is run that uses the *hmmsearch_server* server definition. The HMMSEARCH_OPTIONS server option is only valid with server definition that specify TYPE *SEARCH*.

Server definitions when a proxy server is used

You must use the proxy server options in the CREATE SERVER statement if all of the following conditions are true:

- You want to retrieve data using a URI
- The URI used will retrieve data from behind a firewall, through a proxy
- The firewall or proxy used is HTTP or SOCKS

The options that you specify depend on the type of proxy server that you want to access.

Check with your network administrator for information about the type of proxy that you use, and the settings that you should specify in the proxy options.

Registering a server definition for an HTTP proxy server

To register a server definition and specify an HTTP proxy server, use the following statement:

```
CREATE SERVER hmmer_server_http
    WRAPPER hmmer_wrapper
    OPTIONS (PROXY_TYPE 'HTTP', PROXY_SERVER_NAME 'proxy_http',
        PROXY_SERVER_PORT '8080');
```

hmmer_server_http

A name that you assign to the HMMER server definition. Duplicate server definition names are not allowed.

WRAPPER *hmmer_wrapper*
: The wrapper name that you specified in the CREATE WRAPPER statement.

PROXY_TYPE *'HTTP'*
: Specifies the proxy type that is used to access the Internet when behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy_http'*
: Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'8080'*
: Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Registering a server definition for a SOCKS proxy server

To register a server definition and specify a SOCKS proxy server when authentication information is not required, use the following statement:

```
CREATE SERVER hmmer_server_socks
    WRAPPER hmmer_wrapper
    OPTIONS (PROXY_TYPE 'SOCKS', PROXY_SERVER_NAME 'proxy_socks',
        PROXY_SERVER_PORT '1081');
```

hmmer_server_socks
: A name that you assign to the HMMER server definition. Duplicate server definition names are not allowed.

WRAPPER *hmmer_wrapper*
: The wrapper name that you specified in the CREATE WRAPPER statement.

PROXY_TYPE *'SOCKS'*
: Specifies the proxy type that is used to access the Internet when behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy_socks'*
: Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'1080'*
: Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Registering a server definition for a SOCKS proxy server with authentication information

To register a server definition and specify a SOCKS proxy server with authentication information, use the following statement:

```
CREATE SERVER xml_server_socks
    WRAPPER hmmer_wrapper
    OPTIONS (PROXY_TYPE 'SOCKS', PROXY_SERVER_NAME 'proxy_socks',
        PROXY_SERVER_PORT '1081', PROXY_AUTHID 'Laura',
        PROXY_PASSWORD 'not4me');
```

PROXY_AUTHID *'Laura'*
: Specifies the user name on the proxy server. This server option is required when the value for the PROXY_TYPE server option is 'SOCKS'.

PROXY_PASSWORD *'not4me'*
: Specifies the password on the proxy server that is associated with the user

name *'Laura'*. This server option is required when the value for the PROXY_TYPE server option is 'SOCKS'.

Creating the user mappings for a HMMER data source (optional)

When you attempt to access a HMMER server, the federated server establishes a connection to the HMMER server. If your federated system uses a proxy server to access HMMER data sources, you must create user mappings.

A user mapping is an association between each federated server user ID and password and the corresponding data source user ID and password.

About this task

There are two methods for specifying user mappings with federated systems. You can use an external repository, such as LDAP, to store the user mappings or you can create the user mappings in the federated database catalog.

If you have an external repository, such as LDAP, to store the user mappings, you do not need to create user mappings. You must specify the DB2_UM_PLUGIN option on the HMMER wrapper. You can specify this option when you register or alter the wrapper.

Procedure

To map a local user ID to the HMMER server user ID and password:

Choose the method that you want to use to register the user mappings:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.

Using the command line	Issue the CREATE USER MAPPING statement. For example:
	`CREATE USER MAPPING FOR local_userID`
	`SERVER server_definition_name`
	`OPTIONS (REMOTE_AUTHID 'remote_userID',`
	`REMOTE_PASSWORD 'remote_password')`
	`PROXY_AUTHID 'proxy_server_userID',`
	`PROXY_PASSWORD 'proxy_server_password';`

If you specify authentication information for the proxy server when you register a server definition and a user mapping, the values that you specify in the CREATE USER MAPPING statement take precedence over the values that you specify in the CREATE SERVER statement.

For example, you have ten people in your organization and you specify authentication information when you register the server definition. You create user mappings for three of the ten people. When the three people access the federated system, the authentication information that you specified when you created the user mappings is used. For the remaining seven people, the authentication information that you specified when you registered the server definition is used.

CREATE USER MAPPING statement - examples for the HMMER wrapper

Use the CREATE USER MAPPING statement to map a federated server user ID to a HMMER server user ID and password.

You can create a user mapping by specifying a proxy server.

Proxy server example

The following example shows how to map a federated server user ID to a HMMER proxy server user ID and password:

To register a server definition and specify a SOCKS proxy server with authentication information, use the following statement:

```
CREATE USER MAPPING FOR katherine SERVER hmmer_proxy
      OPTIONS (REMOTE_AUTHID 'kathy',
      REMOTE_PASSWORD 'all4one'
      PROXY_AUTHID 'kate'
      PROXY_PASSWORD 'them2us');
```

katherine
: Specifies the local user ID that you are mapping to a user ID that is defined at the HMMER proxy server.

SERVER *hmmer_server*
: Specifies the server definition name that you registered in the CREATE SERVER statement for the HMMER server.

REMOTE_AUTHID *'kathy'*
: Specifies the user ID at the HMMER server to which you are mapping *katherine*. This remote ID must be in a format that is expected by the HMMER server.

REMOTE_PASSWORD *'all4one'*
: Specifies the password that is associated with *'kathy'*.

PROXY_AUTHID *'kate'*
: Specifies the user ID on the proxy server. This user mapping option is required when the proxy server requires authentication.

PROXY_PASSWORD *'them2us'*
: Specifies the password on the proxy server that is associated with the user name *'kate'*. This user mapping option is required when the proxy server requires authentication.

Registering nicknames for HMMER data sources

For each HMMER server definition that you register, you must register a nickname for each HMMER database that you want to access. Use these nicknames when you query the HMMER databases.

About this task

When you register a HMMER nickname, a set of fixed input and output columns for the profile database are automatically created with the nickname. The fixed columns are created in the federated database system catalog. You reference the fixed columns in SQL queries.

The names that you give the nicknames can be up to 128 characters in length.

You can register a nickname by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps that are necessary to register the nickname.

Procedure

To register a nickname for a HMMER database:

Choose the method that you want to use to register the nickname:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.

Using the command line	Issue the CREATE NICKNAME statement. For example:
	`CREATE NICKNAME nickname`
	`FOR SERVER server_definition_name`
	`OPTIONS (DATASOURCE data_source_name);`

The *data_source_name* must match an existing *data_source_name* in the HMMER_DAEMON.config file on the HMMER server.

Repeat this step for each HMMER database that you want to create a nickname for.

CREATE NICKNAME statement - examples for HMMER wrapper

Use the CREATE NICKNAME statement to register a nickname for a HMMER profile database that you want to access. This topic includes an example with the required parameters and an example with additional nickname options.

Required parameters example

The following example shows you how to register a nickname for searches that use the hmmpfam program. This example uses the *hmmpfam_server* server definition to access the *'myhmms'* data source.

```
CREATE NICKNAME hmmpfam_nickname
   FOR SERVER hmmpfam_server
   OPTIONS(DATASOURCE 'myhmms')
```

hmmpfam_nickname
A name that you assign to the nickname. This name must be unique.

SERVER *hmmpfam_server*
The name of the server definition that you want the nickname to be associated with.

DATASOURCE *'myhmms'*
The name of the database that you will run HMMER searches on. This database must be listed in the HMMER daemon configuration file.

Although the DATASOURCE nickname option is specified as an option in the CREATE NICKNAME statement, it is required for HMMER data sources.

Optional nickname options example

You can specify how long the HMMER wrapper waits for the results from the HMMER daemon. The following example shows you how to register a nickname for searches that use the hmmpfam program and how to specify the TIMEOUT nickname option.

```
CREATE NICKNAME hmmpfam_nickname
   FOR SERVER hmmpfam_server
   OPTIONS(DATASOURCE 'myhmms', TIMEOUT '30')
```

TIMEOUT *'30'*
The maximum time, in minutes, that the HMMER wrapper waits for results from the daemon. The default time is 5 minutes.

Fixed columns for HMMER nicknames

When you issue the CREATE NICKNAME statement for a HMMER data source, a set of fixed columns are automatically created with the nickname. You can reference the fixed columns in SQL queries.

There are fixed input columns and fixed output columns that are automatically created when you register a HMMER nickname. The fixed columns are added to the federated database system catalog.

If you want to change the default data type that is assigned to a fixed column, you can specify the column name and data type in the CREATE NICKNAME statement.

For example, to limit the output for the AlignmentConsensus column to no more than the first 100 characters, you issue the following statement:

```
CREATE NICKNAME nucleo1
    (AlignmentConsensus VARCHAR(100))
    FOR SERVER searchtest
     OPTIONS (DATASOURCE 'nucleo1', TIMEOUT '1');
```

Fixed input columns for HMMER nicknames

The fixed input columns are used as parameter-passing predicates in SQL queries.

The input columns pass standard HMMER switches to either the hmmpfam or hmmsearch programs. The HMMER program then runs on the specified data source using these switches. Fixed input columns can also be referenced in the query SELECT list and are returned as part of the results table.

Each nickname that you create is associated with a specific server definition. The server definitions are designated as either a PFAM type of server or a SEARCH type of server. There are different fixed input columns for each type of server. You can specify these columns in the WHERE clause of your SQL statements.

Fixed input columns for PFAM

The following table lists the fixed input columns that you can use with the hmmpfam program.

Table 67. Fixed input columns for servers of type PFAM

Name	Data type	Description	Operator	Switches	Returned Value
HmmQSeq	VARCHAR (32000)	Input gene sequence that is used to search	=		Same as the input value that you specify. This column is required.
ModelEValue	DOUBLE	Estimated e-value	<	-E *n*	See the query output.
ModelScore	DOUBLE	Raw score	>	-T *n*	See the query output.
DBSize	INTEGER	Calculate e-values as if the database had ′n′ gene sequences	=	-Z *n*	Same as the input value that you specify. Uses hmmpfam default if not specified.
CutMode	CHAR(2)	Cutoff mode; can be ga, tc or nc (case sensitive)	=	--cut_ga --cut_tc --cut_nc	Same as the input value that you specify. NULL if not specified.
DomainScore	DOUBLE	Domain score	>	--domT *n*	See the query output.
DomainEValue	DOUBLE	Domain e-value	<	--domE *n*	See the query output.
ForwardAlgorithm	CHAR	Use Forward algorithm rather than Viterbi; value can be ′Y′ or ′N′	=	--forward	Same as the input value that you specify. ′N′ is the default.

Fixed input columns for SEARCH

The following table lists the fixed input columns that you can use with the hmmsearch program.

Table 68. Fixed input columns for servers of type SEARCH

Name	Data type	Description	Operator	Options	Returned Value
Model	VARCHAR (32000)	Name of the HMM profile file used in the search. The name must be one of the data source names listed in the database specification entry in the `HMMER_DAEMON.config` file.	=		Same as the input value that you specify. This column is required.
SequenceEValue	DOUBLE	Estimated e-value	<	-E *n*	See the query output.
SequenceScore	DOUBLE	Raw score	>	-T *n*	See the query output.
DBSize	INTEGER	Calculate e-values as if the database had 'n' gene sequences	=	-Z *n*	Same as the input value that you specify. Uses hmmpfam default if not specified.
CutMode	CHAR(2)	Cutoff mode; can be ga, tc or nc (case sensitive)	=	--cut_ga --cut_tc --cut_nc	Same as the input value that you specify. NULL if not specified.
DomainScore	DOUBLE	Domain score	>	--domT *n*	See the query output.
DomainEValue	DOUBLE	Domain e-value	<	--domE *n*	See the query output.
ForwardAlgorithm	CHAR	Use Forward algorithm rather than Viterbi; value can be 'Y' or 'N'	=	--forward	Same as the input value that you specify. 'N' is the default.

Fixed output columns for HMMER nicknames

The fixed output columns are specified in the SELECT list in your queries. You can also specify fixed output columns as predicates in the WHERE clause.

Fixed output columns for PFAM

Each nickname that you create is associated with a specific server definition. The server definitions are designated as either a PFAM type of server or a SEARCH type of server.

The following tables list the fixed output columns for each type of server.

The following table lists the fixed columns that are returned as output for PFAM.

Table 69. Fixed output columns for servers of type PFAM

Name	Data type	Description
Model	VARCHAR(32)	Name of model.
ModelDescription	VARCHAR(64)	Text description of model.
ModelScore	DOUBLE	Raw score (″bit score″).
ModelEValue	DOUBLE	Estimated e-value.
ModelHits	INTEGER	Number of domains hit within the model.
DomainNumber	INTEGER	Specific domain (within one model).
SequenceFrom	INTEGER	Starting point of gene sequence.
SequenceFromGlobal	CHAR	′Y′ if the alignment starts at the beginning of the gene sequence.
HmmFrom	INTEGER	Starting point of consensus model.
HmmFromGlobal	CHAR	′Y′ if the alignment starts at the beginning of the consensus model.
HmmTo	INTEGER	Ending point in consensus model.
HmmToGlobal	CHAR	′Y′ if the alignment ends at the end of the consensus model.
DomainScore	DOUBLE	Raw score (″bit score″) for the isolated domain.
DomainEValue	DOUBLE	Expected value for the isolated domain.
AlignmentConsensus	VARCHAR(32000)	The HMM consensus. The amino acid shown for the consensus is the highest probability amino acid at that position according to the HMM, not necessarily the highest scoring amino acid.
AlignmentExactMatch	VARCHAR(32000)	Matches the highest probability residue in the HMM.
AlignmentSubSequence	VARCHAR(32000)	Shows the gene sequence itself.

Fixed output columns for SEARCH

The following table lists the fixed columns that are returned as output for SEARCH.

Table 70. Fixed output columns for servers of type SEARCH

Name	Data type	Description
Sequence	VARCHAR(32)	The sequence identifier.
SequenceDescription	VARCHAR(64)	Text description of the sequence.
SequenceScore	DOUBLE	Raw score (″bit score″).
SequenceEValue	DOUBLE	Estimated e-value.
SequenceHits	INTEGER	Number of domains hit within the sequence.
DomainNumber	INTEGER	Specific domain (within one sequence).

Table 70. Fixed output columns for servers of type SEARCH (continued)

Name	Data type	Description
SequenceFrom	INTEGER	Starting point of gene sequence.
SequenceFromGlobal	CHAR	'Y' if the alignment starts at the beginning of the gene sequence.
HmmFrom	INTEGER	Starting point of consensus model.
HmmFromGlobal	CHAR	'Y' if the alignment starts at the beginning of the consensus model.
HmmTo	INTEGER	Ending point in consensus model.
HmmToGlobal	CHAR	'Y' if the alignment ends at the end of the consensus model.
DomainScore	DOUBLE	Raw score ("bit score") for the isolated domain.
DomainEValue	DOUBLE	Expected value for the isolated domain.
AlignmentConsensus	VARCHAR(32000)	The HMM consensus. The amino acid shown for the consensus is the highest probability amino acid at that position according to the HMM, not necessarily the highest scoring amino acid.
AlignmentExactMatch	VARCHAR(32000)	Matches the highest probability residue in the HMM.
AlignmentSubSequence	VARCHAR(32000)	Shows the gene sequence itself.

HMMER data source - complete example

All the SQL statements that you need to issue to add HMMER data sources to a federated server are shown in this example. This examples also shows you how to use a nickname in a query.

The SQL queries for HMMER data sources must contain special input predicates that are used to pass standard HMMER options to the program executable file.

To be valid, every query passed to the HMMER wrapper must contain at least the `HmmQSeq` input predicate (for TYPE PFAM) or `model` predicate (for TYPE SEARCH). All other predicates are optional.

To construct a query against a HMMER nickname, specify the input columns in the WHERE clause and the output columns in the SELECT list.

Example for the hmmpfam program

This example creates a wrapper, server definition, and nickname on an AIX federated server for the hmmpfam program. This example also runs a query that uses a string literal for the search sequence. The SQL statements to register the federated objects are:

```
CREATE WRAPPER hmmer_wrapper LIBRARY 'libdb2lshmmer.a';

CREATE SERVER hmmpfam_server TYPE pfam
   VERSION 2.2 WRAPPER hmmer_wrapper
   OPTIONS(NODE 'HMMERserv.MyCompany.com');

CREATE NICKNAME hmmpfam_nickname
   FOR SERVER hmmpfam_server
   OPTIONS(DATASOURCE 'myhmms', TIMEOUT '1');
```

Run the 7LES_DROME gene sequence on the *hmmpfam_nickname*. The SQL query is:

```
SELECT Model, substr(ModelDescription,1,50) as ModelDescription,
   ModelScore, ModelEValue, ModelHits, DomainNumber,
   SequenceFrom, SequenceTo, SequenceFromGlobal, SequenceToGlobal,
   HmmFrom, HmmTo, HmmFromGlobal, HmmToGlobal, DomainScore, DomainEValue,
   length(HmmQSeq) as "length(HmmQSeq)",
   length(AlignmentConsensus) as "length(AConsensus)",
   length(AlignmentMatch) as "length(AMatch)",
   length(AlignmentSubSeq) as "length(ASubSeq)",
   substr(HmmQSeq,1,64) as HmmQSeq,
   substr(AlignmentConsensus,1,64) as AlignmentConsensus,
   substr(AlignmentMatch, 1,64) as AlignmentMatch,
   substr(AlignmentSubSeq, 1,64) as AlignmentSubSeq
   FROM hmmpfam_nickname
   WHERE HmmQSeq =
      'MTMFWQQNVDHQSDEQDKQAKGAAPTKRLNISFNVKIAVNVNTKMTTTHINQQAPGTSS';
```

Example for the hmmsearch program

This example creates a wrapper, server definition, and nickname on a Windows federated server for the hmmsearch program. This example also runs a query that uses a string literal for the search sequence.

The SQL statements to register the federated objects are:

```
CREATE WRAPPER hmmer_wrapper LIBRARY 'db2lshmmer.dll';

CREATE SERVER hmmsearch_serv TYPE search VERSION 2.2
   WRAPPER hmmer_wrapper
   OPTIONS(NODE 'localhost');

CREATE NICKNAME artemia FOR SERVER hmmsearch_server
   OPTIONS(DATASOURCE 'artemia', TIMEOUT '1');
```

The SQL query is:

```
SELECT Model, Sequence, substr(SequenceDescription,1,50) as SequenceDescription,
   SequenceScore, SequenceEValue, SequenceHits, DomainNumber,
   SequenceFrom, SequenceTo, SequenceFromGlobal, SequenceToGlobal,
   HmmFrom, HmmTo, HmmFromGlobal, HmmToGlobal, DomainScore, DomainEValue,
   length(AlignmentConsensus) as "length(AConsensus)",
   length(AlignmentMatch) as "length(AMatch)",
   length(AlignmentSubSeq) as "length(ASubSeq)",
   substr(AlignmentConsensus,1,200) as AlignmentConsensus,
   substr(AlignmentMatch, 1,200) as AlignmentMatch,
   substr(AlignmentSubSeq, 1,200) as AlignmentSubSeq
   FROM artemia
   WHERE Model = 'globin' and DomainScore > 50;
```

Construct new HMMER queries with samples

The following samples show you how to construct queries for HMMER data sources.

In these queries, the nickname is a name that describes the type of HMMER search and the data source, for example hmmpfam_nickname. Some examples also show how to use the HMMER wrapper with other data sources.

Query 1.

When this SQL statement runs, the wrapper uses the indicated sequence and the HMM database defined by the nickname to run the hmmpfam program. The wrapper returns the columns that are listed in the SELECT statement.

```
SELECT Model, ModelScore, ModelEValue, DomainNumber, DomainScore, DomainEvalue
    FROM hmmpfam_nickname
    WHERE HmmQSeq = 'MTMFWQQNVDHQSDEQDKQAKGAAPTKRLNISFNVKIAVNVNTKMTTTHINQ...';
```

Query 2.

When this SQL statement runs, the wrapper performs an hmmpfam search of hmmpfam_nickname that uses the indicated gene sequence. In addition, the wrapper passes the -T 0 option to the hmmpfam command. This option is from the list of fixed input columns for HMMER nicknames. The wrapper returns the three columns that are listed after SELECT.

```
SELECT Model, ModelScore, ModelEValue
   FROM hmmpfam_nickname
   WHERE HmmQSeq = 'MTMFWQQNVDHQSDEQDKQAKGAAPTKRLNISFNVKIAVNVNTKMTTTHINQ...'
   AND ModelScore > 0;
```

Query 3.

When this SQL statement runs, the wrapper performs an hmmpfam search of hmmpfam_nickname that uses the indicated gene sequence. In addition, the wrapper passes the -E 1 option to the hmmpfam command. This option is from the list of fixed input columns for HMMER nicknames. The wrapper returns the four columns that are listed after SELECT and sorts the result from highest to lowest by the DomainScore.

```
SELECT Model, DomainNumber, DomainScore, DomainEValue
   FROM hmmpfam_nickname
   WHERE HmmQSeq = 'MTMFWQQNVDHQSDEQDKQAKGAAPTKRLNISFNVKIAVNVNTKMTTTHINQ...'
   AND ModelEValue < 1
   ORDER BY DomainScore DESC;
```

Query 4.

When this SQL statement runs, the wrapper runs hmmsearch against the sequence file artemia, using the HMM specified by globin. The rows with a DomainScore greater than 50 are returned, because the wrapper passes the --domT 50 option to the **hmmsearch** command. The wrapper returns the columns specified after SELECT. Column values that are longer than 200 characters are truncated. Only the first 200 characters in these columns are returned.

```
CREATE WRAPPER hmmer_wrapper
   LIBRARY 'db2lshmmer.dll';

CREATE SERVER hmmsearch_server
   TYPE search VERSION 2.2
   WRAPPER hmmer_wrapper
   OPTIONS(NODE 'HMMERserv.MyCompany.com');

CREATE NICKNAME artemia_nickname
   FOR SERVER hmmsearch_server
   OPTIONS(DATASOURCE 'artemia', TIMEOUT '1');

SELECT Model, Sequence, substr(SequenceDescription,1,50)
   as SequenceDescription, SequenceScore, SequenceEValue,
   SequenceHits, DomainNumber, SequenceFrom,
   SequenceTo, SequenceFromGlobal, SequenceToGlobal,
   HmmFrom, HmmTo, HmmFromGlobal, HmmToGlobal, DomainScore,
   DomainEValue,
   length(AlignmentConsensus)  as "length(AConsensus)",
   length(AlignmentMatch)      as "length(AMatch)",
   length(AlignmentSubSeq)     as "length(ASubSeq)",
   substr(AlignmentConsensus,1,200) as AlignmentConsensus,
   substr(AlignmentMatch, 1,200)    as AlignmentMatch,
   substr(AlignmentSubSeq, 1,200)   as AlignmentSubSeq
   FROM artemia_nickname
   WHERE Model = 'globin'
   AND DomainScore > 50;
```

Chapter 11. Informix data sources

Configuring access to Informix data sources

To configure a federated server to access Informix data sources, you must provide the federated server with information about the data sources and objects that you want to access.

Before you begin

- The Informix Client SDK software must be installed and configured on the server that will act as the federated server.
- Check the setup of the federated server.
- Check the federated parameter to ensure that federation is enabled.
- On AIX federated servers, the AIX Base Application Development Math Library must be installed. You can determine if the Library is installed by issuing the AIX command lslpp -l bos.adt.libm.

About this task

You can configure a federated server to access data that is stored in Informix data sources by using the DB2 Control Center or by issuing SQL statements on the DB2 command line. The DB2 Control Center includes a wizard to guide you through the steps that are necessary to configure the required federated objects.

Procedure

To add Informix data sources to a federated server:

1. Set up and test the Informix client configuration file.
2. Set the Informix environment variables.
3. Register the wrapper.
4. Register the server definition.
5. Create the user mappings.
6. Test the connection to the server.
7. Register nicknames for Informix tables, views, and synonyms.

Setting up and testing the Informix client configuration file

The Informix client configuration file is used to connect to Informix databases, by using the client libraries that are installed on the federated server.

Before you begin

The Informix Client SDK software must be installed on the federated server.

About this task

The client configuration file specifies the location of each Informix database server and type of connection (protocol) for the database server.

The default location of the client configuration file depends on the operating system that is used by the federated server.

- On federated servers that run UNIX, the default location and name of the file is $INFORMIXDIR/etc/sqlhosts. The sqlhosts file is installed with the Informix client SDK.
- On federated servers that run Windows, the default location of the sqlhosts registry is the local computer.

The format of sqlhosts is described in the *Administrator's Guide for Informix Dynamic Server*.

Procedure

To set up and test the Informix client configuration file:

1. Configure the Informix Client SDK.
   - On federated servers that run UNIX, you can configure the Informix Client SDK by editing the sqlhosts file. You can also copy the sqlhosts file from another system that has Informix Connect or Informix Client SDK installed.
   - On federated servers that run Windows, you can configure the Informix Client SDK with the Informix Setnet32 utility. The Setnet32 utility sets up the sqlhosts registry.
2. Verify the location of the sqlhosts file or registry.
   - On federated servers that run UNIX, the sqlhosts file is located in the $INFORMIXDIR/etc/ directory.
   - On federated servers that run Windows, the sqlhosts information is kept in the following key in the Windows registry:

   ```
   HKEY_LOCAL_MACHINE\SOFTWARE\INFORMIX\SQLHOSTS
   ```
3. If you want to place the sqlhosts file or registry in a path other than the default search path, set the INFORMIXSQLHOSTS environment variable to specify the file location. Use one of the following options to set the INFORMIXSQLHOSTS environment variable.
   - On federated servers that run UNIX, set the INFORMIXSQLHOSTS environment variable to the fully-qualified name of the sqlhosts file.
   - On federated servers that run Windows, use the Setnet32 utility to set the INFORMIXSQLHOSTS environment variable to the name of the Windows computer that stores the registry.
4. Test the connection to ensure that the client software is able to connect to the Informix server. If the Informix dbaccess utility is on the federated server, use this tool to test the connection. Otherwise, run the Informix demo program to test the client setup.

Setting the Informix environment variables

The Informix environment variables must be set in the db2dj.ini file on the federated server.

Restrictions

Restrictions for the db2dj.ini file

About this task

There are required and optional environment variables for Informix data sources.

If you installed the Informix client software before you installed the Informix wrapper, the required Informix environment variables are set in the db2dj.ini file.

You must set the environment variables using the steps in this task if you did not install the Informix client software before you installed the Informix wrapper or if you want to set any of the optional environment variables.

Procedure

To set the Informix environment variables:

1. Use one of the following methods to set the Informix environment variables that you want to use:

Method	Step
To set the environment variables using the DB2 Control Center	Use the Federated Objects wizard in the DB2 Control Center. To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.
To set the environment variables automatically	Run the DB2 Setup program again and specify the **Custom** installation option. Follow the instructions in the wizard. **Important:** Running the installation program again will set only the required environment variables. The optional environment variables must be set manually.
To set the environment variables manually	Edit the db2dj.ini file. • On federated servers that run UNIX, the db2dj.ini file is located in the sqllib/cfg directory. • On federated servers that run Windows, the db2dj.ini file is located in the %DB2PATH%\cfg directory.

   The db2dj.ini file contains configuration information about the Informix client software that is installed on your federated server. If the file does not exist, you can create a file with the name db2dj.ini by using any text editor. In the db2dj.ini file, you must specify the fully qualified path for the environment variables; otherwise you will encounter errors. The following environment variables show what an entry in the db2dj.ini file on UNIX might look like:

   ```
   INFORMIXDIR=/informix/csdk
   INFORMIXSERVER=inf10
   ```

2. Set the Informix code page conversion environment variables (as necessary).
3. To ensure that the environment variables are set on the federated server, recycle the federated database instance.

   Issue the following commands to recycle the federated database instance:

   ```
   db2stop
   db2start
   ```

Informix environment variables

There are required and optional environment variables for Informix data sources. These variables are set in the db2dj.ini file.

The valid environment variables for Informix are:

- INFORMIXDIR
- INFORMIXSERVER
- INFORMIXSQLHOSTS (optional)
- CLIENT_LOCALE (optional)

- DB_LOCALE (optional)
- DBNLS (optional)

The CLIENT_LOCALE, DB_LOCALE, and DBNLS environment variables are code page environment variables.

Variable descriptions

INFORMIXDIR

Specifies the directory path where the Informix Client SDK software is installed.

For example:

- On federated servers that run UNIX, set the path to:

```
INFORMIXDIR=/informix/csdk
```

- On federated servers the run Windows, set the path to:

```
INFORMIXDIR=C:\informix\csdk
```

INFORMIXSERVER

Identifies the name of the default Informix server. This setting must be a valid entry in the sqlhosts file (UNIX) or the SQLHOSTS registry key (Windows). To get a value for INFORMIXSERVER, read the sqlhosts file. Select one of the *dbservername* values. The *dbservername* is the first value in each entry in the sqlhosts file.

For example:

```
INFORMIXSERVER=inf10
```

Requirement: Although the Informix wrapper does not use the value of this environment variable, the Informix client requires that this environment variable be set. The wrapper uses the value of the NODE server option, which specifies the Informix database server that you want to access.

INFORMIXSQLHOSTS

If you are using the default path for the Informix sqlhosts file, you do not need to set this environment variable. However, if you are using some other path for the Informix sqlhosts file, then you need to set this environment variable. Set the INFORMIXSQLHOSTS variable to the full path name where the Informix sqlhosts file resides.

- On federated servers that run UNIX, the default path is $INFORMIXDIR/etc/sqlhosts.
- On federated servers that run Windows, if the SQLHOSTS registry key does not reside on the local computer, then the value for the INFORMIXSQLHOSTS environment variable is the name of the Windows computer that stores the registry.

A UNIX example of setting this environment variable to another path is:

```
INFORMIXSQLHOSTS=/informix/csdk/etc/my_sqlhosts
```

Informix code page conversion:

Each time that the Informix wrapper connects to an Informix data source, the wrapper determines which code page value to use for that connection. You can have the Informix wrapper set the code page value or you can designate a code page by setting the CLIENT_LOCALE environment variable.

The environment variables that specify Informix code page conversion are set in the db2dj.ini file on your federated server.

For Informix code page conversion, you can set the following optional environment variables:

- CLIENT_LOCALE
- DB_LOCALE
- DBNLS

The Informix code page environment variables are:

CLIENT_LOCALE

Specifies the Informix locale that you want to use. Use this variable when you do not want the Informix wrapper to automatically determine the variable setting.

For example:

```
CLIENT_LOCALE=Informix_client_locale_value
```

- If the CLIENT_LOCALE variable is set in the db2dj.ini file on the federated server, then the wrapper uses the code page value in the db2dj.ini file.
- If the CLIENT_LOCALE variable is not set on the federated server, the wrapper determines the territory and the code page of the federated database. The wrapper sets the CLIENT_LOCALE variable to the closest matching Informix locale. If there is no matching Informix locale, the wrapper sets the CLIENT_LOCALE variable to the en_us.8859-1 locale for UNIX systems and to the en_us.CP1252 locale for Windows systems.

You can see the list of valid Informix locales by issuing the glfiles command on the Informix server.

Refer to the *Informix Guide to GLS Functionality* for more information about code page conversions.

DB_LOCALE

Specifies that the Informix database uses a different code page than your client locale. Use this variable when you want Informix to perform conversions between the two code pages. Set the DB_LOCALE environment variable to the name of the Informix database locale.

For example:

```
DB_LOCALE=Informix_db_locale_value
```

DBNLS

Specifies that Informix verifies that the DB_LOCALE setting matches the actual locale of the Informix database. Set this environment variable to 1.

For example:

```
DBNLS=1
```

Force Informix to perform code page conversion

The Informix database uses a different code page than your client locale and you want Informix to perform conversions between the two code pages. You need to:

1. Set Informix environment variable DB_LOCALE to the name of the Informix database locale. You set this variable in the db2dj.ini file on the federated server.

2. To verify that the DB_LOCALE setting matches the actual locale of the Informix database, set the Informix environment variable DBNLS to 1. You set this variable in the db2dj.ini file on the federated server.

Informix data that uses the Chinese code page GB 18030

To access data that uses the Chinese code page GB 18030, use the UTF-8 code page on your federated database and add the following setting to your db2dj.ini file, so that Informix correctly translates the GB 18030 data to unicode.

```
DB_LOCALE=zh_cn.GB18030-2000
```

Registering the Informix wrapper

You must register a wrapper to access Informix data sources. Wrappers are used by federated servers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.

Procedure

To register the Informix wrapper:

Issue the CREATE WRAPPER statement and specify the default name for the Informix wrapper.

For example:

```
CREATE WRAPPER INFORMIX;
```

Recommendation: Use the default wrapper name. The default wrapper name for Informix is INFORMIX. When you register the wrapper by using the default name, the federated server automatically uses the appropriate Informix wrapper library for the operating system that your federated server is running on.

If the default wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name that you choose. When you use a name that is different from the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement.

For example, to register a wrapper with the name informix_wrapper on a federated server that uses the AIX operating system, issue the following statement:

```
CREATE WRAPPER informix_wrapper LIBRARY 'libdb2informix.a';
```

The name of the wrapper library file that you specify depends on the operating system of the federated server. See the list of Informix wrapper library files for the correct library name to specify in the CREATE WRAPPER statement.

Informix wrapper library files

The Informix wrapper library files are added to the federated server when you install WebSphere Federation Server.

When you install WebSphere Federation Server, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2informix.a, libdb2informixF.a, and libdb2informixU.a. The default wrapper library file is libdb2informix.a. The other wrapper library files are used internally by the default wrapper library.

If you decide not to use the default wrapper name when you register the wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

The default directory paths and default wrapper library file names are listed in the following table.

Table 71. Informix wrapper library locations and file names

Operating system	Directory path	Library file name
AIX	/usr/opt/*install_path*/lib32/ /usr/opt/*install_path*/lib64/	libdb2informix.a
HP-UX	/opt/IBM/db2/*install_path*/lib32 /opt/IBM/db2/*install_path*/lib64	libdb2informix.sl
Linux	/opt/IBM/db2/*install_path*/lib32 /opt/IBM/db2/*install_path*/lib64	libdb2informix.so
Solaris	/opt/IBM/db2/*install_path*/lib32 /opt/IBM/db2/*install_path*/lib64	libdb2informix.so
Windows	%DB2PATH%\bin	db2informix.dll

- *install_path* is the directory path where WebSphere Federation Server is installed on UNIX or Linux.
- %DB2PATH% is the environment variable that is used to specify the directory path where WebSphere Federation Server is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

Registering the server definitions for an Informix data source

You must register each Informix server that you want to access in the federated database.

Procedure

To register a server definition for an Informix data source:

1. Locate the node name in the Informix sqlhosts file or registry.

   Sample sqlhosts file:

   ```
   inf10an onsoctcp anaconda inmx10
   inf10bo onsoctcp boa ifmx10
   inf10py onsoctcp python ifmx10
   ```

   - The first value in each line is the *node_name*, such as `inf10an`.
   - The second value in each line is the *nettype*, or type of connection. In this example `onsoctcp` indicates this is a TCP/IP connection.
   - The third value in each line is the host name, such as anaconda, boa, and python.
   - The fourth value in each line is the service name, such as `inmx10`. The service name field depends on the *nettype* listed in the second value.

   For more information about the format of the sqlhosts file and the meaning of these fields, see the Informix manual *Administrators Guide for Informix Dynamic Server*.
2. Use one of the following methods to create the server definition.
   - Use the Federated Objects wizard in the DB2 Control Center. To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.

- Issue the CREATE SERVER statement.

  For example:

```
CREATE SERVER server_definition_name TYPE informix
    VERSION version_number WRAPPER INFORMIX
    OPTIONS (NODE 'node_name', DBNAME 'database_name');
```

  Although the *'node_name'* and *'database_name'* variables are specified as options in the CREATE SERVER statement, these options are required for Informix data sources.

  After the server definition is registered, use the ALTER SERVER statement to add or drop server options.

CREATE SERVER statement - Examples for the Informix wrapper

Use the CREATE SERVER statement to register server definitions for the Informix wrapper. This topic includes a complete example with the required parameters, and examples with additional server options.

Complete example

The following example shows you how to register a server definition for an Informix wrapper by issuing the CREATE SERVER statement:

```
CREATE SERVER asia TYPE informix VERSION 10 WRAPPER INFORMIX
      OPTIONS (NODE 'abc', DBNAME 'sales');
```

asia A name that you assign to the Informix database server. Duplicate server definition names are not allowed.

TYPE *informix*
Specifies the type of data source server to which you are configuring access. For the Informix wrapper, the server type must be `informix`.

VERSION *10*
The version of the Informix database server that you want to access.

WRAPPER *INFORMIX*
The wrapper name that you specified in the CREATE WRAPPER statement.

NODE *'abc'*
The name of the node where the Informix database server resides. Obtain the node name from the sqlhosts file. This value is case sensitive.

Although the name of the node is specified as an option in the CREATE SERVER statement, it is required for Informix data sources.

DBNAME *'sales'*
The name of the Informix database that you want to access. This value is case sensitive.

Although the name of the database is specified as an option in the CREATE SERVER statement, it is required for Informix data sources.

Additional server options

When you create a server definition, you can specify additional server options in the CREATE SERVER statement. The server options can be general server options and Informix-specific server options.

FOLD_ID and FOLD_PW server options

When the federated server connects to a data source, the federated server tries to connect using all possible combinations of uppercase and lowercase for the user ID and password, as well as the current case. The federated server might make up to nine connect attempts before successfully connecting to the data source server. These attempts can slow down connect times and might result in the user ID being locked out. You can prevent lock outs by specifying values for the FOLD_ID and FOLD_PW server options. You can set the FOLD_ID and FOLD_PW server options to ′N′ (do not fold the user ID or password).

If you set the FOLD_ID and FOLD_PW server options to ′N′, you must specify the user ID and password in the correct case. The advantage to setting these server options to ′N′ is that when an invalid user ID or password is specified, the wrapper will not keep trying the various uppercase and lowercase combinations. These two server options can reduce the chance of exceeding the maximum number of failed login attempts and the ID getting locked out.

The following example shows an Informix server definition with these server options:

```
CREATE SERVER asia TYPE informix VERSION 10 WRAPPER INFORMIX
      OPTIONS (NODE 'abc', DBNAME 'sales', FOLD_ID 'N', FOLD_PW 'N');
```

IUD_APP_SVPT_ENFORCE server option example

The IUD_APP_SVPT_ENFORCE option specifies whether the federated server should enforce detecting or building of application savepoint statements. Informix does not support application savepoint statements. When set to ′N′, the federated server will not roll back transactions when an error is encountered. Your application must handle the error recovery.

The IUD_APP_SVPT_ENFORCE server option must be set to ′N′ to enable replication to or from Informix data sources. The following example shows an Informix server definition with the IUD_APP_SVPT_ENFORCE server option.

```
CREATE SERVER asia TYPE informix VERSION 10 WRAPPER INFORMIX
      OPTIONS (NODE 'abc', DBNAME 'sales', IUD_APP_SVPT_ENFORCE 'N');
```

INFORMIX_DB_LOCALE and INFORMIX_CLIENT_LOCALE options

The INFORMIX_DB_LOCALE option sets the database locale environment variable (DB_LOCALE) that is used for the connection between the federated server and the data source server. If the INFORMIX_DB_LOCALE option is not specified, the Informix DB_LOCALE environment variable is set to the value that is specified in the db2dj.ini file. If the db2dj.ini file does not specify the DB_LOCALE environment variable, the Informix DB_LOCALE environment variable is not set A valid value is any valid Informix locale. This option is optional. The default setting is None.

The INFORMIX_CLIENT_LOCALE option sets the client locale environment variable (CLIENT_LOCALE) to be used for the connection between the federated server and the data source server. If the INFORMIX_CLIENT_LOCALE option is not specified, the Informix CLIENT_LOCALE environment variable is set to the value specified in the db2dj.ini file. If db2dj.ini does not specify CLIENT_LOCALE, then the Informix CLIENT_LOCALE environment variable is set to the Informix locale that most closely matches the code page and territory of the federated database. A valid value is any valid Informix locale. This option is optional. The default setting is None.

The following example shows an Informix server definition with the INFORMIX_DB_LOCALE and INFORMIX_CLIENT_LOCALE options.

```
CREATE SERVER asia TYPE informix VERSION 10 WRAPPER INFORMIX
       OPTIONS (NODE 'abc', DBNAME 'sales', INFORMIX_DB_LOCALE 'en_us.8859-1',
      INFORMIX_CLIENT_LOCALE 'en_us.CP1252');
```

Creating the user mappings for an Informix data source

When you attempt to access an Informix server, the federated server establishes a connection to the Informix server by using a user ID and password that are valid for that data source. You must define an association (a user mapping) between each federated server user ID and password and the corresponding data source user ID and password.

About this task

Create a user mapping for each user ID that will access the federated system to send distributed requests to the Informix data source.

Procedure

To map a local user ID to the Informix server user ID and password:

Issue a CREATE USER MAPPING statement.

For example:

```
CREATE USER MAPPING FOR local_userID SERVER server_definition_name
       OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password');
```

Although the REMOTE_AUTHID and REMOTE_PASSWORD variables are specified as options in the CREATE USER MAPPING statement, these options are required to access Informix data sources.

CREATE USER MAPPING statement - Examples for the Informix wrapper

Use the CREATE USER MAPPING statement to map a federated server user ID to an Informix server user ID and password. This topic includes a complete example with the required parameters, and an example that shows you how to use the DB2 special register USER with the CREATE USER MAPPING statement.

Complete example

The following example shows how to map a federated server user ID to an Informix server user ID and password:

```
CREATE USER MAPPING FOR VINCENT SERVER asia
       OPTIONS (REMOTE_AUTHID 'vinnie', REMOTE_PASSWORD 'close2call');
```

VINCENT
Specifies the local user ID that you are mapping to a user ID that is defined at the Informix server.

SERVER *asia*
Specifies the server definition name that you registered in the CREATE SERVER statement for the Informix server.

REMOTE_AUTHID *'vinnie'*

Specifies the user ID at the Informix database server to which you are mapping *VINCENT*. Use single quotation marks to preserve the case of this value unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

Although the remote user ID is specified as an option in the CREATE USER MAPPING statement, it is required for Informix data sources.

REMOTE_PASSWORD *'close2call'*

Specifies the password that is associated with *'vinnie'*. Use single quotation marks to preserve the case of this value unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

Although the remote password is specified as an option in the CREATE USER MAPPING statement, it is required for Informix data sources.

Special register example

You can use the DB2 special register USER to map the authorization ID of the person who is issuing the CREATE USER MAPPING statement to the data source authorization ID that is specified in the REMOTE_AUTHID user option.

The following example shows a CREATE USER MAPPING statement that includes the special register USER:

```
CREATE USER MAPPING FOR USER SERVER asia
      OPTIONS (REMOTE_AUTHID 'vinnie', REMOTE_PASSWORD 'close2call');
```

Testing the connection to the Informix server

Test the connection to the Informix data source server to determine if the federated server is properly configured to access Informix data sources.

About this task

You can test the connection to the Informix server by using the server definition and user mappings that you defined.

Procedure

To test the connection to the Informix server:

Open a pass-through session and issue a SELECT statement on the Informix system tables. If the SELECT statement returns a count, your server definition and your user mapping are set up properly.

For example:

```
SET PASSTHRU server_definition_name
SELECT count(*) FROM informix.systables
SET PASSTHRU RESET
```

If the SELECT statement returns an error, you should troubleshoot the connection errors.

Performance tuning for the Informix wrapper

You can use the FOLD_ID and FOLD_PW server options to improve connectivity between the federated server and Informix data sources.

When the federated server connects to a data source, the server tries to connect using all possible combinations of uppercase and lowercase for the user ID and password. The server might make up to nine connect attempts before successfully connecting to the data source server. These attempts can slow down connect times and might result in the user ID getting locked out.

You can improve performance by specifying the values for the FOLD_ID and FOLD_PW server options.

- If all your Informix user IDs and passwords are in lowercase, setting the FOLD_ID and FOLD_PW server options with the value ′L′ can improve your connect time.

  For example:

  ```
  ALTER SERVER TYPE INFORMIX
     OPTIONS (ADD FOLD_ID 'L');
  ALTER SERVER TYPE INFORMIX
     OPTIONS (ADD FOLD_PW 'L');
  ```
- The federated server attempts each combination of uppercase and lowercase values for the user ID and password. You can reduce the chance of the maximum number of failed login attempts being exceeded by setting these options to ′N′ (do not fold the user ID and the password). If you establish these settings, then you need to always specify the user ID and password in the correct case. If an invalid user ID and password are specified, the wrapper will not keep trying the various combinations.

  For example:

  ```
  ALTER SERVER TYPE INFORMIX
     OPTIONS (ADD FOLD_ID 'N');
  ALTER SERVER TYPE INFORMIX
     OPTIONS (ADD FOLD_PW 'N');
  ```

Registering nicknames for Informix tables, views, and synonyms

For each Informix server definition that you register, you must register a nickname for each table, view, or synonym that you want to access. Use these nicknames, instead of the names of the data source objects, when you query the Informix servers.

Before you begin

Update the statistics at the Informix data source before you register a nickname. The federated database relies on the data source catalog statistics to optimize query processing. You can use the Informix UPDATE STATISTICS command, which is equivalent to the DB2 RUNSTATS command to update the data source statistics.

Procedure

To register a nickname for an Informix table, view, or synonym:

Issue the CREATE NICKNAME statement. Nicknames can be up to 128 bytes in length.

For example:

```
CREATE NICKNAME nickname FOR server_definition_name."remote_schema"."remote.table" ;
```

When you create the nickname, the federated server queries the data source catalog using the nickname. This query tests the connection to the data source table, view, or synonym. If the connection does not work, you will receive an error message.

Repeat this step for each Informix table, view, or synonym that you want to create a nickname for.

CREATE NICKNAME statement - Examples for the Informix wrapper

Use the CREATE NICKNAME statement to register a nickname for an Informix table, view, or synonym that you want to access. This topic includes a complete example with the required parameters.

Complete example

```
CREATE NICKNAME JPSALES FOR asia."vinnie"."japan" ;
```

JPSALES

A unique nickname that is used to identify the Informix table, view, or synonym.

Important: The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authorization ID of the user who registers the nickname.

asia."vinnie"."japan"

A three-part identifier for the remote object:

- *asia* is the server definition name that you assigned to the Informix database server in the CREATE SERVER statement.
- *vinnie* is the name of the owner to which the table, view, or synonym belongs unless the database is ANSI-compliant. In an ANSI-compliant database, it is the schema name.
- *japan* is the name of the remote table, view, or synonym that you want to access.

The federated server folds the names of the Informix schemas and tables to uppercase unless you enclose the names in quotation marks.

Chapter 12. Microsoft SQL Server data sources

Configuring access to Microsoft SQL Server data sources

To configure the federated server to access Microsoft SQL Server data sources, you must provide the federated server with information about the data sources and objects that you want to access.

Before you begin

- The ODBC driver must be installed and configured on the federated server.
- IBM WebSphere Federation Server must be installed on a server that acts as the federated server.
- Check the setup of the federated server.
- Check the federated parameter to ensure that federation is enabled.

About this task

You can configure the federated server to access Microsoft SQL Server data sources by using the DB2 Control Center or by issuing SQL statements on the DB2 command line.

Procedure

To configure access to Microsoft SQL Server data sources:

1. Use one of the following methods to prepare the federated server and federated database depending on your operating system.
   - Prepare the federated server and federated database (Windows).
   - Prepare the federated server and federated database (UNIX)
2. Set the environment variables for the Microsoft SQL Server wrapper.
3. Register the wrapper.
4. Register the server definition.
5. Create the user mappings.
6. Test the connection to the Microsoft SQL Server remote server
7. Register nicknames for Microsoft SQL Server tables and views.

Preparing the federated server to access Microsoft SQL Server data sources (Windows)

On federated servers that run Windows, the federated server must be able to access Microsoft SQL Server data sources. To prepare the federated server, you must verify the settings in the ODBC System DSN and test the connection to Microsoft SQL Server data sources.

Procedure

To prepare the federated server to access Microsoft SQL Server data sources:

1. Verify that the ODBC System DSN is set to connect to the Microsoft SQL Server data source. In the Control Panel, locate the existing DSN entry for the Microsoft SQL Server remote server or create a DSN entry.

The DSN entry for the Microsoft SQL Server remote server is the value that you will use for the NODE server option when you register the server definition in the federated database.

2. Use one of the following methods to test the connection to the Microsoft SQL Server data source:
   - Select **Configure** from the ODBC Data Source Administrator window.
   - Use the Microsoft SQL Server query tool.

After you complete this task, you can set the environment variables.

Preparing the federated server to access Microsoft SQL Server data sources (Linux, UNIX)

On federated servers that run Linux or UNIX, the federated server must be able to access Microsoft SQL Server data sources. To prepare the federated server, you must verify the settings in the odbc.ini file, create symbolic links, and test the connection to Microsoft SQL Server data sources.

Procedure

To prepare the federated server to access Microsoft SQL Server data sources:

1. Verify that the odbc.ini file is updated on the federated server. If the odbc.ini file does not exist on the federated server, you can create it in a text editor. Consult the documentation from the ODBC client vendor for information about the odbc.ini file.

   Remember: Place the odbc.ini file or a copy of this file in the home directory of the DB2 instance owner to ensure that it can be accessed if the instance owner is not the root user.
2. Verify that the path to the odbc.ini is in the ODBCINI environment variable.

   From an operating system command prompt, issue the following command:

   ```
   export ODBCINI=$HOME/.odbc.ini
   ```

3. Create the appropriate symbolic links:

Federated server operating system	Step

Linux	Create the following symbolic links:
	`ln -s $DJX_ODBC_LIBRARY_PATH/.. /locale/usr/local/locale`
	`ln -s $DJX_ODBC_LIBRARY_PATH/libodbcinst.so/ usr/lib/libodbcinst.so`
	If you are using the DataDirect Technologies Connect for the ODBC driver, verify the library name and create the symbolic link. The name of the library varies depending on the version of the driver and whether you are using a 32-bit or a 64-bit driver.
	For example, if you're using DataDirect Version 4.2, you would create the following link:
	`ln -s $DJX_ODBC_LIBRARY_PATH/libivicu19.so/ usr/lib/libivicu19.so`
	If you're using DataDirect Version 5.0, you would create the following link:
	`ln -s $DJX_ODBC_LIBRARY_PATH/libivicu20.so/ usr/lib/libivicu20.so`
	If you use DataDirect and do not include the symbolic link, CREATE WRAPPER MSSQLODBC3 might fail with the following error message:
	`SQL10013N The specified library name could not be loaded.`

Solaris	Create the following symbolic link: `ln -s $DJX_ODBC_LIBRARY_PATH/../ locale $HOME/sqllib/locale` $HOME is the home directory of the DB2 instance owner.

4. Run the /opt/odbc/odbc.sh script. This script sets up several operating system specific environment variables.
5. Test the connection from the federated server to the Microsoft SQL server data source by using the DataDirect Connect ODBC demoodbc utility. The demoodbc utility is located in the /demo subdirectory of the DataDirect Connect ODBC libraries.

After you complete this task, you can set the environment variables.

Setting the Microsoft SQL Server environment variables

The Microsoft SQL Server environment variables must be set in the db2dj.ini file on the federated server.

Restrictions

Review the restrictions for the db2dj.ini file before you begin this task.

About this task

The db2dj.ini file contains configuration information about the Microsoft SQL Server ODBC driver that is installed on your federated server.

There are required and optional environment variables for Microsoft SQL Server data sources.

If you installed the Microsoft SQL Server client software before you installed the Microsoft SQL Server wrapper, the required Microsoft SQL Server environment variables are set in the db2dj.ini file.

You must set the environment variables by using the steps in this task if you did not install the Microsoft SQL Server client software before you installed the Microsoft SQL Server wrapper or if you want to set any of the optional environment variables.

Procedure

To set the Microsoft SQL Server environment variables:

1. Use one of the following methods:

Method	Step
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.
Automatically set the environment variables.	Run the WebSphere Federation Server installation wizard. Follow the instructions in the wizard. **Important:** Set the required environment variables by running the installation wizard. The optional environment variables must be set manually.

Manually set the environment variables.	Edit the db2dj.ini file.
	• On federated servers that run UNIX, this file is located in the sqllib/cfg directory.
	• On federated servers that run Windows, this file is located in the %DB2PATH%\cfg directory.
	If the file does not exist, you can create a file with the name db2dj.ini by using any text editor. In the db2dj.ini file, you must specify the fully qualified path in the value of the environment variables; otherwise you will encounter errors.
	For example:
	`DJX_ODBC_LIBRARY_PATH=/opt/odbc/lib`
	`ODBCINI=/opt/odbc/.odbc.ini`

2. To ensure that the environment variables are set on the federated server, recycle the DB2 instance with these commands:

   ```
   db2stop
   db2start
   ```

After you complete this task, you can register the wrapper.

Microsoft SQL Server environment variables

There are required and optional environment variables for Microsoft SQL Server data sources. These variables are set in the db2dj.ini file.

The following environment variables are valid for Microsoft SQL Server:

- DJX_ODBC_LIBRARY_PATH
- ODBCINI
- LD_LIBRARY_PATH (Solaris only)

Variable descriptions

DJX_ODBC_LIBRARY_PATH

Specifies the directory path to the ODBC library files. This variable must also be specified on federated servers that run Solaris.

For example:

```
DJX_ODBC_LIBRARY_PATH=ODBC_driver_directory/lib
```

ODBC_driver_directory is the directory path where the ODBC driver is installed.

ODBCINI

Specifies the directory path where your ODBC configuration file (odbc.ini) is located.

For example:

```
ODBCINI=/home/db2inst1/.odbc.ini
```

Do not set the ODBCINI environment variable as a system variable.

LD_LIBRARY_PATH (Solaris only)

On federated servers that run Solaris, specifies the directory path to the ODBC library files.

For example:

```
LD_LIBRARY_PATH=ODBC_driver_directory/lib
```

Registering the Microsoft SQL Server wrapper

You must register a wrapper to access Microsoft SQL Server data sources. Federated servers use wrappers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files

Procedure

To register the Microsoft SQL Server wrapper:

Use one of the following methods:

Method	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.

Issue the CREATE WRAPPER statement and specify the default name for the Microsoft SQL Server wrapper.	For example:
	`CREATE WRAPPER MSSQLODBC3;`
	Remember: When you register the wrapper by using the default name, MSSQLODBC3, the federated server automatically uses the appropriate Microsoft SQL Server wrapper library for the operating system that your federated server is running on.
	If the default wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name that you choose. When you use a name that is different from the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement.
	For example, to register a wrapper with the name sqlserver_wrapper on a federated server that uses AIX, issue the following statement:
	`CREATE WRAPPER sqlserver_wrapper LIBRARY 'libdb2mssql3.a';`
	The wrapper library file that you specify depends on the operating system of the federated server.

After you complete this task, you can register the server definition.

Microsoft SQL Server wrapper library files

The Microsoft SQL Server wrapper library files are added to the federated server when you install the wrapper.

When you install the Microsoft SQL Server wrapper, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2mssql3.a, libdb2mssql3F.a, and libdb2mssql3U.a. The default wrapper library file is libdb2mssql3.a. The other wrapper library files are used internally by the Microsoft SQL Server wrapper.

If you do not use the default wrapper name when you register a wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

The default directory paths and default wrapper library file names are listed in the following table.

Table 72. Microsoft SQL Server client library locations and file names

Operating system	Directory path	Library file name
AIX	/usr/opt/*install_path*/lib32/ /usr/opt/*install_path*/lib64/	libdb2mssql3.a
Linux	/opt/IBM/db2/*install_path*/lib32 /opt/IBM/db2/*install_path*/lib64	libdb2mssql3.so
Solaris	/opt/IBM/db2/*install_path*/lib32 /opt/IBM/db2/*install_path*/lib64	libdb2mssql3.so
Windows	%DB2PATH%\bin	db2mssql3.dll

install_path is the directory path where IBM WebSphere Federation Server is installed on UNIX or Linux.

Registering the server definitions for a Microsoft SQL Server data source

You must register each Microsoft SQL Server remote server that you want to access in the federated database.

Procedure

To register a server definition for a Microsoft SQL Server data source:

1. Locate the node name for the Microsoft SQL Server.
   - On federated servers that run Windows, the node name is the System DSN name that you specified for the Microsoft SQL Server remote server that you are accessing.
   - On federated servers that run UNIX, the node name is defined in the .odbc.ini file.

   At the top of the .odbc.ini file, there is a section labeled ODBC Data Sources, which lists the nodes. Each of the nodes has a section in the .odbc.ini file that describes the node.

   The following example is a .odbc.ini file on AIX. The node names are [rawilson] and [medusa].

```
[ODBC Data Sources]
rawilson=MS SQL Server 2000
medusa=MS SQL Server 2000
[rawilson]
Driver=/opt/odbc/lib/ddmsss20.so
Description=MS SQL Server Driver for AIX
  Address=9.112.30.39,1433
[medusa]
Driver=/opt/odbc/lib/ddmsss20.so
Description=MS SQL Server Driver for AIX
Address=9.112.98.123,1433
[ODBC]
InstallDir=/opt/odbc
```

2. Use one of the following methods to create the server definition.

Method	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.

Method	Description

Issue the CREATE SERVER statement	For example:
	`CREATE SERVER server_definition_name`
	`TYPE MSSQLSERVER`
	`VERSION version_number`
	`WRAPPER wrapper_name`
	`OPTIONS (NODE 'node_name',`
	`DBNAME 'database_name');`
	Although the *'node_name'* and *'db_name'* variables are specified as options in the CREATE SERVER statement, these options are required for Microsoft SQL Server data sources.
	After the server definition is registered, use the ALTER SERVER statement to add or drop server options.

After you complete this task, you can create user mappings.

Creating the user mappings for a Microsoft SQL Server data source

When you attempt to access an Microsoft SQL Server remote server, the federated server establishes a connection to the Microsoft SQL Server remote server by using a user ID and password that are valid for that data source. You must define an association (a user mapping) between each federated server user ID and password and the corresponding data source user ID and password.

About this task

Create a user mapping for each user ID that will access the federated system to send distributed requests to the Microsoft SQL Server data source.

Procedure

To map a local user ID to the Microsoft SQL Server remote server user ID and password:

Issue a CREATE USER MAPPING statement.

For example:

```
CREATE USER MAPPING FOR local_userID SERVER server_definition_name
       OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password');
```

Although the REMOTE_AUTHID and REMOTE_PASSWORD variables are specified as options in the CREATE USER MAPPING statement, these options are required to access Microsoft SQL Server data sources.

After you complete this task, you can test the connection to the Microsoft SQL Server tables and views.

Testing the connection to the Microsoft SQL Server remote server

Test the connection to the Microsoft SQL Server remote server to determine if the federated server is properly configured to access Microsoft SQL Server data sources.

About this task

You can test the connection to the Microsoft SQL Server remote server by using the server definition and user mappings that you defined.

Procedure

To test the connection to the Microsoft SQL Server remote server:

Open a pass-through session and issue a SELECT statement on the Microsoft SQL Server system tables. If the SELECT statement returns a count, your server definition and your user mapping are set up properly.

For example:

```
SET PASSTHRU server_definition_name
SELECT count(*) FROM dbo.sysobjects
SET PASSTHRU RESET
```

After you complete this task, you can register nicknames.

Registering nicknames for Microsoft SQL Server tables and views

For each Microsoft SQL Server remote server definition that you register, you must register a nickname for each table or view that you want to access. Use these nicknames, instead of the names of the data source objects, when you query the Microsoft SQL Server remote servers.

Before you begin

To ensure that the federated database has current and complete statistics, execute the Microsoft SQL Server sp_createstats stored procedure and the Microsoft SQL Server CREATE STATISTICS command from the Microsoft SQL Server database before creating the nickname.

The sp_createstats stored procedure gathers statistics on all of the default columns in a table in an Microsoft SQL Server data source, but does not gather statistics for columns that appear first within an index. To ensure that the federated database has complete statistics on the Microsoft SQL Server table, you also must use the Microsoft SQL Server CREATE STATISTICS command to gather statistics for each column that appears first in an index.

When you use the CREATE STATISTICS command from the Microsoft SQL Server database, you must give the statistic the same name for the column on which the statistics are being collected. By giving the statistic the same name as the column, you ensure that when you register the nickname with the CREATE NICKNAME statement, the federated database reads the statistics collected by the Microsoft SQL Server CREATE STATISTICS command.

Procedure

To register a nickname for a Microsoft SQL Server table or view:

Use one of the following methods:

Method	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.

Issue the CREATE NICKNAME statement. Nicknames can be up to 128 characters in length.	For example:
	`CREATE NICKNAME` *`nickname`*
	`FOR` *`server_definition_name."remote_schema"."remote.table"`*`;`

When you create the nickname, the federated server queries the data source catalog using the nickname. This query tests the connection to the data source table, view, or synonym. If the connection does not work, you receive an error message.

Repeat this step for each Microsoft SQL Server table or view that you want to create a nickname for.

Using ODBC tracing information to troubleshoot connections to Microsoft SQL Server data sources

If you experience problems connecting to the data source, you can obtain ODBC tracing information to analyze and resolve the problems.

Symptom

However, activating a trace does impact system performance. You should turn off tracing after you have resolved the connectivity problems.

If you are unable to connect to the data source with the Microsoft SQL Server wrapper, running a trace might help you diagnose the problem.

Cause

The cause of the problem might be an error in the wrapper configuration.

Diagnosing the problem

To diagnose the problem on a federated server running Windows:

1. In the Control Panel, open the **Administrative Tools** folder.
2. Click **Data Sources (ODBC)** to open the ODBC Data Source Administrator window.
3. Click the Tracing tab.
4. Click **Start Tracing Now** to start the trace utility.

On a federated server running UNIX:

1. Change the odbc.ini file.

   For example, if you use the DataDirect ODBC 3.x driver, find the example of the odbc.ini file in the client directory. The odbc.ini file contains a sample of the settings that are necessary to activate the trace files:

```
[ODBC]
Trace=1
TraceFile=/home/user1/trace_dir/filename.xxx
TraceDll==ODBC_driver_directory/odbctrac.so
InstallDir=/opt/odbc
```

To turn tracing on, set the first line to `Trace=1`. To turn tracing off, set the first line to `Trace=0`. The value of the TraceFile setting is the path and file name that the federated database instance has write access to.

Resolving the problem

Check the trace log file for problems.

On Windows, open the ODBC Data Source Administrator and click the Tracing tab. The path to the trace log file is shown in the Log File Path field.

On UNIX, open the odbc.ini file. The path to the trace log file is indicated by the TraceFile setting.

CREATE USER MAPPING statement - Examples for the Microsoft SQL Server wrapper

Use the CREATE USER MAPPING statement to map a federated server user ID to a Microsoft SQL Server remote server user ID and password. This topic provides a complete example with the required parameters, and an example that shows you how to use the DB2 special register USER with the CREATE USER MAPPING statement.

The following example shows how to map a federated server user ID to a Microsoft SQL Server remote server user ID and password:

```
CREATE USER MAPPING FOR elizabeth SERVER sqlserver
      OPTIONS (REMOTE_AUTHID 'liz', REMOTE_PASSWORD 'abc123')
```

elizabeth
: Specifies the local user ID that you are mapping to a user ID that is defined at the Microsoft SQL Server remote server.

SERVER *sqlserver*
: Specifies the server definition name that you registered in the CREATE SERVER statement for the Microsoft SQL Server remote server.

REMOTE_AUTHID *'liz'*
: Specifies the user ID at the Microsoft SQL Server remote server to which you are mapping *elizabeth*. Use single quotation marks to preserve the case of this value unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

REMOTE_PASSWORD *'abc123'*
: Specifies the password that is associated with *'liz'*. Use single quotation marks to preserve the case of this value unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

DB2 special register USER

You can use the DB2 special register USER to map the authorization ID of the person who is issuing the CREATE USER MAPPING statement to the data source authorization ID that is specified in the REMOTE_AUTHID user option.

The following example shows a CREATE USER MAPPING statement that includes the special register USER:

```
CREATE USER MAPPING FOR USER SERVER sqlserver
       OPTIONS (REMOTE_AUTHID 'liz', REMOTE_PASSWORD 'abc123');
```

CREATE NICKNAME statement - Examples for the Microsoft SQL Server wrapper

Use the CREATE NICKNAME statement to register a nickname for a Microsoft SQL Server table or view that you want to access. This topic provides a complete example with the required parameters.

The following example shows how to register a nickname for a Micrsoft SQL Server table or view using the CREATE NICKNAME statement.

```
CREATE NICKNAME cust_africa FOR sqlserver."vinnie"."egypt"
```

cust_africa
A unique nickname that is used to identify the Microsoft SQL Server table or view.

Important: The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authorization ID of the user who registers the nickname.

sqlserver."vinnie"."egypt"
A three-part identifier for the remote object:

- *sqlserver* is the server definition name that you assigned to the Microsoft SQL Server remote server in the CREATE SERVER statement.
- *vinnie* is the user ID of the owner to which the table or view belongs.
- *egypt* is the name of the remote table or view that you want to access.

The federated server folds the names of the Microsoft SQL Server schemas and tables to uppercase unless you enclose the names in quotation marks.

Chapter 13. ODBC data sources

Configuring access to ODBC data sources

To configure the federated server to access ODBC data sources, you must provide the federated server with information about the data sources and objects that you want to access.

Before you begin

- The ODBC driver must be installed and configured on the server that acts as the federated server.
- IBM WebSphere Federation Server must be installed on the server that acts as the federated server.
- Check the setup of the federated server.
- Check the federated parameter to ensure that federation is enabled.
- The proper setup of the system environment variables, db2dj.ini file variables, and DB2 Profile Registry (db2set) variables. Check the documentation provided by the ODBC data source for the variables that are required for your ODBC client. The LIBPATH environment variable might be required.

Restrictions

- The ODBC wrapper cannot be used to access any DB2 family data sources. Use the DRDA wrapper to access DB2 family data sources.
- The ODBC wrapper does not support the following functions and statements:
  - LOCK TABLE statements on nicknames
  - Features deprecated in ODBC 3.x
  - X/Open or SQL/CLI drivers
  - Stored procedure nicknames
  - Statement-level atomicity enforcement using remote savepoint statements
  - WITH HOLD cursors
- For data sources that do not support positioned update and delete operations, positioned UPDATE and DELETE statements and certain searched UPDATE and DELETE statements on a nickname will fail if a unique index on non-nullable columns does not exist on the nickname or its corresponding remote table. The error SQL30090 with the reason code 21 is returned when these statements fail.
- The ODBC wrapper does not support INSERT, UPDATE, or DELETE statements against data sources that restrict the number of active statements for each connection. Consult the documentation for your data source to determine if the data source restricts the number of active statements for each connection. One of the ODBC data sources that this restriction applies to is IBM Red Brick™ Warehouse.
- The ODBC wrapper does not support operations on tables that contain columns with data types that use driver-specific SQL data type indicators. The type of operations that are not supported included the CREATE NICKNAME and SELECT statements in the pass-through mode. The ODBC wrapper supports only the SQL data type indicators that are defined by the ODBC standard in the *Microsoft ODBC Programmer's Reference*.

About this task

You can configure the federated server to access ODBC data sources by using the DB2 Control Center or by issuing SQL statements on the DB2 command line. The DB2 Control Center includes a wizard to guide you through the steps that are required to configure the required federated objects.

The data sources that are accessed through the ODBC API are referred to in this text as ODBC data sources.

Depending on your needs, you can access Excel data using the ODBC wrapper instead of using the Excel wrapper. The specific steps to configure the ODBC wrapper to access Excel data are documented in a separate topic.

Procedure

To add ODBC data sources to a federated server:

1. Use one of the following methods to prepare the federated server and federated database depending on your operating system:
   - Prepare the federated server to access data sources through ODBC (Windows).
   - Prepare the federated server to access data sources through ODBC (Linux, UNIX).
2. Register the ODBC wrapper.
3. Register the server definitions for an ODBC data source.
4. Create a user mapping for an ODBC data source.
5. Test the connection to the ODBC data source server.
6. Register nicknames for ODBC data source tables and views.

Accessing Excel data using the ODBC wrapper

You can access Microsoft Excel workbooks with the ODBC wrapper by using the Excel ODBC driver.

Before you begin

- The Excel ODBC driver must be on the federated server.
- The federated server must be able to open and read the worksheets in the Excel workbook to retrieve the data. Therefore, the Excel workbooks must be on the same computer as the federated server or on a accessible mapped network drive.
- Format the columns according to the type of data the column is expected to have.
- The data inserted in the columns must comply with the format type that is specified for the column.
- If the first eight rows of the spreadsheet have no data make sure that they are empty. To ensure that a cell is empty, open the spreadsheet in Microsoft Excel and select **Edit** → **Clear All** .
- Ensure that the data inserted into the columns in the spreadsheet comply with the specified type.

Restrictions

- The ODBC wrapper cannot access a worksheet when the workbook is already opened by a user or an application in the read/write mode. However, if the ODBC wrapper opens the workbook before a user or an application opens the workbook, the user or application can open the workbook in read-only mode.

- The Excel ODBC driver expects that the first nonblank row contains the labels for the worksheet columns. You must insert a row of column labels in the worksheet if the worksheet does not have the labels.
- Because the Excel ODBC driver is only available for Windows operating systems, you can use the ODBC wrapper to access Excel data only on federated servers that run Windows.
- You can perform insert and update operations on Excel worksheets, but you cannot perform delete operations. The Excel ODBC driver does not support delete operations. To delete data from the worksheet, you must open the worksheet in Excel to make the changes.

About this task

The Excel application does not need to be installed on the federated server. The Excel ODBC driver is automatically installed with Windows.

With the ODBC wrapper and the Excel ODBC driver, you can access data from any of the worksheets within a workbook. The Excel ODBC driver interprets a workbook as a database and interprets each worksheet within the workbook as a table.

The Excel ODBC driver supports earlier versions of Excel workbooks even if the version of Excel application that produced the workbooks is no longer supported. For example, Microsoft no longer supports worksheets created in Excel Version 4.0, but the driver supports Excel worksheets that were created in that version.

Procedure

To access Excel worksheets with the ODBC wrapper:

1. Ensure that the Excel workbook that you want to access is on the federated server or on an accessible mapped network drive.
2. If necessary, change the layout of the data in the Excel worksheets to adhere to the Excel ODBC driver requirements. Repeat this step for each worksheet or named range that you want to access.
3. If necessary, create any named ranges that you want to access.
4. Create a system DSN for the workbook that you want to access. You can use the ODBC Data Source Administrator to configure the system DSN. The name that was specified when you created the system DSN is assigned as the value for the NODE option in the CREATE SERVER statement.
5. Issue the CREATE WRAPPER statement.
6. Specify the location of the workbook by registering a server object in the federated database system catalog. For the ODBC wrapper, you need a server object for each DSN. The DSN is associated with the workbook when the Excel ODBC driver is used. The NODE *compounds_workbook_dsn* is the system DSN that you created. The NODE option is required for the ODBC wrapper to access Excel worksheets.

   To specify the location of the workbook, issue the CREATE SERVER statement and use the DSN as the system DSN for the NODE option.

   For example:

   ```
   CREATE SERVER compounds_workbook WRAPPER odbc
        OPTIONS (NODE 'compounds_workbook_dsn', PASSWORD 'n')
   ```

   Repeat this step for each workbook that you plan to access.

7. Issue the CREATE NICKNAME statement to create a nickname for the worksheet that you want to access. The syntax is:

   ```
   CREATE NICKNAME nickname FOR server_name.remote_table
   ```

8. If you created a named range to access the data, specify the name of the range as the remote_table portion of the CREATE NICKNAME statement.

   For example, if the name of the range is *testing*, the CREATE NICKNAME statement is:

   ```
   CREATE NICKNAME compounds_nickname FOR compounds_workbook.testing
   ```

   To access the data in the entire worksheet instead of a range, you specify the name of the worksheet followed by the $ symbol.

   For example, if the name of the worksheet is *Sheet1*, the CREATE NICKNAME statement is:

   ```
   CREATE NICKNAME compounds_nick FOR compounds_workbook.Sheet1$
   ```

Configuring ODBC access to WebSphere Classic Federation Server for z/OS data sources

The ODBC wrapper has been optimized to access WebSphere Classic Federation Server for z/OS data sources. The ODBC wrapper detects the ODBC driver and automatically configures the performance options.

Before you begin

The following items must be configured on your federated server:

- Federated server
- WebSphere Classic Federation Server for z/OS client
- Federated database
- Variables for the system environment db2dj.ini file, DB2 profile registry (db2set). The LIBPATH environment variable might be required. For Linux and UNIX systems, the CAC_CONFIG variable must be set to the path of the cac.ini file. For example:

  ```
  CAC_CONFIG=/home/db2inst1/cac.ini
  ```

- On AIX and Solaris systems, the DB2_FENCED wrapper option must be set to Y and the DB2_SOURCE_CLIENT_MODE wrapper option must be set to 32BIT.

For information about installation, configuration, and ODBC requirements, see the documentation that is provided with each of these products.

Restrictions

The ODBC wrapper does not support the following statement with WebSphere Classic Federation Server for z/OS data sources:

- CREATE TABLE

The following data types are not supported for use with WebSphere Classic Federation Server for z/OS data sources:

- BLOB
- CLOB
- DBCLOB
- CHAR FOR BIT DATA
- VARCHAR FOR BIT DATA

Procedure

To configure ODBC access to WebSphere Classic Federation Server for z/OS data sources:

1. Complete one of the following tasks depending on your operating system:
   - On Windows, ensure that the node name for the data source is defined as a system DSN. If you used Microsoft ODBC Data Source Administrator to define the DSN, you can use the Control Panel to verify that it is registered as a system DSN.
   - On UNIX, see the documentation for WebSphere Classic Federation Server for z/OS for instructions on how to configure the ODBC client.
2. You might need to add the data source client library path to the DB2LIBPATH registry variable to allow the client libraries to load properly. Issue the following command to set the registry variable:

   ```
   db2set DB2LIBPATH="/opt/IBM/DB2IIClassic82/cli/lib"
   ```

   ″/opt/IBM/DB2IIClassic82/cli/lib″
   The path to the library directory of your data source client.
3. Register the ODBC wrapper.
4. Register the server definitions for an ODBC data source.
5. Create a user mapping for an ODBC data source.
6. Test the connection to the ODBC data source server.
7. Register nicknames for ODBC data source tables and views.

Preparing the federated server to access data sources through ODBC (Windows)

On federated servers that run Windows, the federated server must be able to access ODBC data sources. To prepare the federated server, you must verify the settings in the ODBC System DSN.

Procedure

To prepare the federated server to access data sources through ODBC:

Verify that the ODBC 3.x driver has been installed and configured on the federated server.

The node name for the ODBC data source must be defined in the System DSN. See the ODBC driver documentation for the installation and configuration procedures.

If you used the Microsoft ODBC Data Source Administrator window to configure the DSN, you can check this setting in the Control Panel. Ensure that ODBC data source is registered as a System DSN. Otherwise the federated server might not be able to find the DSN.

After you complete this task, you can register the wrapper.

Preparing the federated server to access data sources through ODBC (Linux, UNIX)

On federated servers that run Linux or UNIX, the federated server must be able to access ODBC data sources. To prepare the federated server, you must verify the settings in the odbc.ini file, create symbolic links, and test the connection to ODBC data sources.

Procedure

To prepare the federated server to access data sources through ODBC:

1. Verify that the odbc.ini file has been updated on the federated server. If the file does not exist, you can create it in a text editor. Consult the documentation from the ODBC client vendor for information about the odbc.ini file.
2. Configure the ODBC client.

   Consult the documentation from the ODBC client vendor for instructions on how to configure the ODBC client.
3. If the client is DataDirect ODBC or RedBrick, verify that the appropriate symbolic links are created. In the following symbolic links, *ODBC_CLIENT_DIR* is the directory where the ODBC client is installed.

Federated server operating system	Step

Linux	Create the following symbolic links:
	`ln -s $ODBC_CLIENT_DIR/../locale /usr/local/locale`
	`ln -s $ODBC_CLIENT_DIR/libodbcinst.so /usr/lib/libodbcinst.so`
	If you are using the DataDirect Technologies Connect for ODBC 4.2 driver, you must also create the following symbolic link:
	`ln -s $ODBC_CLIENT_DIR/libivicu19.so /usr/lib/libivicu19.so`

Solaris	Create the following symbolic link: `ln -s $ODBC_CLIENT_DIR/../locale $HOME/sqllib/locale` $HOME is the home directory of the DB2 instance owner.

4. If the client is DataDirect ODBC you can test the connection from the federated server to the data source by using the DataDirect Connect ODBC demoodbc tool.
   a. Run the /opt/odbc/odbc.sh script. This script sets up several operating system specific environment variables.
   b. Test the connection to the ODBC data source by using the DataDirect Connect ODBC demoodbc tool. The demoodbc tool is located in the /demo subdirectory of the Connect ODBC libraries.

After you complete this task, you can register the wrapper.

Registering the ODBC wrapper

You must register a wrapper to access ODBC data sources. Federated servers use wrappers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.

Procedure

To register the ODBC wrapper:

Use one of the following methods:

Method	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.

Issue the CREATE WRAPPER statement and specify the default name for the ODBC wrapper.	For example:
	`CREATE WRAPPER ODBC;`
	Remember: When you register the wrapper by using the default name, ODBC, the federated server automatically uses the appropriate ODBC wrapper library for the operating system that your federated server is running on.
	You must specify the MODULE wrapper option on federated servers that run Linux or UNIX. The MODULE wrapper option specifies the full path of the library that contains the ODBC Driver Manager.
	For AIX and Solaris, if you are configuring access to WebSphere Classic Federation Server on z/OS data sources, you must specify the DB2_FENCED and DB2_SOURCE_CLIENT_MODE options as shown in the following example.
	`CREATE WRAPPER ODBC`
	` LIBRARY 'libdb2rcodbc.a'`
	` OPTIONS (DB2_FENCED 'Y',`
	`  DB2_SOURCE_CLIENT_MODE '32BIT',`
	` MODULE '/opt/IBM/DB2IIClassic82/cli/lib/cacsqlcli.so')`

Issue the CREATE WRAPPER statement and specify an alternative name for the ODBC wrapper.	If the default wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name that you choose. When you use a name that is different from the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement.
	For example, to register a wrapper with the name odbc_wrapper on a federated server that uses AIX, issue the following statement:
	`CREATE WRAPPER odbc_wrapper`
	`  LIBRARY 'libdb2rcodbc.a';`
	The wrapper library file that you specify depends on the operating system of the federated server.

After you complete this task, you can register the server definitions.

ODBC wrapper library files

The ODBC wrapper library files are added to the federated server when you install the wrapper.

When you install the ODBC wrapper, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files added to the directory path are libdb2rcodbc.a, libdb2rcodbcF.a, and libdb2rcodbcU.a. The default wrapper library file is libdb2rcodbc.a. The other wrapper library files are used internally by the ODBC wrapper.

If you do not use the default wrapper name when you register a wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

The default directory paths and default wrapper library file names are listed in the following table.

Table 73. ODBC client library locations and file names

Operating system	Directory path	Wrapper library file
AIX	/usr/opt/*install_path*/lib32/ /usr/opt/*install_path*/lib64/	libdb2rcodbc.a
Linux	/opt/IBM/db2/*install_path*/lib32 /opt/IBM/db2/*install_path*/lib64	libdb2rcodbc.so
Solaris	/opt/IBM/db2/*install_path*/lib32 /opt/IBM/db2/*install_path*/lib64	libdb2rcodbc.so
Windows	%DB2PATH%\bin	db2rcodbc.dll

install_path is the directory path where IBM WebSphere Federation Server is installed on UNIX or Linux.

Registering the server definitions for an ODBC data source

You must register each ODBC server that you want to access in the federated database.

Procedure

To register a server definition for an ODBC data source:

Method	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.
Issue the CREATE SERVER statement.	For example: `CREATE SERVER server_definition_name TYPE data_source_type VERSION version_number WRAPPER wrapper_name OPTIONS (NODE 'node_name', DBNAME 'database_name');` Although the name of the node is specified as an option in the CREATE SERVER statement, it is required for ODBC data sources. After the server definition is created, use the ALTER SERVER statement to add or drop server options.

After you complete this task, you can create a user mapping.

Creating a user mapping for an ODBC data source

When you attempt to access an ODBC server, the federated server establishes a connection to the ODBC server by using a user ID and password that are valid for that data source. For data sources that require a user mapping, you must define an association (a user mapping) between each federated server user ID and password and the corresponding data source user ID and password.

About this task

Create a user mapping for each user ID that will access the federated system to send distributed requests to the ODBC data source.

Procedure

To map a local user ID to the ODBC data source user ID and password:

Issue a CREATE USER MAPPING statement.

For example:

```
CREATE USER MAPPING FOR local_userID SERVER server_definition_name
       OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password');
```

Although the *remote_userID* and *remote_password* values are specified as options in the CREATE USER MAPPING statement, these options are required to access ODBC data sources.

After you complete this task, you can test the connection to the ODBC data source.

Testing the connection to the ODBC data source server

Test the connection to the ODBC data source server to determine if the federated server is properly configured to access ODBC data sources.

About this task

You can test the connection to the ODBC data source server by using the server definition and the user mappings that you defined.

Procedure

To test the connection to the ODBC data source server:

Open a pass-through session and issue a SELECT statement on the ODBC data source system tables. If the SELECT statement returns a count, your server definition and your user mapping are set up properly.

```
SET PASSTHRU server_definition_name
SELECT count(*) FROM schema_name.table_name
SET PASSTHRU RESET
```

If the SELECT statement returns an error, you should troubleshoot the connection errors

After you complete this task, you can register nicknames for the ODBC data source tables and views.

Registering nicknames for ODBC data source tables and views

For each ODBC server definition that you register, you must register a nickname for each table or view that you want to access. Use these nicknames, instead of the names of the data source objects, when you query the ODBC data sources.

Before you begin

Update the statistics at the ODBC data source before you register a nickname. The federated database relies on the data source catalog statistics to optimize query processing. Use the data source command that is equivalent to the DB2 RUNSTATS command to update the data source statistics.

Procedure

To register a nickname for an ODBC data source table or view, use one of the following methods.

Method	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.

Issue the CREATE NICKNAME statement. Nicknames can be up to 128 characters in length.	For example:
	`CREATE NICKNAME` *`nickname`*
	`FOR` *`server_definition_name."remote_schema"."remote.table"`* `;`

When you create the nickname, the federated server queries the data source catalog using the nickname. This query tests the connection to the data source table or view. If the connection does not work, you receive an error message.

Repeat this step for each ODBC table or view that you want to create a nickname for.

Optimizing ODBC wrapper performance with the ODBC tuning utility (db2fedsvrcfg)

You can optimize the performance of the ODBC wrapper with the ODBC tuning utility, db2fedsvrcfg. This utility runs a set of predefined queries against the data source and tests the results for accuracy. The utility creates a set of ALTER SERVER statements that you can run against the server to set the server options for optimal performance.

Before you begin

The following items must be configured on your federated server:

- WebSphere Federation Server
- The ODBC wrapper
- The ODBC client software
- Variables for the system environment. The ODBCINI and LIBPATH variables might be required.

For information about installation, configuration, and ODBC requirements, see the documentation that is provided with each of these products.

Procedure

To optimize ODBC wrapper performance with the ODBC tuning utility:

1. If your ODBC server has not been defined, use the DB2 Control Center or DB2 Command Line Processor to connect to the federated server and create an ODBC wrapper and server.

2. Optional: If the test tables already exist on your ODBC data source, connect to your ODBC data source and drop them.
3. Run the ODBC tuning utility from the DB2 command line with the options that you want to use. The utility might take some time to complete.
4. Verify that the utility ran successfully. If the utility ran successfully, you see the following message in the command window or in the output file that you specified:

   ```
   ALTER SERVER "DS1" OPTIONS (ADD option1, 'value1')
   ALTER SERVER "DS1" OPTIONS (ADD option2, 'value2')
   ALTER SERVER "DS1" OPTIONS (ADD option3, 'value3')
   .....

   The db2fedsvrcfg command completed successfully.
   ```

   If the command ran unsuccessfully, you see an error message indicating the reason for the error. Correct the problem and run the command again.

   You must create the test tables manually under the following circumstances:

   - If you want to use table names that are different from the default
   - If the data source that you are accessing through ODBC is read-only
   - If the ODBC tuning utility is unable to create the test tables that are required by the utility
5. Connect to the federated server where the data source is defined.
6. Use the db2look utility to save your existing server settings before you run the file that is created by the ODBC tuning utility. See the documentation for the db2look utility for information about saving your existing server settings.
7. Optional: If your ODBC server is defined, you can connect to the federated server and drop the server options. The utility creates ALTER SERVER statements in the format described in step 4. If these server options have already been added, the ALTER SERVER statements will fail.
8. Use the following command to run the ALTER SERVER statements that were generated by the utility against the federated database. The ALTER SERVER statements that were created by the ODBC tuning utility are contained in the db2fedsvrcfg.sql file.

   ```
   db2 -tvf db2fedsvrcfg.sql
   ```
9. Verify that the results of the ALTER SERVER statements. If any of the statements failed, you can modify the statements in the db2fedsvrcfg.sql file and run the statements again until they succeed.
10. After you have completed tuning the server with the ODBC tuning utility, set the PUSHDOWN server option to ’Y’ to complete the optimization process.

db2fedsvrcfg command syntax - ODBC tuning utility

Use the db2fedsvrcfg command to improve the performance of the ODBC wrapper.

Syntax

db2fedsvrcfg *-s serverName* [*-m odbcDriverManagerLibrary*] *-dsn odbcDSNname* [*-dbname dsDBname*] [*-u userid*] [*-p password*] [*-noprep*] [*-prefix tableNamePrefix*] [*-suffix tableNameSuffix*] [*-dscp codePage*] [*-v*] [*-o outputFile*] [*-h*]

Parameters

Use the command db2fedsvrcfg32 when you use 32-bit ODBC drivers on AIX or Solaris. Otherwise, use the command db2fedsvrcfg.

-dbname dsDBname
: The database name of the data source.

-dscp codePage
: The code page identifier for the data source. If this option is not specified, the utility uses the code page of the user's environment. This parameter is optional.

-dsn odbcDSNname
: The system data source name (DSN) for the data source.

-h Causes detailed help to be displayed. This parameter is optional.

-m odbcDriverManagerLibrary
: The fully qualified file name of the ODBC driver manager library. The ODBC driver manager library file name is optional for Windows.

-noprep
: Prevents the testing tables from being created at the data source before testing. This parameter is optional.

-o outputFile
: The fully qualified file name for the ODBC tuning utility output file. The output file contains the ALTER SERVER statements that are used to tune the ODBC wrapper performance. This parameter is optional. If this parameter is not specified, the output is displayed in the command window.

-p password
: The password for connecting to the data source. This parameter is optional.

-prefix tableNamePrefix
: The prefix of the ODBC data source table names that the utility uses for the analysis. If a prefix is not specified, the default prefix, IITEST, is used. This parameter is optional.

-s serverName
: The name of the federated server.

-suffix tableNameSuffix
: The suffix of the ODBC data source table names that the utility uses for the analysis. If a suffix is not specified an empty string is used.

-u userid
: The user name for connecting to the data source. This parameter is optional.

-v Specifies that the output of the utility is verbose. This parameter is optional.

Example

The following example shows the command that is run against datastore, the ODBC data source. In this example, the test tables are named ABC*n*XYZ, where *n* is a number from 1 to 7.

```
db2fedsvrcfg -s DS1 -m "/usr/lib/odbc.a"
   -dsn datastore -dbname db1 -u authid -p password -noprep
     -prefix ABC -suffix XYZ -o "/home/user1/db2fedsvrcfg.sql"
```

Test table definitions for the ODBC tuning utility (db2fedsvrcfg)

In some cases, you must create the test tables for the ODBC tuning utility manually. The test table definitions are described in this topic.

You must create the test tables for the ODBC tuning utility under the following circumstances:

- If you want to use table names that are different from the default
- If the data source that you are accessing through ODBC is read-only
- If the ODBC tuning utility is unable to create the test tables that are required by the utility

The following table definition applies to all seven of the test tables that are needed (IITEST1 through IITEST7). The default table name prefix is IITEST and the default suffix is an empty string. If you specify a different prefix and suffix, you must specify the -prefix and -suffix options when you run the ODBC tuning utility.

Table 74. ODBC tuning utility test table definition for the table IITEST1

Column name	SQL data type	SQL data type identifier	Length
IT1C1	integer	SQL_INTEGER	
IT1C2	integer	SQL_INTEGER	
IT1C3	char(1)	SQL_CHAR	1
IT1C4	char(3)	SQL_CHAR	3
IT1C5	char(10)	SQL_CHAR	10
IT1C6	varchar(10)	SQL_VARCHAR	10
IT1C7	char(100)	SQL_CHAR	100

Table 75. ODBC tuning utility test table definition for the table IITEST2

Column name	SQL data type	SQL data type identifier	Length
IT2C1	integer	SQL_INTEGER	
IT2C2	integer	SQL_INTEGER	
IT2C3	char(30)	SQL_CHAR	30

Table 76. ODBC tuning utility test table definition for the tables IITEST3 through IITEST7

Number of columns in each table	Column name	SQL data type	SQL data type identifier	Length
66	IT*x*C*n*	char(100)	SQL_CHAR	100

x The number corresponding to the table that is being defined.

n The column number.

For example, IT3C1 is the column name for the first column in table IITEST3.

CREATE SERVER statement - Examples of the ODBC wrapper

Use the CREATE SERVER statement to register server definitions for the ODBC wrapper. This topic provides a complete example with the required parameters, and an example with additional server options.

The following example shows you how to register a server definition for a MySQL data source by issuing the CREATE SERVER statement:

```
CREATE SERVER mysql_server TYPE mysql
       VERSION 4.0 WRAPPER wrapper_name
        OPTIONS (NODE 'odbc_node', DBNAME 'venice')
```

mysql_server

A name that you assign to the ODBC data source server. Duplicate server definition names are not allowed.

TYPE *mysql*

Specifies the type of data source server to which you are configuring access. This parameter is optional.

VERSION *4.0*

The version of the ODBC data source that you want to access. This parameter is optional.

WRAPPER *wrapper_name*

The wrapper name that you specified in the CREATE WRAPPER statement.

NODE *'odbc_node'*

The name of the node (the system DSN name) that was assigned to the ODBC data source when the DSN was defined. On federated servers that run Windows, this value must be the name of a system DSN in the ODBC Data Source Administrator window. On federated servers that run UNIX, the name of the node is the DSN defined in the ODBC configuration file. The ODBC configuration file is usually called odbc.ini.

Although the name of the node is specified as an option in the CREATE SERVER statement, it is required for ODBC data sources.

DBNAME *'venice'*

Optional. The name of the ODBC data source that you want to access.

Server options

When you create the server definition, you can specify additional server options in the CREATE SERVER statement. The server options can be general server options and ODBC-specific server options.

Some ODBC data sources (for example, MySQL) cannot process quotation marks around table names and column names in SQL statements. To access these data sources, you must include the following server options in the CREATE SERVER statement:

- DB2_TABLE_QUOTE_CHAR ’ ` ’
- DB2_ID_QUOTE_CHAR ’ ` ’
- DB2_AUTHID_QUOTE_CHAR ’ ` ’

The ` character is the delimiter for identifiers such as schema names, table names, and column names.

For example:

```
CREATE SERVER mysql_server TYPE mysql
       VERSION 4.0 WRAPPER wrapper_name
       OPTIONS (NODE 'mysql_node', DB2_TABLE_QUOTE_CHAR '`',
       DB2_ID_QUOTE_CHAR '`' DB2_AUTHID_QUOTE_CHAR '`')
```

CREATE USER MAPPING statement - Examples for the ODBC wrapper

Use the CREATE USER MAPPING statement to map a federated server user ID to an ODBC data source user ID and password. This topic provides a complete example with the required parameters, and an example that shows you how to use the DB2 special register USER with the CREATE USER MAPPING statement.

The following example shows how to map a federated server user ID to an ODBC data source user ID and password:

```
CREATE USER MAPPING FOR arturo SERVER mysql_server
      OPTIONS (REMOTE_AUTHID 'art', REMOTE_PASSWORD 'red4blue')
```

arturo Specifies the local user ID that you are mapping to a user ID that is defined at the ODBC data source.

mysql_server
Specifies the server definition name that you defined in the CREATE SERVER statement for the ODBC data source.

'art' Specifies the user ID at the ODBC data source to which you are mapping *arturo*. Use single quotation marks to preserve the case of this value unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

'red4blue'
Specifies the password that is associated with *'art'*. Use single quotation marks to preserve the case of this value unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

DB2 special register USER

You can use the DB2 special register USER to map the authorization ID of the person who is issuing the CREATE USER MAPPING statement to the data source authorization ID that is specified in the REMOTE_AUTHID user option.

The following example shows a CREATE USER MAPPING statement that includes the special register USER:

```
CREATE USER MAPPING FOR USER SERVER mysql_server
      OPTIONS (REMOTE_AUTHID 'art', REMOTE_PASSWORD 'red4blue');
```

CREATE NICKNAME statement - Examples for the ODBC wrapper

Use the CREATE NICKNAME statement to register a nickname for an ODBC table or view that you want to access. This topic provides a complete example with the required parameters.

The following example shows how to register a nickname for an ODBC table or view using the CREATE NICKNAME statement.

```
CREATE NICKNAME cust_europe FOR mysql_server."vinnie"."italy"
```

cust_europe
A unique nickname that is used to identify the ODBC table or view. The nickname must be unique within the schema.

Important: The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authorization ID of the user who registers the nickname.

If your ODBC data source does not support schemas, omit the schema from the CREATE NICKNAME statement.

mysql_server."vinnie"."italy"
A three-part identifier for the remote object:

- *mysql_server* is the server definition name that you assigned to the ODBC data source server in the CREATE SERVER statement.
- *vinnie* is the user ID of the owner to which the table or view belongs.
- *italy* is the name of the remote table or view that you want to access.

The federated server folds the names of the ODBC schemas and tables to uppercase unless you enclose the names in quotation marks.

CREATE WRAPPER statement - Examples for the ODBC wrapper

Use the CREATE WRAPPER statement to register the ODBC wrapper. This topic provides examples for Linux, UNIX and Windows.

In the following examples, *odbc_wrapper* is the name that you assign to the wrapper that you are registering in the federated database.

Linux and Solaris federated server

The following example shows you how to register a wrapper with the default name on a federated server that runs Linux or Solaris:

```
CREATE WRAPPER odbc OPTIONS (MODULE '/opt/lib/odbc.so');
```

You must specify the MODULE wrapper option on federated servers that run Linux or UNIX. The MODULE wrapper option specifies the full path of the library that contains the ODBC Driver Manager.

The following example shows you how to register a wrapper with an alternative name on a federated server the runs Linux or Solaris:

```
CREATE WRAPPER odbc_wrapper LIBRARY 'libdb2rcodbc.so'
     OPTIONS (MODULE '/opt/lib/odbc.so');
```

AIX federated server

The following example shows you how to register a wrapper with the default name on a federated server that runs AIX:

```
CREATE WRAPPER odbc
     OPTIONS (MODULE '/usr/lib/odbc.a');
```

You must specify the MODULE wrapper option on federated servers that run UNIX. The MODULE wrapper option specifies the full path of the library that contains the ODBC Driver Manager.

The following example shows you how to register a wrapper with an alternative name on a federated server that runs AIX:

```
CREATE WRAPPER odbc_wrapper LIBRARY 'libdb2rcodbc.a'
     OPTIONS (MODULE '/usr/lib/odbc.a');
```

WebSphere Classic Federation Server for z/OS (AIX)

The following example shows you how to register an ODBC wrapper by issuing the CREATE WRAPPER statement on an AIX operating system. On AIX and

Solaris, the DB2_FENCED and DB2_SOURCE_CLIENT_MODE options must be specified as shown in the following example.

```
CREATE WRAPPER odbc
  OPTIONS (DB2_FENCED 'Y', DB2_SOURCE_CLIENT_MODE '32BIT',
  MODULE '/opt/IBM/DB2IIClassic82/cli/lib/cacsqlcli.so');
```

You must specify the MODULE wrapper option on federated servers that run UNIX. The MODULE wrapper option specifies the full path of the library that contains the ODBC Driver Manager.

The following example shows you how to register a wrapper for accessing WebSphere Classic Federation Server for z/OS data sources with an alternative name:

```
CREATE WRAPPER odbc_wrapper LIBRARY 'libdb2rcodbc.a'
  OPTIONS (DB2_FENCED 'Y', DB2_SOURCE_CLIENT_MODE '32BIT',
  MODULE '/opt/IBM/DB2IIClassic82/cli/lib/cacsqlcli.so');
```

Windows federated server

The following example shows you how to register a wrapper with the default name on a federated server that runs Windows:

```
CREATE WRAPPER odbc;
```

The following example shows you how to register a wrapper with an alternative name on a federated server that runs Windows:

```
CREATE WRAPPER odbc_wrapper LIBRARY 'db2rcodbc.dll';
```

Chapter 14. OLE DB data sources

Registering the OLE DB wrapper

You must register a wrapper to access OLE DB data sources. Federated servers use wrappers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.

About this task

The OLE DB wrapper enables you to access OLE DB providers that are compliant with Microsoft OLE DB version 2.0 (or later).

The OLE DB wrapper is used only to assist you in registering user-defined OLE DB external table functions in the federated database. Unlike other wrappers, the OLE DB wrapper does not use nicknames to access data that is stored in data sources.

Procedure

To register the OLE DB wrapper:

Use one of the following methods to register the OLE DB wrapper:

Method	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.

Issue the CREATE WRAPPER statement and specify the default name for the OLE DB wrapper.	For example:
	`CREATE WRAPPER OLEDB`
	Remember: When you register the wrapper by using the default name, OLEDB, the federated server automatically uses the appropriate OLE DB wrapper library for the operating system that your federated server is running on.
	If the default wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name that you choose. When you use a name that is different from the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement.
	For example, to register a wrapper with the name oledb_wrapper on a federated server that uses Windows, issue the following statement:
	`CREATE WRAPPER oledb_wrapper`
	`  LIBRARY 'db2oledb.dll'`
	The wrapper library file that you specify depends on the operating system of the federated server.

After you complete this task, you can register the server definition.

OLE DB wrapper library files

The OLE DB wrapper library files are added to the federated server when you install the wrapper.

When you install the OLE DB wrapper, the library file is added to the default directory path.

If you do not use the default wrapper name when you register the wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the library file name.

The default directory path and default wrapper library file name is listed in the following table.

Table 77. OLE DB client library locations and file names

Operating system	Directory path	Library file name
Windows	%DB2PATH%\bin	db2oledb.dll

Registering the server definitions for an OLE DB data source

You must register each OLE DB data source server that you want to access in the federated database.

About this task

You register a server definition for and OLE DB data source from the DB2 command line.

Procedure

To register a server definition for an OLE DB data source:

Issue the CREATE SERVER statement.

For example:

```
CREATE SERVER server_definition_name WRAPPER wrapper_name
      OPTIONS (CONNECTSTRING 'keyword=value;keyword=value');
```

Although the CONNECTSTRING variable is specified as an option in the CREATE SERVER statement, this option is required for OLE DB data sources.

After you complete this task, you can create a user mapping.

Creating the user mappings for an OLE DB data source

When you attempt to access an OLE DB server, the federated server establishes a connection to the OLE DB server by using a user ID and password that are valid for that data source. You must define an association (a user mapping) between each federated server user ID and password and the corresponding data source user ID and password.

About this task

Create a user mapping for each user ID that will access the federated system.

Procedure

To map a local user ID to the OLE DB data source user ID and password:

Issue a CREATE USER MAPPING statement.

For example:

```
CREATE USER MAPPING FOR local_ userid SERVER server_definition_name
      OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password');
```

Although the REMOTE_AUTHID and REMOTE_PASSWORD variables are specified as options in the CREATE USER MAPPING statement, these options are required to access OLE DB data sources.

If the length of either the password on the OLE DB data source or the password on the federated server is less than eight characters, SQL statements that access the OLE DB data source will fail. The following error message appears:

```
SQL30082N Attempt to establish connection failed with security reason "15"
   ("PROCESSING FAILURE"). SQLSTATE=08001
```

To avoid this problem, either change the OLE DB data source password or change the password on the federated server to eight or more characters.

CREATE SERVER statement - Examples for the OLE DB wrapper

Use the CREATE SERVER statement to register server definitions for the OLE DB wrapper. This topic provides a complete example with the required parameters, and an example with additional server options.

The following example shows you how to register a server definition for an OLE DB wrapper by issuing the CREATE SERVER statement:

```
CREATE SERVER Nwind WRAPPER OLEDB
      OPTIONS (CONNECTSTRING 'Provider=Microsoft.Jet.OLEDB.4.0;
      Data Source=c:\msdasdk\bin\oledb\nwind.mdb');
```

Nwind A name that you assign to the OLE DB data source. Duplicate server definition names are not allowed.

WRAPPER *OLEDB*
The wrapper name that you specified in the CREATE WRAPPER statement.

CONNECTSTRING *'Provider=Microsoft.Jet.OLEDB.4.0; Data Source=c:\msdasdk\bin\oledb\nwind.mdb'*
Provides the initialization properties that are necessary to connect to an OLE DB provider.

The value for the CONNECTSTRING option contains a series of keyword and value pairs that are separated by semicolons. The equal sign (=) separates each keyword and its value. Keywords are the descriptions of the OLE DB initialization properties (property set DBPROPSET_DBINT) or provider-specific keywords.

For the complete syntax and semantics for the CONNECTSTRING option, see the *Microsoft OLE DB 2.0 Programmer's Reference and Data Access SDK*, Microsoft Press, 1998.

Server options

When you create the server definition, you can specify additional server options in the CREATE SERVER statement. The server options can be general server options and OLE DB-specific server options.

The COLLATING_SEQUENCE server option specifies whether the data source uses the same collating sequence as the federated server. The default setting for the COLLATING_SEQUENCE server option is ′N′. When the OLE DB data source uses the same collating sequence as the federated server, set the COLLATING_SEQUENCE server option is ′Y′.

The following example an OLE DB server definition that includes the COLLATING_SEQUENCE server option:

```
CREATE SERVER Nwind WRAPPER OLEDB
       OPTIONS (CONNECTSTRING 'Provider=Microsoft.Jet.OLEDB.4.0;
       Data Source=c:\msdasdk\bin\oledb\nwind.mdb',
     COLLATING_SEQUENCE 'Y')
```

CREATE USER MAPPING statement - Examples for the OLE DB wrapper

Use the CREATE USER MAPPING statement to map a federated server user ID to an OLE DB data source user ID and password. This topic includes a complete example with the required parameters, and an example that shows you how to use the DB2 special register USER with the CREATE USER MAPPING statement.

The following example shows how to map a federated server user ID to an OLE data source user ID and password:

```
CREATE USER MAPPING FOR laura SERVER Nwind
       OPTIONS (REMOTE_AUTHID 'lulu', REMOTE_PASSWORD 'raiders')
```

laura Specifies the local user ID that you are mapping to a user ID that is defined at the OLE DB data source.

SERVER *Nwind*
Specifies the server definition name that you registered in the CREATE SERVER statement for the OLE DB server.

REMOTE_AUTHID *′lulu′*
Specifies the user ID at the OLE DB database server to which you are mapping *lulu*. Use single quotation marks to preserve the case of this value unless you set the FOLD_ID server option to ′U′ or ′L′ in the CREATE SERVER statement.

Although the remote user ID is specified as an option in the CREATE SERVER statement, it is required for OLE DB data sources.

REMOTE_PASSWORD *′raiders′*
Specifies the password that is associated with *′lulu′*. Use single quotation marks to preserve the case of this value unless you set the FOLD_PW server option to ′U′ or ′L′ in the CREATE SERVER statement.

Although the remote password is specified as an option in the CREATE SERVER statement, it is required for OLE DB data sources.

DB2 special register USER

You can use the DB2 special register USER to map the authorization ID of the person who is issuing the CREATE USER MAPPING statement to the data source authorization ID that is specified in the REMOTE_AUTHID user option.

The following example shows a CREATE USER MAPPING statement that includes the special register USER:

```
CREATE USER MAPPING FOR USER SERVER Nwind
       OPTIONS (REMOTE_AUTHID 'lulu', REMOTE_PASSWORD 'raiders');
```

Chapter 15. Oracle data sources

Configuring access to Oracle data sources

To configure a federated system to access Oracle data sources, you must provide the federated system with information about the data sources and objects that you want to access.

Before you begin

- The Oracle client software must be installed and configured on the server that acts as the federated server.
- IBM WebSphere Federation Server must be installed on a server that acts as the federated server.
- Check the setup of the federated server.
- Check the federated parameter to ensure that federation is enabled.

Procedure

To add Oracle data sources to a federated system:

1. Set the Oracle environment variables.
2. Set up and test the Oracle client configuration file.
3. Register the Oracle wrapper.
4. Register the server definitions for an Oracle data source.
5. Create the user mappings.
6. Test the connection to the Oracle server.
7. Register the nicknames.

Setting the Oracle environment variables

The Oracle environment variables must be set in the db2dj.ini file on the federated server.

Restrictions

Review the restrictions for the db2dj.ini file.

About this task

The db2dj.ini file contains configuration information about the Oracle client software that is installed on the federated server.

There are required and optional environment variables for Oracle data sources.

If you installed the Oracle client software before you installed the Oracle wrapper, the required Oracle environment variables are set in the db2dj.ini file.

If you did not install the Oracle client software before you installed the Oracle wrapper, or if you want to set any of the optional environment variables you must set the environment variables by using the steps in this task.

Procedure

To set the Oracle environment variables:

1. Use one of the following methods:

Method	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.
Automatically set the environment variables.	Run the WebSphere Federation Server installation wizard. Follow the instructions in the wizard. **Important:** Set the required environment variables by running the installation wizard. The optional environment variables must be set manually.
Manually set the environment variables.	Edit the db2dj.ini file: The db2dj.ini file is located in the directory that the DB2 registry variable DB2_DJ_INI specifies. When the DB2_DJ_INI variable is not set, the db2dj.ini file is in one of the following default paths depending on the operating system: • On Linux and UNIX: *instancehome*/sqllib/cfg/db2dj.ini. *instancehome* The home directory of the instance owner. • On Windows: %DB2PATH%\cfg\db2dj.ini **%DB2PATH%** The directory where the DB2 database system is installed, for example, C:\Program Files\IBM\sqllib. If the file does not exist, you can create a file with the name db2dj.ini by using any text editor. In the db2dj.ini file, you must specify the fully qualified path in the value of the environment variables; otherwise you will encounter errors.

2. On federated servers that run UNIX, add the Oracle environment variable to the .profile file of the DB2 instance. For example:

   ```
   export ORACLE_HOME=oracle_home_directory
   export PATH=$ORACLE_HOME/bin:$PATH
   ```

   Where *oracle_home_directory* is the directory where the Oracle client software is installed.

3. On federated servers that run Linux or UNIX, execute the DB2 instance .profile file by entering:

   ```
   . $HOME/ .profile
   ```

4. To ensure that the environment variables are set on the federated server, recycle the DB2 instance with these commands:

   ```
   db2stop
   db2start
   ```

After you complete this task, you can set up and test the Oracle client.

Oracle environment variables

There are required and optional environment variables for Oracle data sources. These variables are set in the db2dj.ini file.

The following environment variables are valid for Oracle::

- ORACLE_HOME
- ORACLE_BASE (optional)
- ORA_NLS (optional)
- NLS_LANG (optional)
- TNS_ADMIN (optional)

Variable descriptions

ORACLE_HOME

Set the ORACLE_HOME environment variable to the directory path where the Oracle client software is installed. Specify the fully qualified path for the environment variable: ORACLE_HOME=*oracle_home_directory*. For example, if the Oracle home directory is \usr\oracle\8.1.7, the entry in the db2dj.ini file is `ORACLE_HOME=\usr\oracle\8.1.7`.

If an individual user of the federated instance sets the ORACLE_HOME environment variable locally, the federated instance does not use that setting. The federated instance uses only the value of ORACLE_HOME that is set in the db2dj.ini file.

ORACLE_BASE

ORACLE_BASE represents the root of the Oracle client directory tree. If you set the ORACLE_BASE environment variable when you installed the Oracle client software, set the ORACLE_BASE environment variable on the federated server.

For example:

```
ORACLE_BASE=oracle_root_directory
```

ORA_NLS

If multiple versions of Oracle are running on your system, you must ensure that:

- The appropriate ORA_NLS environment variable is set
- The corresponding NLS data files for the versions that you are using are available

The location-specific data is stored in a directory that is specified by the ORA_NLS environment variable. Each version of Oracle has a different ORA_NLS data directory.

Table 78. ORA_NLS environment variable by version

Oracle version	Environment variable
8.x, 9.x	ORA_NLS33
10.x	ORA_NLS10

For example, on federated servers that run UNIX that access Oracle 8.1 data sources, the ORA_NLS33 environment variable setting is:

```
ORA_NLS33=oracle_home_directory/ocommon/nls/admin/<data>
```

NLS_LANG

The NLS_LANG environment variable is a code page environment variable. Refer to the Oracle NLS documentation for information about setting this variable.

TNS_ADMIN

On federated servers that run Windows

The Oracle client looks for the tnsnames.ora file in the %ORACLE_HOME%\NETWORK\ADMIN directory, where `%ORACLE_HOME%` is defined in the db2dj.ini file. If the tnsnames.ora file is not in the %ORACLE_HOME%\NETWORK\ADMIN directory, you must set the TNS_ADMIN environment variable in the db2dj.ini file on the federated server. You set the environment variable in the db2dj.ini file to the path where the tnsnames.ora file is located.

On federated servers that run AIX or Linux

The Oracle client looks for the tnsnames.ora file in the /etc directory. If the tnsnames.ora file is not in the /etc directory, then the Oracle client looks for the tnsnames.ora file in the $ORACLE_HOME/network/admin directory, where `$ORACLE_HOME` is defined in db2dj.ini file. If the tnsnames.ora file is not in the $ORACLE_HOME/network/admin directory, you must set the TNS_ADMIN environment variable on the federated server. You set the environment variable in the db2dj.ini file to the path where the tnsnames.ora file is located.

For example, if the tnsnames.ora file is in the /home/oracle directory, you set the environment variable to:

```
TNS_ADMIN=/home/oracle
```

On federated servers that run Solaris

The Oracle client looks for the tnsnames.ora file in the /var/opt/oracle directory. If the tnsnames.ora file is not in the /var/opt/oracle directory, then the Oracle client looks for the tnsnames.ora file in the $ORACLE_HOME/network/admin directory, where `$ORACLE_HOME` is defined in db2dj.ini file. If the tnsnames.ora file is not in the $ORACLE_HOME/network/admin directory, you must set the TNS_ADMIN environment variable. You set the variable in the db2dj.ini file to the path where the tnsnames.ora file is located.

For example, if the tnsnames.ora file is in the `/home/oracle` directory, you set the environment variable to:

```
TNS_ADMIN=/home/oracle
```

Oracle code page conversion

Each time that the Oracle wrapper connects to an Oracle data source, the wrapper determines which code page value to use for that connection. You can specify that the Oracle wrapper set the code page value or you can designate a code page by setting the NLS_LANG environment variable.

The environment variables that specify Oracle code page conversion are set in the db2dj.ini file on your federated server.

If the NLS_LANG environment variable is set in the db2dj.ini file on the federated server, the wrapper uses the code page value in the db2dj.ini file.

If the NLS_LANG environment variable is not set in the db2dj.ini file on the federated server, the wrapper determines the territory and the code page of the federated database. The wrapper sets the NLS_LANG environment variable to the closest matching Oracle locale. If there is no closely matching locale, the NLS_LANG environment variable is set to American_America.US7ASCII.

See the documentation that accompanies your Oracle software for a list of valid locales.

Example of Chinese code page GB 18030:

If you access an Oracle data source that contains data that is encoded with the Chinese code page GB 18030, and your federated database uses the UTF-8 code page, the Oracle wrapper sets the Oracle NLS_LANG environment variable to:

```
NLS_LANG=Simplified Chinese_China.UTF8
```

This setting is correct if you are using an Oracle 8 client, but if you are using the Oracle Version 9i (or later) client, you must set the NLS_LANG environment variable to override the default setting of the Oracle wrapper. Set the NLS_LANG environment variable to Simplified Chinese_China.AL32UTF8, so that the Oracle 9i client translates the GB 18030 data into Unicode correctly.

For example:

```
NLS_LANG=Simplified Chinese_China.AL32UTF8
```

Setting up and testing the Oracle client configuration file

The Oracle client configuration file is used to connect to Oracle databases, by using the client libraries that are installed on the federated system.

About this task

The client configuration file specifies the location of each Oracle database server and type of connection (protocol) for the database server.

The default name for the Oracle client configuration file is tnsnames.ora.

The default location of the client configuration file depends on the operating system that is used by the federated system:

- On Linux and UNIX systems, the default location of the file is $ORACLE_HOME/network/admin.
- On Windows systems, the default location of the file is %ORACLE_HOME%\NETWORK\ADMIN.

Procedure

To set up and test the Oracle client configuration file:

1. Create the tnsnames.ora using the Oracle NET8/NET Configuration utility that comes with the Oracle client software.

   Within the tnsnames.ora file, the SID (or SERVICE_NAME) is the name of the Oracle instance, and the HOST is the host name where the Oracle server is located.
2. If you want to place the tnsnames.ora file in a path other than the default search path, set the TNS_ADMIN environment variable to specify the file location:
   a. Edit the db2dj.ini file in the sqllib/cfg directory, and set the TNS_ADMIN environment variable. For example:

      ```
      TNS_ADMIN=x:/path/
      ```
   b. Issue the following commands to recycle the DB2 instance and ensure that the environment variable is set in the program:

```
db2stop
db2start
```

3. Test the connection to ensure that the client software can connect to the Oracle server. Use the Oracle sqlplus utility to test the connection.

After you complete this task, you can register the Oracle wrapper.

Registering the Oracle wrapper

You must register a wrapper to access Oracle data sources. Federated servers use wrappers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.

Procedure

To register an Oracle wrapper:

Use one of the following methods:

Method	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.

Issue the CREATE WRAPPER statement and specify the default name for the Oracle wrapper.	For example:
	`CREATE WRAPPER NET8 ;`
	Remember: When you register the wrapper by using the default name, NET8, the federated server automatically uses the appropriate Oracle wrapper library for the operating system that your federated server is running on.
	If you do not use the default wrapper name when you register a wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.
	For example, to register a wrapper with the name oracle_wrapper on the federated server that uses AIX, issue the following statement:
	`CREATE WRAPPER oracle_wrapper LIBRARY 'libdb2net8.a'`
	The wrapper library file that you specify depends on the operating system of the federated server.

After you complete this task, you can register the server definitions.

Oracle wrapper library files

The Oracle wrapper library files are added to the federated server when you install the wrapper.

When you install the Oracle wrapper, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the following directory path are libdb2net8.a, libdb2net8F.a, and libdb2net8U.a, The default wrapper library file is libdb2net8.a. The other wrapper library files are used internally by the Oracle wrapper.

The default directory paths and default wrapper library file names are listed in the following table.

Table 79. Oracle wrapper library locations and file names

Operating system	Directory path	Library file names
AIX	/usr/opt/*install_path*/lib32/ /usr/opt/*install_path*/lib64/	libdb2net8.a
Linux	/opt/IBM/db2/*install_path*/lib32 /opt/IBM/db2/*install_path*/lib64	libdb2net8.so
Solaris	/opt/IBM/db2/*install_path*/lib32 /opt/IBM/db2/*install_path*/lib64	libdb2net8.so
Windows	%DB2PATH%\bin	db2net8.dll

install_path is the directory path where IBM WebSphere Federation Server is installed on UNIX or Linux.

Registering the server definitions for an Oracle data source

You must register each Oracle server that you want to access in the federated database.

Procedure

To register a server definition for an Oracle data source:

1. Locate the node name in the Oracle tnsnames.ora file. Example tnsnames.ora file:

   ```
   paris_node =
     (DESCRIPTION =
      (ADDRESS_LIST =
         (ADDRESS = (PROTOCOL = TCP)(HOST = somehost)(PORT = 1521)))
        (CONNECT_DATA = (SERVICE_NAME = ora9i.seel)))
   ```

   In this example, the node value to use in the CREATE SERVER statement is *paris_node*.

   Although the *node_name* is specified as an option in the CREATE SERVER SQL statement, it is required for Oracle data sources.

2. To register the server definitions for an Oracle data source, use one of the following methods:

Method	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.

Method	Description
Issue the CREATE SERVER statement.	For example: `CREATE SERVER server_name TYPE oracle VERSION 8.1.7 WRAPPER net8 OPTIONS (NODE 'node_name')` After the server definition is created, use the ALTER SERVER statement to add or drop server options.

The next task in this sequence of tasks is creating the user mappings for an Oracle data source.

Creating the user mappings for an Oracle data source

To access an Oracle server, the federated server establishes a connection to the Oracle server by using a user ID and password that are valid for that data source. You must define an association (a user mapping) between each federated server user ID and password and the corresponding data source user ID and password.

Restrictions

The user ID at the Oracle data source must be created by using the Oracle create user command with the `identified by` clause, instead of the `identified externally` clause.

About this task

Create a user mapping for each user ID that will access the federated system to send distributed requests to the Oracle data source.

Procedure

To create the user mappings for an Oracle data source:

Issue a CREATE USER MAPPING statement. For example:

```
CREATE USER MAPPING FOR local_userID SERVER server_definition_name
      OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password') ;
```

Although the REMOTE_AUTHID and REMOTE_PASSWORD variables are specified as options in the CREATE USER MAPPING statement, they are required to access Oracle data sources.

After you complete this task, test the connection to the Oracle server.

Testing the connection to the Oracle server

Test the connection to the Oracle server to determine if the federated server is properly configured to access Oracle data sources.

About this task

You can test the connection to the Oracle server by using the server definition and user mappings that you defined.

Procedure

To test the connection to the Oracle server:

Open a pass-through session and issue a SELECT statement on the Oracle system tables. If the SELECT statement returns a count, your server definition and your user mapping are set up properly.

For example:

```
SET PASSTHRU remote_server_name
SELECT count(*) FROM sys.all_tables
SET PASSTHRU RESET
```

If the SELECT statement returns an error, you should troubleshoot the connection errors.

After you complete this task, you can register the nicknames for the Oracle tables and views.

Troubleshooting connectivity problems with Oracle data sources

The most common problem that you might encounter when you configure the federated server to access Oracle data sources is connectivity.

Symptom

If you are not able to connect to an Oracle data source from a federated server, you might need to update the TCP/IP hosts file.

Cause

This problem can be caused by an out of date TCP/IP hosts file.

Resolving the problem

For each host in the DESCRIPTION section of the tnsnames.ora file, you might need to update the TCP/IP hosts file. Whether you update this file depends on how TCP/IP is configured on your network. Part of the network must translate the remote host name that is specified in the DESCRIPTION section in the tnsnames.ora file to an address.

If your network has a named server that recognizes the host name, you do not need to update the TCP/IP hosts file. If your network does not have a named server that recognizes the host name, you need to add an entry in the TCP/IP hosts file for the remote host.

The location of the TCP/IP hosts file depends on the operating system that is running on the federated server:

On federated servers that run Linux or UNIX
The hosts file is in the /etc/hosts directory.

On federated servers that run Windows
The hosts file is in the x:\winnt\system32\drivers\etc\hosts directory.

Registering nicknames for Oracle tables and views

For each Oracle server definition that you register, you must register a nickname for each table or view that you want to access. Use these nicknames, instead of the names of the data source objects, when you query the Oracle servers.

Before you begin

Update the statistics at the Oracle data source before you register a nickname. The federated database relies on the data source catalog statistics to optimize query processing. Use the data source command that is equivalent to the DB2 RUNSTATS command to update the data source statistics.

About this task

For data source objects that use Oracle Label Security, the nickname data cannot be cached. You can turn caching on or off using the ALTER NICKNAME statement.

Procedure

To register a nickname for an Oracle table or view, use one of the following methods:

Method	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.

Issue the CREATE NICKNAME statement. Nicknames can be up to 128 characters in length.	For example:
	`CREATE NICKNAME nickname`
	`FOR server_definition_name."remote_schema"."remote.table" ;`

When you create the nickname, the federated server queries the data source catalog by using the nickname. This query tests the connection to the data source table or view. If the connection does not work, you receive an error message.

Repeat this step for each Oracle table or view that you want to create a nickname for.

Troubleshooting connectivity problems with Oracle data sources

The most common problem that you might encounter when you configure the federated server to access Oracle data sources is connectivity.

Symptom

If you are not able to connect to an Oracle data source from a federated server, you might need to update the TCP/IP hosts file.

Cause

This problem can be caused by an out of date TCP/IP hosts file.

Resolving the problem

For each host in the DESCRIPTION section of the tnsnames.ora file, you might need to update the TCP/IP hosts file. Whether you update this file depends on how TCP/IP is configured on your network. Part of the network must translate the remote host name that is specified in the DESCRIPTION section in the tnsnames.ora file to an address.

If your network has a named server that recognizes the host name, you do not need to update the TCP/IP hosts file. If your network does not have a named server that recognizes the host name, you need to add an entry in the TCP/IP hosts file for the remote host.

The location of the TCP/IP hosts file depends on the operating system that is running on the federated server:

On federated servers that run Linux or UNIX
The hosts file is in the /etc/hosts directory.

On federated servers that run Windows
The hosts file is in the x:\winnt\system32\drivers\etc\hosts directory.

CREATE SERVER statement - Examples for the Oracle wrapper

Use the CREATE SERVER statement to register server definitions for the Oracle wrapper. This topic provides a complete example with the required parameters, and an example with additional server options.

The following example shows you how to register a server definition for an Oracle wrapper by issuing the CREATE SERVER statement:

```
CREATE SERVER oraserver TYPE oracle VERSION 8.1.7 WRAPPER wrapper_name
      OPTIONS (NODE 'paris_node') ;
```

oraserver
A name that you assign to the Oracle database server. Duplicate server definition names are not allowed.

TYPE *oracle*
Specifies the type of data source server to which you are configuring access. For the Oracle wrapper, the server type must be `oracle`.

VERSION *8.1.7*
The version of Oracle database server that you want to access.

WRAPPER *wrapper_name*
The wrapper name that you specified in the CREATE WRAPPER statement.

NODE *'paris_node'*
The name of the node where the Oracle database server resides. Obtain the node name from the tnsnames.ora file. This value is case sensitive.

Although the name of the node is specified as an option in the CREATE SERVER statement, it is required for Oracle data sources.

Server options

When you create the server definition, you can specify additional server options in the CREATE SERVER statement. These server options can be general server options and Oracle-specific server options.

The federated server assumes that all of the Oracle VARCHAR columns contain trailing blanks. If you are certain that all of the VARCHAR columns in the Oracle database do not contain trailing blanks, you can set a server option to specify that the data source use a non-blank padded VARCHAR comparison semantic.

The following example shows an Oracle server definition with the VARCHAR_NO_TRAILING_BLANKS server option:

```
CREATE SERVER oraserver TYPE oracle VERSION 8.1.7 WRAPPER wrapper_name
      OPTIONS (NODE 'paris_node', VARCHAR_NO_TRAILING_BLANKS 'Y') ;
```

Use the VARCHAR_NO_TRAILING_BLANKS server option when none of the columns contains trailing blanks. If only some of the VARCHAR columns contain trailing blanks, you can set an option on those columns with the ALTER NICKNAME statement.

CREATE USER MAPPING statement - Examples for the Oracle wrapper

Use the CREATE USER MAPPING statement to map a federated server user ID to an Oracle server user ID and password. This topic includes a complete example with the required parameters, and an example that shows you how to use the DB2 special register USER with the CREATE USER MAPPING statement.

The following example shows how to map a federated server user ID to an Oracle server user ID and password:

```
CREATE USER MAPPING FOR robert SERVER oraserver
      OPTIONS (REMOTE_AUTHID 'rob', REMOTE_PASSWORD 'then4now') ;
```

robert Specifies the local user ID that you are mapping to a user ID that is defined at the Oracle server.

SERVER *oraserver*
Specifies the server definition name that you registered in the CREATE SERVER statement for the Oracle server.

REMOTE_AUTHID *'rob'*
Specifies the user ID at the Oracle database server to which you are mapping *robert*. Use single quotation marks to preserve the case of this value unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

Although the remote user ID is specified as an option in the CREATE SERVER statement, it is required for Oracle data sources.

REMOTE_PASSWORD *'then4now'*
Specifies the password that is associated with *rob*. Use single quotation marks to preserve the case of this value unless you set the FOLD_PW server option to U or L in the CREATE SERVER statement.

Although the remote password is specified as an option in the CREATE SERVER statement, it is required for Oracle data sources.

DB2 special register USER

You can use the DB2 special register USER to map the authorization ID of the person who is issuing the CREATE USER MAPPING statement to the data source authorization ID that is specified in the REMOTE_AUTHID user option.

The following example shows a CREATE USER MAPPING statement that includes the special register USER:

```
CREATE USER MAPPING FOR USER SERVER oraserver
      OPTIONS (REMOTE_AUTHID 'rob', REMOTE_PASSWORD 'then4now') ;
```

CREATE NICKNAME statement - Examples for the Oracle wrapper

Use the CREATE NICKNAME statement to register a nickname for an Oracle table or view that you want to access. This topic provides a complete example with the required parameters.

The following example shows you how to register a nickname for an Oracle table or view using the CREATE NICKNAME statement.

```
CREATE NICKNAME PARISINV FOR oraserver."vinnie"."inventory" ;
```

PARISINV
A unique nickname that is used to identify the Oracle table or view. The nickname consists of the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authorization ID of the user who registers the nickname.

oraserver."vinnie"."inventory"
A three-part identifier for the remote object:

- *oraserver* is the server definition name that you assigned to the Oracle database server in the CREATE SERVER statement.
- *vinnie* is the user ID of the owner to which the table or view belongs.
- *inventory* is the name of the remote table or view that you want to access.

The federated server changes the names of the Oracle schemas and tables to uppercase unless you enclose the names in quotation marks.

Chapter 16. Script data sources

Configuring access to scripts as data sources

To configure a federated system to access scripts as data sources, you must register custom functions, a wrapper, a server definition, and nicknames for the scripts.

Before you begin

- IBM WebSphere Federation Server must be installed on a server that acts as the federated server.
- A federated database must exist on the federated server.

About this task

You can configure a federated server to access data through scripts by issuing SQL statements on the DB2 command line.

Procedure

To add scripts as data sources to a federated server:

1. Identify or write a script.
2. Register the custom function.
3. Configure the script daemon.
4. Start the script daemon.
5. Register the script wrapper.
6. Register the server definition for a data source that is accessed by a script.
7. Register nicknames for data sources.

Script wrapper overview

You can use scripts, such as Perl scripts, to access information in databases or to generate data. You can integrate that data with data from other federated data sources by using the script wrapper.

You might have existing scripts that return data from data sources such as life sciences data banks, or that generate data on their own. The script wrapper enables the use of scripts as if they are federated data sources. The script wrapper enables access to data that can then be integrated by using a federated system. The scripts must return results in XML.

Script wrapper nicknames can include input and output columns. These nicknames use function templates in predicates to pass input values to the script. Output data from the script is represented as XML in a hierarchical form, which can then map to nicknames by using primary and foreign keys.

The script wrapper is a read-only wrapper. The script wrapper cannot write data to a data source.

In the following diagram, data flows from a script through the script wrapper and script daemon to the federated database, where the data can be integrated with data from other sources and viewed with the federated client. Optionally, the federated system can access the script through a proxy server and firewall.

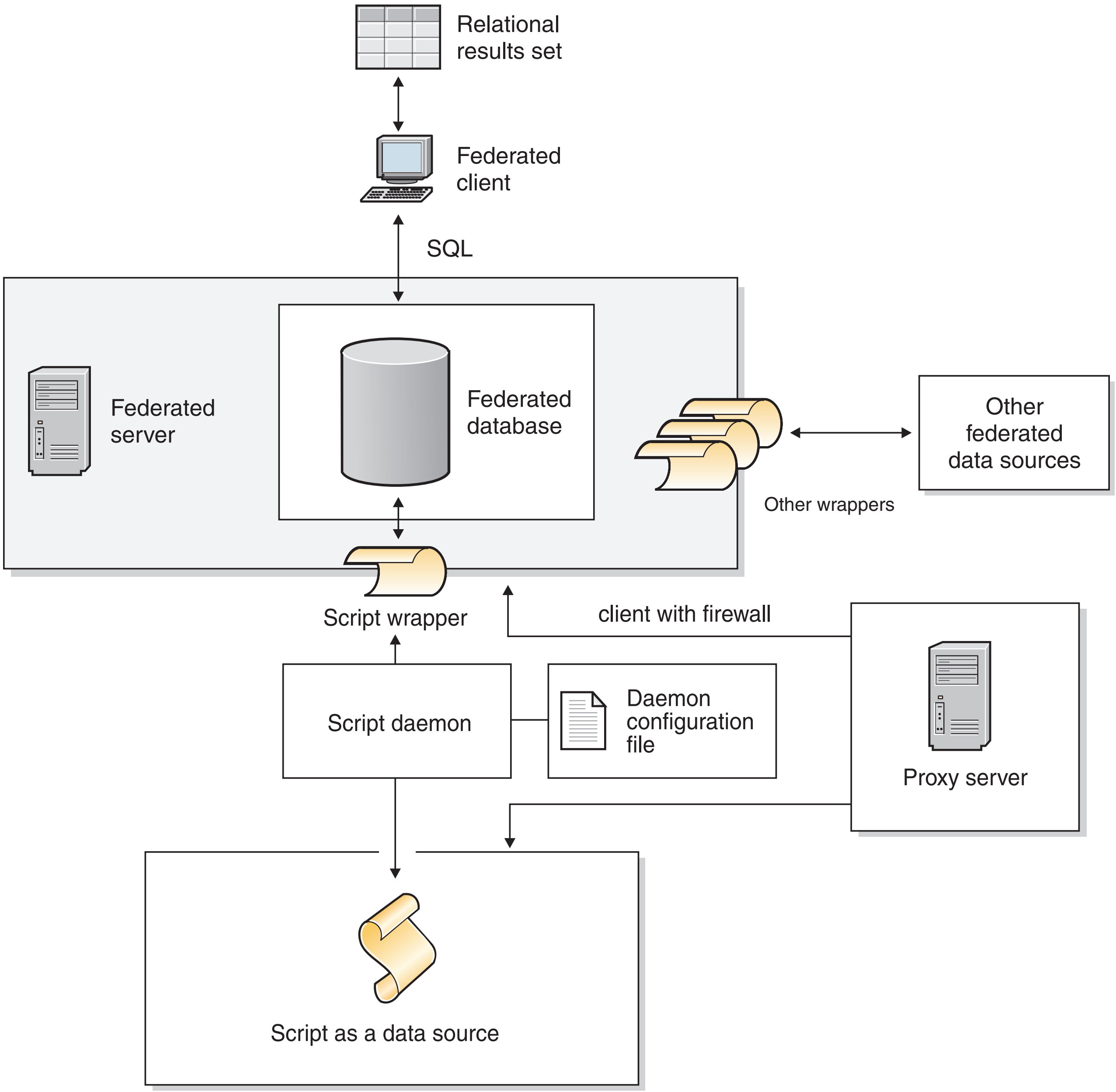


Figure 21. The script wrapper in a federated system

A script is invoked from the directory that contains the script daemon. If the script retrieves data from its own data file and cannot locate the data file, the script might include relative paths. Use absolute paths in scripts.

Adding scripts as data sources to a federated system

To configure a federated server to access scripts as data sources, you must configure the script daemon, a wrapper, a server definition, and nicknames for the scripts.

Registering the custom function for the script

You must register the script custom function WSSCRIPT.ARGS before you register the script wrapper.

About this task

You must register the custom function on each federated database instance where the script wrapper is installed.

The custom function for the script wrapper must be registered with the schema name WSSCRIPT.

You must include specific keywords when you register the custom function for the script wrapper. Include the AS TEMPLATE, DETERMINISTIC, and NO EXTERNAL ACTION keywords in the CREATE FUNCTION statement.

The federated environment uses two query engines. For the script wrapper, these query engines are the federated database query engine and the script wrapper query engine. You can specify that predicates get pushed down to the script wrapper engine by using the script wrapper custom functions in the WHERE clause of your SELECT statement.

The create_function_mappings.ddl file in the sqllib/samples/lifesci/script directory on the federated server specifies the data types for the custom function.

Procedure

To register the script custom function:

Run the create_function_mappings.ddl file on each federated database instance where the script wrapper is installed.

The following example shows the syntax for the WSSCRIPT.ARGS function:

```
CREATE FUNCTION WSSCRIPT.ARGS (input_column_data_type(), input_column_data_type())
    RETURNS INTEGER AS TEMPLATE
    DETERMINISTIC NO EXTERNAL ACTION;
```

Data types for the custom function for the script wrapper

You must specify the data type of the input column twice in each custom function.

Register a separate WSSCRIPT custom function for each valid data type for the column_name argument. The WSSCRIPT custom function has the following data types:

- VARCHAR
- INTEGER
- CLOB
- DOUBLE
- DATE

Both parameters to the custom function must be of the same data type, which is the data type of the corresponding input column. When used in a query predicate, the first parameter is the name of the switch input column. The second column is the value to be passed to the script for this switch.

Configuring the script daemon

The script wrapper requires a script daemon that listens for script job requests from the script wrapper. The script daemon must be configured before you register the script wrapper.

Before you begin

The script daemon has the following prerequisites:

- Has write access to a directory in which the daemon can write temporary files.
- Runs on a server that you can access through TCP/IP from your federated system. This server can be the same server that operates as the federated server or a separate script server.
- Requires a configuration file that must be on the same server as the script daemon.
-
- Runs separately from the script wrapper and the federated database.

Procedure

To configure the script daemon:

1. Ensure that the executable files for the script daemon are on the correct server. You might need to copy the script daemon executable files to another server.

   During the installation of IBM WebSphere Federation Server, the script daemon executable files are installed on the federated server. The name and location of the file is as follows:

   UNIX db2script_daemon is installed in the $DB2PATH/bin directory. $DB2PATH is the directory in which the federated server is installed.

   Windows
   db2script_daemon.exe is installed in the %DB2PATH%\bin directory. %DB2PATH% is the directory in which the federated server is installed, usually C:\SQLLIB\bin.

   If you use a separate script server, copy the script daemon executable and configuration files from the federated server to the script server. The script daemon executable files can run in any directory on the script server that does not contain spaces in the names in the directory path.
2. Ensure that the configuration file for the script daemon is on the correct server.

   During the installation of the federated system, a sample configuration file for the script daemon is installed on the federated server. The name of the sample configuration file is SCRIPT_DAEMON.config. The location of the file is as follows:

   UNIX The daemon configuration file is installed in the $DB2PATH/bin directory.

   Windows
   The daemon configuration file is installed in the %DB2PATH%\bin directory.

   By default, the daemon looks for the configuration file in the working directory from which the daemon is started. You can copy the configuration file to another location. If you use a script server, copy the daemon configuration file from the directory on the federated server to a directory on the script server. You can copy the daemon configuration file to any directory on the script server that the daemon can access.

3. Edit the sample configuration file for the script daemon.
   a. Rename the configuration file so that you can use the sample file again.
   b. Ensure that the first line in the configuration file is an equal sign (=). If the equal sign is missing, the daemon does not start. An error message will indicate that the DAEMON_PORT was not specified.
   c. Ensure that the last line in the configuration file ends with a new line.

      The sample configuration file that is provided with the federated system ends with a new line character. If the last line does not end with a new line character, you receive an error message when you try to run your first script query that uses the data source that is listed on the last line.
   d. Ensure that there are no extra spaces after directory paths or at the end of the configuration file.
   e. Specify the following options in the configuration file. For options that require paths, you can specify relative paths. Relative paths are relative to the directory from which the daemon process is started.

      DAEMON_PORT=*port_number*
      The network port on which the daemon listens for script job requests that are submitted by the wrapper. The default value is 4099.

      MAX_PENDING_REQUESTS=*number_of_requests*
      The maximum number of script job requests that can be blocking on the daemon at any one time. This number does not represent the number of script jobs that are running concurrently, only the number of job requests that can block at one time. Set this value to a number greater than five. The script daemon does not restrict the number of script jobs that can run concurrently.

      DAEMON_LOGFILE_DIR=*dir*
      The directory in which the daemon creates its log file. This file contains status and error information that is generated by the script daemon.

      SCRIPT_OUT_DIR_PATH=*path*
      The directory in which the daemon creates the temporary file to store the script output data. The daemon reads data from this file and passes the data back to the wrapper through the network connection. After the data is passed to the wrapper, the daemon cleans up the temporary file

      script specification entry=*entry*
      A list of entries that specifies the name and location of the scripts that can be invoked by the script wrapper. The entry has this format:

      ```
      script_name=fully-qualified_script_path
      ```

      The following examples apply to the designated operating system:

      UNIX For example, to specify a script that accesses an Oracle data source, add the following line to the daemon configuration file:

      ```
      oracle=/dsk/1/data/oracle
      ```

Windows

For example, to specify a script that accesses an Oracle data source, add the following line to the daemon configuration file:

```
oracle=c:\data\oracle.a
```

The following example shows a SCRIPT_DAEMON.cfg file for four scripts:

```
=
DAEMON_PORT=4099
MAX_PENDING_REQUESTS=10
DAEMON_LOGFILE_DIR=./
SCRIPT_OUT_DIR_PATH=./
fee=/home/user_id/fee
fie=/home/user_id/fie
foe=/home/user_id/foe
fum=/home/user_id/fum
```

The sample configuration file for the script daemon provides an example of configuring the script daemon.

Starting the script daemon

Before you can access data sources with scripts, you must start the script daemon.

Before you begin

You must have write access to all the paths that are listed in the daemon configuration file for the DAEMON_LOGFILE_DIR and SCRIPT_OUT_DIR_PATH options.

About this task

The executable file for the script daemon starts a new process in which the script daemon runs.

Procedure

To start the script daemon:

1. Open the directory where the daemon executable file is located.
2. Issue the db2script_daemon command to run the executable files, including appropriate options.

The following example includes options:

```
db2script_daemon -a action -c config_file -d debug_level -u user_id -p password
```

db2script_daemon command - options and examples

The db2script_daemon command starts the script daemon. You can start the script daemon with several options.

The db2script_daemon command can be used on either UNIX or Windows servers. Some of the options listed in the syntax can be used only on Windows servers.

The options that are specified with the start action affect only the current instance of the daemon and override the values that are specified with the install action.

Options - db2script_daemon command

The db2script_daemon command has the following options:

-a *action* **(Windows only)**
Performs the specified activity. The valid actions are status, install, start, stop, and remove.

-c *config_file*
Instructs the daemon service to use the specified configuration file. If you do not specify the configuration file, the daemon searches for the SCRIPT_DAEMON.config file in the directory where the daemon executable files are installed. You can use this option with the install and start actions.

-d *[***1 | 2 | 3***]*
Sets the debugging level for the daemon service to the specified value. A value of 1 turns on logging, 2 traces all commands, and 3 turns on logging, traces all commands, and saves all temporary files to capture XML output. You can use this option with the install and start actions.

-u *user_id* **(Windows only)**
Sets the daemon service to run under the specified user ID. You can use this option with the install action.

-p *password* **(Windows only)**
Specifies the password for the specified user ID. The password is valid and required only when you specify the -u option. If the -p option is not specified when you set the -u option, the program prompts you for the password. You can use this option with the install action.

Examples - db2script_daemon command

The following examples show how to use the script daemon options.

Start the daemon
To start the daemon on UNIX, issue the following command:

```
db2script_daemon
```

This command assumes that the daemon configuration file is in the same directory as the executable file.

To install and start the daemon on Windows, issue the following commands:

```
db2script_daemon -a install
db2script_daemon -a start
```

Specify the daemon configuration file
If you changed the name of the daemon configuration file or if the configuration file is not in the same directory as the daemon executable file, you must use the -c option when you run the executable file. This option specifies the directory path and name for the daemon configuration file.

In this example, the daemon configuration information is in a file called SCRIPT_D.config in the subdirectory cfg on a UNIX server. Issue the following command:

```
db2script_daemon -c cfg/SCRIPT_D.config
```

Specify the debugging level

If you want to start the daemon with debugging turned on with a debugging level of 2, issue the following commands:

```
db2blast_daemon -a install -d 2
db2blast_daemon -a start
```

Check the status of the daemon (Windows)

To check the status of the daemon on a Windows server, issue the following command:

```
db2blast_daemon -a status
```

This command assumes that the daemon configuration file is in the same directory as the executable file.

Stop the daemon

To stop the daemon on UNIX, list the process ID of the daemon by issuing the following command:

```
ps -ef | grep db2script
```

Then use the process ID to stop the daemon by issuing the following command:

```
kill process_ID
```

This command assumes that the daemon configuration file is in the same directory as the executable file.

To stop the daemon on Windows, issue the following command:

```
db2script_daemon -a stop
```

Remove the daemon

You can remove the script daemon when you no longer want to use the script wrapper.

To remove the daemon, issue the following command:

```
db2script_daemon -a remove
```

Registering the script wrapper

You must register the script wrapper to access data sources with scripts. The script wrapper is implemented as a library file.

Procedure

To register the script wrapper:

Issue the CREATE WRAPPER statement, specifying a name for the script wrapper and the name of the wrapper library file.

For example, to register a wrapper with the name script_wrapper on a federated server that uses AIX, issue the following statement:

```
CREATE WRAPPER script_wrapper LIBRARY 'libdb2lsscript.a';
```

The name of the wrapper library file that you specify depends on the operating system of the federated server.

There are no script-wrapper specific options for the script wrapper CREATE WRAPPER statement. The wrapper runs unfenced by default.

Script wrapper library file

To register the script wrapper, specify the script wrapper library file for the operating system of the federated server.

When you install IBM WebSphere Federation Server, a script wrapper library file is added to the default directory path.

The default directory paths and default wrapper library file names are listed in the following table.

Table 80. Script wrapper library locations and file names

Operating system	Directory path	Wrapper library file name
AIX	/usr/opt/*install_path*/lib	libdb2lsscript.a
Linux	/opt/IBM/db2/*install_path*/lib	libdb2lsscript.so
Solaris	/opt/IBM/db2/*install_path*/lib	libdb2lsscript.so
Windows	%DB2PATH%\bin	db2lsscript.dll

- *install_path* is the directory path where WebSphere Federation Server is installed on UNIX or Linux.
- %DB2PATH% is the is the environment variable that is used to specify the directory path where WebSphere Federation Server is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

Registering the server definition for a script as a data source (DB2 command line)

You must register each server that you want to access in the federated database.

Procedure

To register a server definition for a script:

Issue the CREATE SERVER statement.

For example:

```
CREATE SERVER script_server WRAPPER script_wrapper
    OPTIONS (NODE 'myserver.example.com', DAEMON_PORT '4099');
```

The NODE and DAEMON_PORT server options are required for scripts as data sources.

After the server definition is registered, use the ALTER SERVER statement to add or drop server options.

CREATE SERVER statement - examples for the script wrapper

The examples show the use of required options and additional server options.

Required options example

The following example shows you how to register a server definition for the script wrapper by issuing the CREATE SERVER statement:

```
CREATE SERVER server1_scriptn WRAPPER script_wrapper
   OPTIONS(NODE 'big_rs.company.com');
```

server1_scriptn
: A name that you assign to the script server. Duplicate server definition names are not allowed.

WRAPPER *script_wrapper*
: The wrapper name.

NODE *'big_rs.company.com'*
: Host name of the system on which the script daemon process is running. This value is case sensitive.

Although the name of the node is specified as an option in the CREATE SERVER statement, it is required for the script wrapper.

Optional option example

The following example shows an additional server option that you can specify when you register a server definition for the script wrapper:

```
CREATE SERVER server1_scriptn  WRAPPER script_wrapper
   OPTIONS(NODE 'big_rs.company.com', DAEMON_PORT '4088');
```

DAEMON_PORT *'4088'*
: Specifies the number of the port that the daemon listens on for script job requests. The port number must be the same number that you specified in the DAEMON_PORT option of the daemon configuration file. The default port number is 4099.

Registering nicknames for scripts (DB2 command line)

You must register a separate nickname for each script. Use these nicknames when you query the data source that is accessed by a script.

About this task

The script wrapper associates XML data to nicknames. Parent and child nicknames correspond to the root and nested elements in an XML document. The parent and child nicknames are connected by primary and foreign keys that are specified in the CREATE NICKNAME statement. Each nickname is defined by XPath expressions that identify the XML data elements and specify how to extract column values from each element.

The data source is specified by the CREATE NICKNAME statement and associated with a script name with the DATASOURCE nickname option. You must create a column for each input argument to be passed to the script. Use input column options to control the syntax for switches on the command line. The value for each switch must be included in the query predicate by using the ARGS custom function at run time.

Simple scripts that do not require command-line arguments do not need input columns.

Procedure

To register a nickname for a script:

Issue the CREATE NICKNAME statement. Nicknames can be up to 128 characters long.

For example:

```
CREATE NICKNAME nickname
    (
     column_name data_type OPTIONS ('nickname_column_ options'),
     column_name data_type OPTIONS ('nickname_column_ options'),
     column_name data_type OPTIONS ('nickname_column_ options')
     )
     FOR SERVER server_definition_name
     OPTIONS (nickname_options);
```

Issue the statement for each script that you want to create a nickname for.

CREATE NICKNAME statement - examples for the script wrapper

The examples show you how to use the CREATE NICKNAME statement to register nicknames for the script wrapper.

The following example creates a parent nickname for the XML data that is returned from a script that is named fee:

```
CREATE NICKNAME customers
(
    argle  double  OPTIONS(SWITCH '-argle', POSITION 1, DEFAULT 1.0 ),
    argfile CLOB() OPTIONS(SWITCH '-file', INPUT_MODE 'FILE_INPUT', POSITION 2),
    argpos varchar()  OPTIONS(SWITCH ' ', POSITION 3),
    id        VARCHAR(5)    OPTIONS(XPATH './@id')
    name      VARCHAR(16)   OPTIONS(XPATH './name'),
    address   VARCHAR(30)   OPTIONS(XPATH './address/@street'),
    cid       VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(PRIMARY_KEY 'YES'))
    FOR SERVER script_server
    OPTIONS(DATASOURCE 'fee',
          XPATH '/doc/customer', STREAMING 'YES');
```

The following example creates a nickname that is called orders. The nickname orders is a child of the nickname that is called customers, which is created in the previous example:

```
CREATE NICKNAME orders
(
    amount INTEGER       OPTIONS(XPATH './amount'),
    date   VARCHAR(10)   OPTIONS(XPATH './date'),
    oid    VARCHAR(16)   OPTIONS(PRIMARY_KEY 'YES'),
    cid    VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(FOREIGN_KEY 'CUSTOMERS'))
    FOR SERVER script_server
    OPTIONS( XPATH './order');
```

Script wrapper nickname options

You can specify options when you create a nickname for a script. Only the root nickname can include input columns.

Nickname options

The following list describes the nickname options:

DATASOURCE
The name of the script that will be invoked. The script must be listed in the script daemon configuration file. This option is required for data sources in the parent nickname. This option applies only to the root nickname.

NAMESPACES
A comma-separated list of name=value pairs that the wrapper uses to resolve namespace prefixes in the nickname XPath expression.

TIMEOUT
The maximum time, in minutes, that the script wrapper waits for results from the script daemon. The default value is 60 minutes. This option applies only to the root nickname.

VALIDATE
Specifies whether the XML source document must be validated before the XML data is extracted. If this option is set to YES, the DB2 database system verifies that the structure of the source document conforms to an XML schema or to a document type definition (DTD). This option is accepted only for columns of the root nickname (the nickname that identifies the elements at the top level of the XML document). The default value is NO.

The XML source document is not validated if the script wrapper cannot locate the XML schema file or DTD file (.xsd or .dtd). The DB2 database system does not issue an error message if the validation does not occur. Ensure that the XML schema file or DTD file is in the location that is specified in the XML source document. Do not set the VALIDATE parameter to YES if you set the STREAMING parameter to YES.

STREAMING
Specifies whether the XML source document is separated into logical fragments that correspond to the node that matches the XPath expression of the nickname. The script wrapper then processes the XML source data fragment by fragment, reducing total memory use. This option is accepted only for columns of the root nickname (the nickname that identifies the elements at the top level of the XML document). The default streaming value is NO. Do not set the STREAMING parameter to YES if you set the VALIDATE parameter to YES.

XPATH
An XPath expression that identifies the XML element that represents individual tuples in the data source. The wrapper evaluates the XPATH nickname option for a child nickname in the context of the path that is specified by the XPATH nickname option of the parent nickname. This XPath expression is used as a context for evaluating column values that are identified by the XPATH nickname column options.

Nickname column options

The nickname column options are described in the following:

DEFAULT
The default value for an input column. This option applies only to input columns.

The default value is used if no value is provided by the SQL query. This option is not required.

FOREIGN_KEY
Indicates that this nickname is a child nickname and specifies the corresponding parent nickname.

A nickname can have at most one FOREIGN_KEY column option. The value for this option is case-sensitive. The column that is designated by the FOREIGN_KEY option contains a key that is generated by the wrapper. The value of the column cannot be retrieved in a SELECT query, and the XPATH option must not be specified. The column can only be used to join parent nicknames and child nicknames. A CREATE NICKNAME statement with a FOREIGN_KEY option fails if the parent nickname has a different schema name. Unless the nickname that is referred to in a FOREIGN_KEY clause is explicitly defined to be lowercase or mixed case by enclosing it in quotation marks under the corresponding CREATE NICKNAME statement, then you must specify the nickname in uppercase when you refer to this nickname in the FOREIGN_KEY clause.

Foreign key columns must be designated as FOR BIT DATA and NOT NULL.

INPUT_MODE

Specifies the input mode for a column. Valid values are CONFIG or FILE_INPUT. The wrapper passes the specified value to the script daemon.

CONFIG

The value is treated as a configurable parameter.

FILE_INPUT

A file is created that stores the value and the file name is passed as the command line argument.

POSITION

An integer value for positional parameters. The position sequence order starts at 1. This option applies only to input columns.

If the positional value is set to an integer, then this input must be in this position in the command line. If this option is set, the switch is inserted into the appropriate location when the query is run. If POSITION is set to -1, the option is added as the last command line option. For example, if a column value must be at the end of the command line and there is no SWITCH option, setting the value of POSITION to -1 adds the value at the end of the command line. POSITION integer values cannot be duplicated in a nickname. This option is not required.

PRIMARY_KEY

Indicates that this nickname is a parent nickname. The column data type must be VARCHAR(16). A nickname can have at most one PRIMARY_KEY column option. The only valid value is Y.

The column that is designated as PRIMARY_KEY contains a key that is generated by the wrapper. The value of the column cannot be retrieved in a SELECT query, and the XPATH option must not be specified. The column can only be used to join parent nicknames and child nicknames. Primary key columns must be designated as FOR BIT DATA and NOT NULL.

SWITCH

A character string to specify a parameter for the script on the command line. This option applies only to input columns.

On the command line, the value of this option precedes the column value that is supplied by WSSCRIPT.ARGS or the default value, if any. If the value for the switch is an empty string and a default value exists for the column, the default value is added without any SWITCH information when the command line is generated. If no default value is provided and no value for the column is provided by the SQL query, then this input column is ignored when the command line is generated. This option is required for an input column.

SWITCH_ONLY

Enables the use of switches without a command line argument.

If the SWITCH_ONLY option is specified with a value of Y, then valid input values are Y or N. For an input value of Y, only the switch is added to the command line. For an input value of N, no value is added to the command line.

VALID_VALUES
A semicolon-separated set of valid values for a column.

XPATH
Specifies the XPath expression in the XML document that contains the data that corresponds to this column. The script wrapper evaluates the XPath expression after the CREATE NICKNAME statement applies this XPath expression from this XPATH nickname option. If you run a query on a column name that has an incorrectly configured XPATH tag reference such as incorrect case, your query returns null values in this column for all returned rows.

SQL queries with the script wrapper

SQL queries that are made through the script wrapper use the custom function to carry the parameter input values for the script.

Any SELECT statement that passes parameter values to a script through the script wrapper must contain at least one predicate with a custom function to take the parameter input values for the script.

Root nicknames

For example, the following statement creates a root nickname for a script named myscript:

```
CREATE NICKNAME customers (
 argle double OPTIONS(SWITCH '-argle', POSITION 1, DEFAULT 1.0),
 argfile CLOB() OPTIONS(SWITCH '-file', INPUT_MODE 'FILE_INPUT', POSITION 2),
 argpos varchar() OPTIONS(SWITCH' ', POSITION 3),
 id  varchar(10) OPTIONS(XPATH'./@id'),
 name varchar OPTIONS(XPATH '/name'))
 FOR SERVER script_server
   OPTIONS(DATASOURCE 'myscript', XPATH 'doc/customer',
       TIMEOUT '300', VALIDATE 'YES');
```

The custom function statements are as follows:

```
CREATE FUNCTION wsscript.args (varchar(), varchar())
 RETURNS INTEGER AS TEMPLATE
 DETERMINISTIC NO EXTERNAL ACTION;

CREATE FUNCTION wsscript.args (date(), date())
 RETURNS INTEGER AS TEMPLATE
 DETERMINISTIC NO EXTERNAL ACTION;

CREATE FUNCTION wsscript.args (integer(), integer())
 RETURNS INTEGER AS TEMPLATE
 DETERMINISTIC NO EXTERNAL ACTION;

CREATE FUNCTION wsscript.args (CLOB(), CLOB())
 RETURNS INTEGER AS TEMPLATE
 DETERMINISTIC NO EXTERNAL ACTION;

CREATE FUNCTION wsscript.args (double(), double)
 RETURNS INTEGER AS TEMPLATE
 DETERMINISTIC NO EXTERNAL ACTION;
```

The configuration file specifies the following configuration parameters:

```
SCRIPT_OUT_DIR_PATH=C:\temp
myscript=C:\perl\bin\perl myscript.pl -model
```

To run a query and send the contents of the table t1.bigdata to the daemon and to the local file C:\temp\f12345, issue the following query and wsscript.args command:

```
SELECT id, name FROM customers, t1
 WHERE wsscript.args (customers.argfile, t1.bigdata) = 1
```

The preceding query results in the following command line:

```
C:\perl\bin\perl myscript.pl -model -argle 1.0 -file C:\temp\f12345
```

To run a query that uses the default values for all parameters, run a query on the nickname with no predicates. To avoid problems with excessive output, include the STREAMING option in the nickname.

Child nicknames

The script wrapper maps the XML result set from the script into nicknames that have a parent-child relationship. To retrieve data from a child nickname, join the child nickname to its parent nickname up to the root. The SELECT statements that reference a child nickname must be joined with the parent nickname of the child nickname by using primary and foreign key columns.

The following query displays the customer names and amounts for each order of each customer:

```
SELECT c.name, o.amount FROM customers c, orders o
 WHERE c.cid=o.cid
 AND wsscript.args (customers.argfile, t1.bigdata) = 1
 AND wsscript.args (customers.argpos, VARCHAR('test1')) = 1
 AND wsscript.args (customers.argle, FLOAT('3.5')) = 1
```

Specify the join c.cid=o.cid statement to indicate the parent-child relationship between the customers nickname and the orders nickname. If you join a child nickname to itself, an error message is returned.

Optimizing script wrapper performance

The location of the script daemon can affect query performance.

To improve network communication performance, use a separate script server for the script daemon. Place the federated server and the script server on separate servers. Also, place the script daemon on the script server.

Chapter 17. Sybase data sources

Configuring access to Sybase data sources

To configure a federated server to access Sybase data sources, you must provide the federated server with information about the data sources and objects that you want to access.

Before you begin

- The Sybase Client SDK software must be installed on a server that will act as the federated server. When you install the Sybase client on Windows, you must specify the **Full** or **Custom** option. If you specify the custom option, you must the **XA Interface Library for ASE Distributed Transaction Manager** option.
- IBM WebSphere Federation Server must be installed on a server that acts as the federated server.
- Check the setup of the federated server.
- Check the federated parameter to ensure that federation is enabled.

About this task

You can configure a federated server to access data that is stored in Sybase data sources by using the DB2 Control Center or by issuing SQL statements on the DB2 command line. The DB2 Control Center includes a wizard to guide you through the steps that are necessary to configure the required federated objects.

Procedure

To configure access to Sybase data sources:

1. Set the Sybase environment variables.
2. Set up and test the Sybase client configuration file using one of the following methods depending on your operating system:
   - Set up and test the Sybase client configuration file (Linux, UNIX).
   - Set up and test the client configuration file (Windows).
3. Register the wrapper.
4. Register the server definition.
5. Create the user mappings.
6. Test the connection to the Sybase server.
7. Register nicknames for Sybase tables and views.

Setting the Sybase environment variables

The Sybase environment variables must be set in the db2dj.ini file on the federated server.

Restrictions

Review the restrictions for the db2dj.ini file.

About this task

The db2dj.ini file contains configuration information about the Sybase Open Client software that is installed on your federated server.

There are required and optional environment variables for Sybase data sources.

If you installed the Sybase Open Client software before you installed the Sybase wrapper, the required Sybase environment variables are set in the db2dj.ini file.

You must set the environment variables by using the steps in this task if you did not install the Sybase Open Client software before you installed the Sybase wrapper or you want to set any of the optional environment variables.

Procedure

To set the Sybase environment variables:

1. Use one of the following methods:

Method	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.
Automatically set the environment variables.	Run the WebSphere Federation Server installation wizard. Follow the instructions in the wizard. **Important:** Set the required environment variables by running the installation wizard. The optional environment variables must be set manually.

Manually set the environment variables.	Edit the db2dj.ini file:
	The db2dj.ini file is located in the directory that the DB2 registry variable DB2_DJ_INI specifies. When the DB2_DJ_INI variable is not set, the db2dj.ini file is in one of the following default paths depending on the operating system:
	• On Linux and UNIX: *instancehome*/sqllib/cfg/db2dj.ini.
	instancehome The home directory of the instance owner.
	• On Windows: %DB2PATH%\cfg\db2dj.ini
	%DB2PATH% The directory where the DB2 database system is installed, for example, C:\Program Files\IBM\sqllib.
	If the file does not exist, you can create a file with the name db2dj.ini by using any text editor. In the db2dj.ini file, you must specify the fully qualified path in the value of the SYBASE environment variable; otherwise you will encounter errors. For example:
	`SYBASE=/opt/sybase`
	`SYBASE_OCS=OCS-12_5`

2. On Linux and UNIX, update the .profile file that is on the federated database instance. Specify information about the Sybase environment variables that you added to the db2dj.ini file. On Windows, the file is updated when you install the Sybase Open Client software.

   Issue the following commands to update the .profile file:

```
export SYBASE=sybase_home_directory
export SYBASE_OCS=OCS-version_release
export PATH=$SYBASE/bin:$PATH
```

3. From the home directory, run the .profile file that is on federated database instance.

   Issue the following command to run the .profile file:

   ```
   . .profile
   ```
4. To ensure that the environment variables are set on the federated server, recycle the federated database instance with these commands:

   ```
   db2stop
   db2start
   ```

After you complete this task, you can register the wrapper.

Sybase environment variables

There are required and optional environment variables for Sybase data sources. These variables are set in the db2dj.ini file.

The following environment variables are valid for Sybase:

- SYBASE
- SYBASE_OCS
- SYBASE_CHARSET (optional)

Variable descriptions

SYBASE

Specifies the directory path where the Sybase Open Client software is installed. Specify the fully qualified path for this environment variable.

For example, if Sybase Open Client Version 12.5 is installed in the directory path D:\djxclient\sybase\V125, specify the following SYBASE environment variable:

```
SYBASE=D:\djxclient\sybase\V125
```

If Sybase Open Client Version 12.0 is installed in the directory path D:\djxclient\sybase\V12, specify the following SYBASE environment variable:

```
SYBASE=D:\djxclient\sybase\V12
```

SYBASE_OCS

Specifies the directory, version and release of the Sybase Open Client software that is installed. Do not specify the fully qualified path when you specify this environment variable.

```
SYBASE_OCS=OCS-version_release
```

For example, if Sybase Open Client Version 12.0 is installed in the directory path D:\djxclient\sybase\V12\OCS-12_0, specify the following value for the SYBASE_OCS environment variable:

```
 SYBASE_OCS=OCS-12_0
```

If Sybase Open Client Version 12.5 is installed in the directory path D:\djxclient\sybase\V125\OCS-12_5, specify the following value for the SYBASE_OCS environment variable:

```
 SYBASE_OCS=OCS-12_5
```

SYBASE_CHARSET

Specifies the name of the character set that you want to use. The Sybase wrapper uses the SYBASE_CHARSET environment variable to determine which character set to use. You can see a list of valid character set names in the `$SYBASE\charsets` directory.

The following example shows how to set the SYBASE_CHARSET when your character set is UTF8:

```
SYBASE_CHARSET=utf8
```

If you do not set the SYBASE_CHARSET environment variable, the wrapper uses the Sybase character set that matches the one that is specified on the code page of the federated database. If there is no matching Sybase character set, the wrapper uses the iso_1 character set.

Setting up and testing the Sybase client configuration file (Windows)

The Sybase client configuration file is used to connect to Sybase databases by using the client libraries that are installed on the federated server.

Before you begin

The Sybase Client SDK software must be installed on the federated server.

About this task

The client configuration file specifies the location of each Sybase SQL Server and Adaptive Server Enterprise instance and the type of connection (protocol) for the database server.

You must set up a client configuration file on each instance in the federated server that will be used to connect to Sybase.

Procedure

To set up and test the Sybase client configuration file on federated servers that run Windows:

1. Set up the client configuration file by using the utility that comes with the Sybase Open Client software. See the Sybase documentation for more information about using this utility.

   The client configuration file is created in the %SYBASE%\ini directory. The name of the file is sql.ini.
2. Test the connection to ensure that the Sybase Open Client software is able to connect to the Sybase server.

   Use an appropriate Sybase query utility, such as isql, to test the connection.

   For example, if the Sybase Open Client software is installed in the directory path, D:\djxclient\sybase\V125, you can issue the following commands from a command prompt:

   ```
   cd D:\djxclient\sybase\V125\OCS-12_5\bin
   isql -Ssybnode -Umary
   ```

   Alternatively, you can issue the following command from a command prompt:

   ```
   %SYBASE%\%SYBASE_OCS%\bin\isql -Ssybnode -Umary
   ```

   Specifying the path to the interfaces file

: If you would like to use an interfaces file other than the default file, use the IFILE server option to specify the path. The Sybase wrapper searches for the interfaces file in the following places, in the order specified:

a. IFILE server option
b. %DB2PATH%\interfaces
c. %SYBASE%\ini\sql.ini

After you complete this task, you can set the environment variables.

Setting up and testing the Sybase client configuration file (UNIX)

The Sybase client configuration file is used to connect to Sybase databases, by using the client libraries that are installed on the federated server.

Before you begin

The Sybase Client SDK software must be installed on the federated server.

About this task

The client configuration file specifies the location of each Sybase SQL Server and Adaptive Server Enterprise instance and the type of connection (protocol) for the database server.

You must set up a client configuration file on each instance in the federated server that will be used to connect to Sybase.

Procedure

To set up and test the Sybase client configuration file on federated servers that run UNIX:

1. Set up the client configuration file by using the utility that comes with the Sybase Open Client software.

   The client configuration file is created in the $SYBASE directory. The default name of the file is interfaces. See the Sybase documentation for more information about using this utility.
2. Test the connection to ensure that the Sybase Open Client software is able to connect to the Sybase server.

   Use an appropriate Sybase query utility, such as isql, to test the connection.

   For example, if the Sybase Open Client software is installed in the directory path, /opt/djxclient/sybase/V125, you can issue the following command from a UNIX prompt:

   ```
   cd /opt/djxclient/sybase/V125/OCS-12_5
   isql -Ssybnode -Umary
   ```

   Alternatively, you can issue the following command from a UNIX prompt:

   ```
   $SYBASE/$SYBASE_OCS/bin/isql -Ssybnode -Umary
   ```

   Specifying the path to the interfaces file

   : If you would like to use an interfaces file other than the default file, use the IFILE server option to specify the path. The Sybase wrapper searches for the interfaces file in the following places, in the order specified:

   a. IFILE server option
   b. sqllib/interfaces

c. $SYBASE/interfaces

After you complete this task, you can set the environment variables.

Registering the Sybase wrapper

You must register a wrapper to access Sybase data sources. Federated servers use wrappers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.

Procedure

To register the Sybase wrapper:

Use one of the following methods:

Method	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.

Issue the CREATE WRAPPER statement and specify the default name for the Sybase wrapper.	For example:
	`CREATE WRAPPER CTLIB`
	Remember: When you register the wrapper by using the default name, CTLIB, the federated server automatically uses the appropriate Sybase wrapper library for the operating system that your federated server is running on.
	If the default wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name that you choose. When you use a name that is different from the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement.
	For example, to register a wrapper with the name sybase_wrapper on a federated server that uses AIX, issue the following statement:
	`CREATE WRAPPER sybase_wrapper LIBRARY 'libdb2ctlib.a';`
	The wrapper library file that you specify depends on the operating system of the federated server.

After you complete this task, you can register the server definition.

Sybase wrapper library files

The Sybase wrapper library files are added to the federated server when you install the wrapper.

When you install the Sybase wrapper, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2ctlib.a, libdb2ctlibF.a, and libdb2ctlibU.a. The other wrapper library files are used internally by the Sybase wrapper.

If you do not use the default wrapper name when you register the wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default library file name.

The default directory paths and default wrapper library file names are listed in the following table.

Table 81. Sybase wrapper library locations and file names

Operating system	Directory path	Library file name
AIX	/usr/opt/*install_path*/lib32/ /usr/opt/*install_path*/lib64/	libdb2ctlib.a
Linux	/opt/IBM/db2/*install_path*/lib32 /opt/IBM/db2/*install_path*/lib64	libdb2ctlib.so
Solaris	/opt/IBM/db2/*install_path*/lib32 /opt/IBM/db2/*install_path*/lib64	libdb2ctlib.so
Windows	%DB2PATH%\bin	db2ctlib.dll

install_path is the directory path where IBM WebSphere Federation Server is installed on UNIX or Linux.

Registering the server definitions for a Sybase data source

You must register each Sybase server that you want to access in the federated database.

Procedure

To register a server definition for a Sybase data source:

1. Locate the node name in the Sybase interfaces file. The following examples show the entries for interfaces files for federated servers that run UNIX or Windows:

   UNIX:
   ```
   sybase125
   query tcp ether anaconda 4100
   ```

   Windows:
   ```
   [sybase125]
   query=TCP,anaconda,4100
   ```

   - The first line in each example is the node name, such as `sybase125`.
   - The second line lists the type of connection, the host name, and the port number. In this example `TCP` indicates that this is a TCP/IP connection, `anaconda` is the host name, and `4100` is the port number. .

2. To create the server definition, use one of the following methods:

Method	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.
Issue the CREATE SERVER statement.	For example: `CREATE SERVER server_definition_name TYPE SYBASE VERSION Sybase_client_version_number WRAPPER wrapper_name OPTIONS (NODE 'node_name', DBNAME 'database_name');` Although the *'node_name'* and *'database_name'* variables are specified as options in the CREATE SERVER statement, these options are required for Sybase data sources. **Important:** If you did not rename the sql.ini file to interfaces when you set up the Sybase client configuration file, you must include the IFILE server option when you register the server definition. After the server definition is registered, use the ALTER SERVER statement to add or drop server options.

After you complete this task, you can create user mappings.

Creating the user mappings for a Sybase data source

When you attempt to access a Sybase server, the federated server establishes a connection to the Sybase server by using a user ID and password that are valid for that data source. You must define an association (a user mapping) between each federated server user ID and password and the corresponding data source user ID and password.

About this task

Create a user mapping for each user ID that will access the federated system to send distributed requests to the Sybase data source.

Procedure

Create user mappings for a Sybase data source:

Issue a CREATE USER MAPPING statement.

For example:

```
CREATE USER MAPPING FOR local_userID SERVER server_definition_name
      OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password');
```

Although the REMOTE_AUTHID and REMOTE_PASSWORD variables are specified as options in the CREATE USER MAPPING statement, these options are required to access Sybase data sources.

After you complete this task, test the connection to the Sybase server.

Testing the connection to the Sybase server

Test the connection to the Sybase data source server to determine if the federated server is properly configured to access Sybase data sources.

About this task

You can test the connection to the Sybase server by using the server definition and user mappings that you defined.

Procedure

To test the connection to the Sybase server:

Open a pass-through session and issue a SELECT statement on the Sybase system tables. If the SELECT statement returns a count, your server definition and your user mapping are set up properly.

For example:

```
SET PASSTHRU server_definition_name
SELECT count(*) FROM dbo.sysobjects
SET PASSTHRU RESET
```

If the SELECT statement returns an error, you should troubleshoot the connection errors.

After you complete this task, you can register nicknames for Sybase tables and views.

Registering nicknames for Sybase tables and views

For each Sybase server definition that you register, you must register a nickname for each table or view that you want to access. Use these nicknames, instead of the names of the data source objects, when you query the Sybase servers.

Before you begin

Update the statistics at the Sybase data source before you register a nickname. The federated database relies on the data source catalog statistics to optimize query processing. Use the data source command that is equivalent to the DB2 RUNSTATS command to update the data source statistics.

Procedure

To register a nickname for Sybase tables or views, use one of the following methods:

Methods	Descriptions
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.

Issue the CREATE NICKNAME statement. Nicknames can be up to 128 characters in length.	For example:
	`CREATE NICKNAME nickname`
	`FOR server_definition_name."remote_schema"."remote.table" ;`

When you create the nickname, the federated server queries the data source catalog by using the nickname. This query tests the connection to the data source table or view. If the connection does not work, you receive an error message.

Repeat this step for each Sybase table or view that you want to create a nickname for.

Troubleshooting the Sybase wrapper configuration

Problems loading the Sybase wrapper library

When you create the Sybase wrapper, you might encounter errors related to the installation of the Sybase Open Client software that prevents the Sybase wrapper library from being loaded.

Symptom

When you create the Sybase wrapper, the following SQL error is issued:

```
SQL10013N The specified library "db2ctlibF.dll"
could not be loaded.
```

Cause

The Sybase XA Interface Library for ASE Distributed Transaction Manager was not installed on the Windows system with the Sybase Open Client software.

Resolving the problem

Reinstall the Sybase Open Client software on Windows and select the **Full** or **Custom** installation option. If you select the **Custom** installation option, specify the **XA Interface Library for ASE Distributed Transaction Manager** option.

Missing SYBASE environment variable

If the db2dj.ini file is not in the correct directory or if it is missing, you will encounter an SQL error.

Symptom

The following SQL error is issued:

```
SQL1822N Unexpected error code "" received from data source
"server name". Associated text and tokens are "SYBASE variable not set"."
```

Cause

The db2dj.ini file was not found or the file does not contain the SYBASE environment variable. The db2dj.ini file is located in the directory that the DB2 registry variable DB2_DJ_INI specifies. When the DB2_DJ_INI variable is not set, the db2dj.ini file is in one of the following default paths depending on the operating system:

- On UNIX: *instancehome*/sqllib/cfg/db2dj.ini.

  instancehome

  The home directory of the instance owner.

- On Windows: %DB2PATH%\cfg\db2dj.ini

  %DB2PATH%
  The directory where the DB2 database system is installed, for example, C:\Program Files\IBM\sqllib.

Resolving the problem

Create a db2dj.ini file with the required Sybase environment variables and place it in the correct directory for your operating system. You can use a text editor to create the db2dj.ini file.

Missing Sybase node name

If the Sybase client configuration file is not properly configured, the federated system might be unable to locate the Sybase node name.

Symptom

The following SQL error is issued:

```
SQL1097N The node name was not found in the node directory.
```

Cause

The Sybase node name was not found.

Resolving the problem

To resolve the problem:

- If the IFILE server option was specified, verify that the node name is declared in the file that it specifies.
- If the interfaces file exists in sqllib directory, verify that the node name is declared in it.
- If the IFILE server option is not specified and the interfaces file does not exist in the sqllib directory, verify that the node name is declared in the %SYBASE%\ini\sql.ini file on Windows or the $SYBASE/interfaces file on UNIX.

CREATE SERVER statement - Examples for the Sybase wrapper

Use the CREATE SERVER statement to register server definitions for the Sybase wrapper. This topic provides a complete example with the required parameters, and an example with additional server options.

The following example shows you how to register a server definition for a Sybase wrapper by issuing the CREATE SERVER statement:

```
CREATE SERVER SYBSERVER TYPE SYBASE VERSION 12.0 WRAPPER CTLIB
      OPTIONS (NODE 'sybnode', DBNAME 'sybdb');
```

SYBSERVER
A name that you assign to the Sybase server. Duplicate server definition names are not allowed.

TYPE *SYBASE*
Specifies the type of data source server to which you are configuring access. For the CTLIB wrapper, the server type must be *SYBASE*.

VERSION *12.0*

The version of the Sybase database client software that is being used for the federated connection.

WRAPPER *CTLIB*

The wrapper name that you specified in the CREATE WRAPPER statement.

NODE *'sybnode'*

The name of the node where the Sybase server resides. Obtain the node name from the interfaces file. This value is case sensitive.

Although the node name is specified as an option in the CREATE SERVER statement, it is required for Sybase data sources.

DBNAME *'sybdb'*

The name of the Sybase database that you want to access. This value is case sensitive.

Although the name of the database is specified as an option in the CREATE SERVER statement, it is required for Sybase data sources.

Server options

When you create a server definition, you can specify additional server options in the CREATE SERVER statement. The server options can be general server options and Sybase-specific server options.

IFILE server option

If you are not using the default interfaces file, you must create an interfaces file and include the IFILE server option in the CREATE SERVER statement. The default interfaces file is $SYBASE/interfaces for Linux and UNIX, and %SYBASE%\ini\sql.ini for Windows.

The value that you specify for the IFILE server option is the full path and name of the Sybase Open Client sql.ini file.

The following example shows how to use the IFILE server option when you register a server definition on a federated server that runs Windows:

```
CREATE SERVER SYBSERVER TYPE SYBASE VERSION 12.0 WRAPPER CTLIB
       OPTIONS (NODE 'sybnode', DBNAME 'sybdb',
       IFILE 'C:\Sybase\ini\sql.ini');
```

TIMEOUT server option

The TIMEOUT server option sets the number of seconds that the wrapper waits for a response from the Sybase server. Use the TIMEOUT option to avoid deadlocks on transactions.

The following example shows how to specify the TIMEOUT server option when you register a server definition:

```
CREATE SERVER SYBSERVER TYPE SYBASE VERSION 12.0 WRAPPER CTLIB
       OPTIONS (NODE 'sybnode', DBNAME 'sybdb',
         TIMEOUT '60');
```

The additional Sybase-specific server options are:

- LOGIN_TIMEOUT

- PACKET_SIZE

CREATE USER MAPPING statement - Examples for the Sybase wrapper

Use the CREATE USER MAPPING statement to map a federated server user ID to a Sybase server user ID and password. This topic provides a complete example with the required parameters, and an example that shows you how to use the DB2 special register USER with the CREATE USER MAPPING statement.

The following example shows how to map a federated server user ID to a Sybase server user ID and password:

```
CREATE USER MAPPING FOR maria SERVER SYBSERVER
      OPTIONS (REMOTE_AUTHID 'mary', REMOTE_PASSWORD 'day2night');
```

maria Specifies the local user ID that you are mapping to a user ID that is defined at the Sybase server.

SERVER *SYBSERVER*
Specifies the server definition name that you registered in the CREATE SERVER statement for the Sybase server.

REMOTE_AUTHID *'mary'*
Specifies the user ID at the Sybase server to which you are mapping *maria*. Use single quotation marks to preserve the case of this value unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

Although the remote user ID is specified as an option in the CREATE SERVER statement, it is required for Sybase data sources.

REMOTE_PASSWORD *'day2night'*
Specifies the password that is associated with *'mary'*. Use single quotation marks to preserve the case of this value unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

Although the remote password is specified as an option in the CREATE SERVER statement, it is required for Sybase data sources.

DB2 special register USER

You can use the DB2 special register USER to map the authorization ID of the person who is issuing the CREATE USER MAPPING statement to the data source authorization ID that is specified in the REMOTE_AUTHID user option.

The following example shows a CREATE USER MAPPING statement that includes the special register USER:

```
CREATE USER MAPPING FOR USER SERVER SYBSERVER
      OPTIONS (REMOTE_AUTHID 'mary', REMOTE_PASSWORD 'day2night');
```

CREATE NICKNAME statement - Examples for the Sybase wrapper

Use the CREATE NICKNAME statement to register a nickname for a Sybase table or view that you want to access. This topic includes a complete example with the required parameters.

The following example shows you how to register a nickname for a Sybase table or view using the CREATE NICKNAME statement.

```
CREATE NICKNAME SYBSALES FOR SYBSERVER."vinnie"."europe";
```

SYBSALES

A unique nickname that is used to identify the Sybase table or view.

Important: The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authorization ID of the user who is registering the nickname.

SYBSERVER."vinnie"."europe"

A three-part identifier for the remote object:

- *SYBSERVER* is the server definition name that you assigned to the Sybase database server in the CREATE SERVER statement.
- *vinnie* is the user ID of the owner to which the table or view belongs.
- *europe* is the name of the remote table or view that you want to access.

The federated server folds the names of the Sybase schemas and tables to uppercase unless you enclose the names in quotation marks.

Chapter 18. Table-structured file data sources

Configuring access to table-structured file data sources

You can integrate the data that is in table-structured file with information from other sources by using a federated system.

Procedure

To configure a federated server to access table-structured file data sources, you must provide the federated server with information about the data sources and objects that you want to access. After you configure the federated server, you can create queries and use the custom functions to access the table-structured file data sources.

Table-structured files - overview

A table-structured file is a file that has a regular structure that consists of a series of records or rows of information. Each record contains the same number of fields. The data in the fields are separated by a delimiter, such as a comma.

The following example shows the contents of a file called DRUGDATA1.TXT. It contains three records, each with three fields, separated by commas:

```
234,DrugnameA,Manufacturer1
332,DrugnameB,Manufacturer2
333,DrugnameC,Manufacturer2
```

The first field is the unique ID number for the drug. The second field is the name of the drug. The third field is the name of the manufacturer who produces the drug.

The field delimiter can be more than one character in length. A single quotation mark cannot be used as a delimiter. The delimiter must be consistent throughout the file. A null value is represented by two delimiters next to each other or a delimiter followed by a line terminator, if the NULL field is the last one on the line. The column delimiter cannot exist as valid data for a column.

For example:

```
234,DrugnameA,Manufacturer1
332,DrugnameB,Manufacturer2
333,DrugnameC,Manufacturer2
356,,Manufacturer1
```

Attributes of table-structured files

The records, or rows, in table-structured files can be sorted or unsorted. The table-structured files wrapper can search files that are sorted more efficiently than files that are not sorted.

If the data in a table-structured file is sorted, the sort must be in ascending order on the key column. You should set the SORTED option to Y on the key column when you create define the columns for the nickname. Otherwise the wrapper processes the data in the table-structured file as unsorted data.

Sorted files

DRUGDATA1.TXT contains sorted records. The file is sorted by the first field, the unique ID number for the drug. This field is the primary key because it is unique for each drug. Sorted files must be sorted in ascending order.

```
234,DrugnameA,Manufacturer1
332,DrugnameB,Manufacturer2
333,DrugnameC,Manufacturer2
```

Unsorted files

DRUGDATA2.TXT contains unsorted records. There is no order to the way the records are listed in the file.

```
556,DrugnameB,Manufacturer2
234,DrugnameA,Manufacturer1
721,DrugnameC,Manufacturer2
```

Table-structured files wrapper

Data in a table-structured file can be joined with data in other table-structured files, relational data, or nonrelational and unstructured data.

Using a wrapper, the federated server can process SQL statements that query data in a table-structured file as if the data is contained in an ordinary relational table or view.

How the federated server works with table-structured files is illustrated in the following figure.

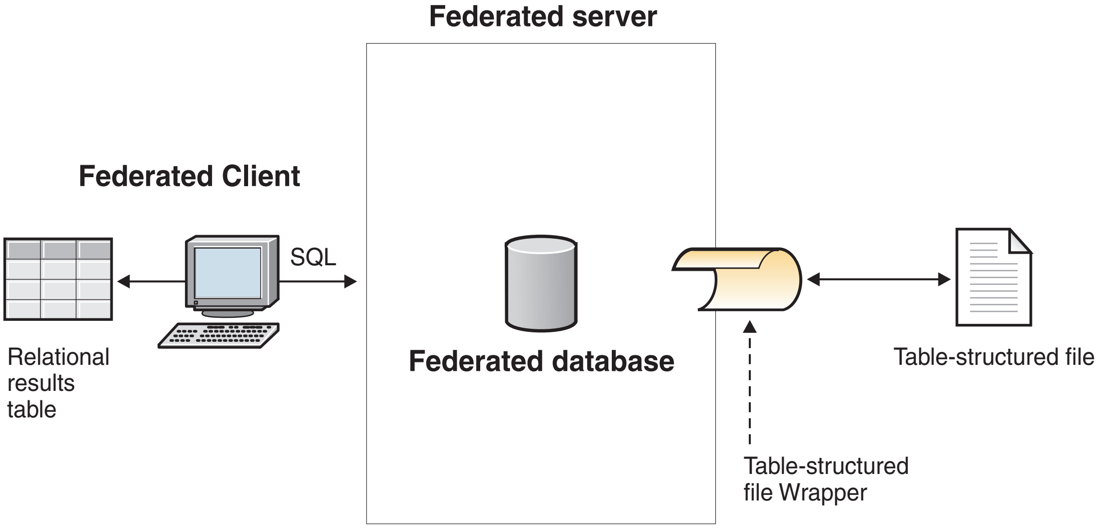


Figure 22. How the table–structured file wrapper works

For example, the table-structured file `DRUGDATA2.TXT` is located on a computer in your laboratory. The data in the file is:

```
556,DrugnameB,Manufacturer1
234,DrugnameA,Manufacturer2
721,DrugnameC,Manufacturer2
```

It is tedious to try to query and match this data with other tables from other data sources that you use.

After you register the DRUGDATA2.TXT file with the federated server, the table-structured file wrapper can access the data in the file as if the data is in a relational table.

For example, you can run the following query:

```
SELECT * FROM DRUGDATA2 ORDER BY DCODE
```

This query produces the following results:

DCODE	DRUG	MANUFACTURER
234	DrugnameA	Manufacturer2
556	DrugnameB	Manufacturer1
721	DrugnameC	Manufacturer2

You can join the data in the DRUGDATA2.TXT file with data from other relational and nonrelational data sources and analyze all of the data together.

Adding table-structured file data sources to a federated server

To configure a federated server to access table-structured file data sources, you must provide the federated server with information about the data sources and objects that you want to access.

Before you begin

- IBM WebSphere Federation Server must be installed on a server that will act as the federated server
- A federated database must exist on the federated server

About this task

You can configure a federated server to access data that is stored in table-structured file data sources by using the Control Center or by issuing SQL statements on the command line. The Control Center includes a wizard to guide you through the steps that are necessary to configure the required federated objects.

Procedure

To add the table-structured file data sources to a federated server:

1. Register the table-structured file wrapper.
2. Register the table-structured file server definition.
3. Register nicknames for the table-structured files.

Registering the table-structured file wrapper

You must register a wrapper to access the table-structured files data sources. Wrappers are used by federated servers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.

About this task

You can register a wrapper by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps that are necessary to register the wrapper.

Procedure

To register the table-structured file wrapper:

Choose the method that you want to use to register the table-structured file wrapper:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.

From the command line	Issue the CREATE WRAPPER statement.
	`CREATE WRAPPER` *`wrapper_name`*
	`LIBRARY` *`library_name`*`;`
	For example to register a wrapper with the name flat_files_wrapper on the federated server that uses the AIX operating system, issue the following statement:
	`CREATE WRAPPER` *`flat_files_wrapper`*
	`LIBRARY` *`'libdb2lsfile.a'`*`;`

You must specify the LIBRARY parameter in the CREATE WRAPPER statement. The name of the wrapper library file that you specify depends on the operating system of the federated server. See the list of table-structured file wrapper library files for the correct library name to specify in the CREATE WRAPPER statement.

Table-structured files wrapper library files

The table-structured files wrapper library files are added to the federated server when you install IBM WebSphere Federation Server.

When you install IBM WebSphere Federation Server, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2lsfile.a, libdb2lsfileF.a, and libdb2lsfileU.a. The default wrapper library file is libdb2lsfile.a. The other wrapper library files are used with specific wrapper options.

You must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

The default directory paths and default wrapper library file names are listed in the following table.

Table 82. Table-structured files client library locations and file names

Operating system	Directory path	Wrapper library file names
AIX	/usr/opt/`<install_path>`/lib32/ /usr/opt/`<install_path>`/lib64/	libdb2lsfile.a
Linux	/opt/IBM/db2/`<install_path>`/lib32 /opt/IBM/db2/`<install_path>`/lib64	libdb2lsfile.so

Table 82. Table-structured files client library locations and file names (continued)

Operating system	Directory path	Wrapper library file names
Solaris	/opt/IBM/db2/<install_path>/lib32 /opt/IBM/db2/<install_path>/lib64	libdb2lsfile.so
Windows	%DB2PATH%\bin	db2lsfile.dll

`<install_path>` is the directory path where IBM WebSphere Federation Server is installed on UNIX or Linux.

%DB2PATH% is the environment variable that is used to specify the directory path where IBM WebSphere Federation Server is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

Registering the server definition for table-structured files

For table-structured files, you must register a server definition because the hierarchy of federated objects requires that the table-structured files, which are identified by nicknames, are associated with a specific server definition object.

About this task

You can register a server definition by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps that are necessary to register the server definition.

Procedure

To register a server definition for a table-structured file data source:

Choose the method that you want to use to register the server definition:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.
From the command line	Issue the CREATE SERVER statement. For example: `CREATE SERVER server_definition_name WRAPPER wrapper_name;`

CREATE SERVER statement - example for the table-structured file wrapper

Use the CREATE SERVER statement to register the server definition for the table-structured file wrapper. This example shows the required parameters.

The following example shows you how to register a server definition called `biochem_lab` for a text file that contains biochemical data. The CREATE SERVER statement that you issue is:

```
CREATE SERVER biochem_lab WRAPPER flat_files_wrapper;
```

biochem_lab
A name that you assign to the table-structured file server definition. Duplicate server definition names are not allowed.

WRAPPER *flat_files_wrapper*
The wrapper name that you specified in the CREATE WRAPPER statement.

Registering nicknames for table-structured files

You must register a nickname for each table-structured file that you want to access. Use these nicknames, instead of the names of the files, when you query table-structured file data sources.

Restrictions

If a nonnumeric field is too long for its column type, the excess data is truncated.

If a decimal field in the file has more digits after the radix character than are allowed, the excess data is truncated. For example, if the number is 10.123456 and the data type specifies that 3 digits are allowed after the radix character, the number is truncated to 10.123. The radix character is determined by the RADIXCHAR item of the LC_NUMERIC National Language Support category. The scale parameter for the column data type specifies the number of digits allowed after the radix character.

If you attempt to access table-structured file data sources that are on a shared drive from federated server that runs Windows 2003, your queries might fail. This is a limitation of Windows 2003. You can avoid this problem by specifying the absolute path in the FILE_PATH option in the CREATE NICKNAME statement.

The maximum length for a line of data is 10 megabytes (10485760 bytes).

About this task

When you create a nickname for a table-structured file, the information in the data in the file is mapped to a relational table.

You create nicknames for your table-structured file in one of two ways:

- Specifying the table-structured file when you create the nickname by using the FILE_PATH nickname option.
- Specifying the table-structured file when you query the data source, by using the DOCUMENT nickname column option. When this option is used, the nickname can be used to represent data from any table-structured file whose schema matches the nickname definition.

The names that you give the nicknames can be up to 128 characters in length.

You can register a nickname by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps that are necessary to register the nickname.

Procedure

To register a nickname for a table-structured file:

Choose the method that you want to use to register the nickname:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.
Using the command line	Issue the CREATE NICKNAME statement. For example: `CREATE NICKNAME nickname ( column_name data_type, column_name data_type, column_name data_type ) FOR SERVER server_definition_name OPTIONS (nickname_options);`

Repeat this step for each table-structured file that you want to create a nickname for.

CREATE NICKNAME statement - examples for table-structured file wrapper

Use the CREATE NICKNAME statement to register a nickname for a table-structured file that you want to access.

You must specify either the FILE_PATH nickname option or the DOCUMENT nickname column option when you register a nickname for a table-structured file.

Creating a nickname with the FILE_PATH nickname option

The following example shows a CREATE NICKNAME statement for the table-structured file DRUGDATA1.TXT:

```
CREATE NICKNAME DRUGDATA1
    (
    Dcode INTEGER NOT NULL,
    Drug CHAR(20),
    Manufacturer CHAR(20)
    )
    FOR SERVER biochem_lab
    OPTIONS(FILE_PATH '/usr/pat/DRUGDATA1.TXT')
```

DRUGDATA1
: A unique nickname that is used to identify the table-structured file.

  The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname is the authorization ID of the user who registers the nickname.

Dcode *INTEGER NOT NULL*
: The name and data type for a table-structured file column that contains a drug code.

Drug *CHAR(20)*
: The name and data type for a table-structured file column that contains the name of a drug.

Manufacturer *CHAR(20)*
: The name and data type for a table-structured file column that contains the name of the drug manufacturer.

FOR SERVER *biochem_lab*
: The name that you assigned to the table-structured file server definition in the CREATE SERVER statement.

FILE_PATH *'/usr/pat/DRUGDATA1.TXT'*
: Specifies the fully qualified directory path and file name for the table-structured file that contains the data you want to access. The path must be enclosed in single quotation marks.

Creating a nickname with the DOCUMENT nickname column option

When you create a nickname using the DOCUMENT nickname column option, you are specifying that the name of table-structured file will be supplied when you run a query that uses the nickname. You can specify the DOCUMENT nickname column option on only one column when you register the nickname. The column that is associated with the DOCUMENT option must be either a VARCHAR or CHAR data type. You must include the full path of the file when you run a query that uses the nickname.

The following example shows a CREATE NICKNAME statement that specifies the DOCUMENT nickname column option:

```
CREATE NICKNAME customers
   (
   doc  VARCHAR(100) OPTIONS(DOCUMENT 'FILE'),
   name VARCHAR(16),
   address VARCHAR(30),
   id VARCHAR(16))
   FOR SERVER biochem_lab
```

customers
: A unique name for the nickname.

  The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname is the authorization ID of the user who registers the nickname.

doc *VARCHAR(100)* **OPTIONS(DOCUMENT 'FILE')**
: The name and data type for a column that is used to specify the name of the table-structured file that you want to access. You specify the file name when you run the query.

name *VARCHAR(16)*
: The name and data type for a table-structured file column that contains the name of the customer.

address *VARCHAR(30)*
: The name and data type for a table-structured file column that contains the address of the customer.

id *VARCHAR(16)*

FOR SERVER *biochem_lab*
: The name that you assigned to the table-structured file server definition in the CREATE SERVER statement.

FILE_PATH *'/usr/pat/DRUGDATA1.TXT'*
: Specifies the fully qualified directory path and file name for the

table-structured file that contains the data you want to access. You must specify either the FILE_PATH or DOCUMENT nickname option in the CREATE NICKNAME statement. The path must be enclosed in single quotation marks.

Creating a nickname with the optional parameters

The following example shows a CREATE NICKNAME statement for the table-structured file DRUGDATA1.TXT:

```
CREATE NICKNAME DRUGDATA1
    (
    Dcode INTEGER NOT NULL,
    Drug CHAR(20),
    Manufacturer CHAR(20)
    )
    FOR SERVER biochem_lab
    OPTIONS(FILE_PATH '/usr/pat/DRUGDATA1.TXT',
        COLUMN_DELIMITER ',',
        SORTED 'Y',
        KEY_COLUMN 'DCODE',
        VALIDATE_DATA_FILE 'Y')
```

DRUGDATA1
A unique nickname that is used to identify the table-structured file.

The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname is the authorization ID of the user who registers the nickname.

Dcode *INTEGER NOT NULL*
The name and data type for a table-structured file column that contains a drug code.

Drug *CHAR(20)*
The name and data type for a table-structured file column that contains the name of a drug.

Manufacturer *CHAR(20)*
The name and data type for a table-structured file column that contains the name of the drug manufacturer.

FOR SERVER *biochem_lab*
The name that you assigned to the table-structured file server definition in the CREATE SERVER statement.

FILE_PATH *'/usr/pat/DRUGDATA1.TXT'*
Specifies the fully qualified directory path and file name for the table-structured file that contains the data you want to access. You must specify either the FILE_PATH or DOCUMENT nickname option in the CREATE NICKNAME statement. The path must be enclosed in single quotation marks.

COLUMN_DELIMITER *','*
Specifies the delimiter that is used to separate the fields in a table-structured file. The delimiter value must be enclosed in single quotation marks. The column delimiter can be more than one character in length. If you do not specify a column delimiter, the default delimiter is a comma. A single quote cannot be used as a delimiter. The column delimiter must be consistent throughout the file. A null value is represented by two delimiters next to each other or a delimiter followed by a line terminator, if the NULL field is the last one on the line. The column delimiter cannot

exist as valid data for a column. For example, a column delimiter of a comma cannot be used if one of the columns contains data with embedded commas.

SORTED *'Y'*
Specifies that the data source file is sorted. Sorted data sources must be sorted in ascending order according to the collation sequence for the current locale, as defined by the settings in the LC_COLLATE National Language Support category. If you specify that the data source is sorted, set the VALIDATE_DATA_FILE option to 'Y'. The default value for the SORTED parameter is 'N'.

KEY_COLUMN *'DCODE'*
The name of the column in the file that forms the key on which the file is sorted. The key column value must be enclosed in single quotation marks. Specify this option only if you specify the SORTED nickname option. A column that is designated with the DOCUMENT nickname column option must not be specified as the key column. The value must be the name of a column that is defined in the CREATE NICKNAME statement. The key column must not contain null values. Only single column keys are supported. Multiple column keys are not allowed. The column must be sorted in ascending order. If the value is not specified for a sorted nickname, it defaults to the first column in the nicknamed file.

VALIDATE_DATA_FILE *'Y'*
Specifies if the wrapper verifies that the key column is sorted in ascending order and checks for null keys. The valid values for this option are Y or N. This option is not allowed if the DOCUMENT nickname column option is used for the file path.

Designating key columns when you register a nickname

You can designate a key column by specifying the NOT NULL constraint in the nickname statement:

```
CREATE NICKNAME tox (tox_id INTEGER NOT NULL, toxicity VARCHAR(100))
FOR SERVER tox_server1
  OPTIONS (FILE_PATH'/tox_data.txt', SORTED 'Y')

CREATE NICKNAME weights (mol_id INTEGER, wt VARCHAR(100) NOT NULL)
FOR SERVER wt_server
  OPTIONS (FILE_PATH'/wt_data.txt', SORTED 'Y', KEY_COLUMN 'WT')
```

NOT NULL
Specifies that the column cannot contain null, or blank, values.

The wrapper does not enforce the NOT NULL constraint, but the federated database does. If you create a nickname and attach a NOT NULL constraint on a column and then select a row containing a null value for the column, the federated database issues a SQL0407N error stating that you cannot assign a NULL value to a NOT NULL column.

The exception to this rule is for sorted nicknames. The key column for sorted nicknames cannot be NULL. If a NULL key column is found for a sorted nickname, the SQL1822N error is issued, stating that the key column is missing.

Case sensitive column names

The federated database changes the column names to uppercase unless the column is defined with double quotation marks. The following example will not work correctly because the value of the KEY_COLUMN option is enclosed in single quotation marks. In the example, the column name will be converted by the federated database to EMPNO. As a result, when you specify empno in a query, the column is not recognized by the federated database.

```
CREATE NICKNAME depart (
 empno char(6) NOT NULL)
 FOR SERVER DATASTORE
 OPTIONS(FILE_PATH'data.txt', SORTED 'Y', KEY_COLUMN 'empno');
```

Windows 2003 federated servers

If you attempt to access table-structured file data sources that are on a shared drive from a federated server that runs Windows 2003, your query might fail with the following error message:

```
SQL1822N  Unexpected error code "ERRNO = 2" received from data source
"SERVERNAME1". Associated text and tokens are "Unable to read file".
SQLSTATE=560BD
```

This is a limitation of Windows 2003. You can avoid this problem by specifying the absolute path in the FILE_PATH option in the CREATE NICKNAME statement.

The following example shows a CREATE NICKNAME statement with an abbreviated path specified in the FILE_PATH option:

```
CREATE NICKNAME nickname
   (
   COL1 CHAR (10) NOT NULL
   )
   FOR SERVER servername1
   OPTIONS (FILE_PATH 'X:\textfile1.txt');
```

where X:\ is the drive that maps to the remote machine. Queries that use this nickname might fail because you specified the abbreviated path.

For federated server that runs Windows 2003, specify the absolute path in the FILE_PATH option in the CREATE NICKNAME statement.

For example:

```
CREATE NICKNAME nickname
   (
   COL1 CHAR (10) NOT NULL
   )
   FOR SERVER servername1
   OPTIONS (FILE_PATH '\\host.svl.ibm.com\D$\textfile1.txt') ;
```

File access control model for the table-structured file wrapper

The wrapper accesses table-structured files using the authorization information of the federated database instance owner. The wrapper can only access files that can be read by this user ID or group ID. The authorization ID that establishes the connection to the federated database is not used to access table-structured files.

On a federated server, any table-structured file for which a nickname has been created must be accessible with the same path name from each node. The file does not have to be on a federated database node as long as the file can be accessed from any node with a common path.

To access a table-structured file, the wrapper needs a user identity for security purposes. The table-structured file wrapper uses the user identity that is associated with the federated database service. The name of the federated database service depends on the name of the database instance. For example, if the database instance name is DB2, then the service name is DB2 - DB2. To determine the user identity that is associated with a federated database service, use the Control Panel in Windows to display the services. Double-click the service name and display the Log On properties page.

Table-structured files located on remote drives

The table-structured files that you want to access must be on a local or mapped drive.

Networks that have a Windows NT® or Windows 2000 domain configured
The logon account for the federated database service must be an account from the domain that has access to the shared folder on the mapped drive where the table-structured files reside.

Networks that do not have a Windows NT or Windows 2000 domain configured
The federated database service logon account should have the same user name and password as a valid user on the computer that shares that folder. That user must be on the permissions list for the shared folder with at least read access

Guidelines for optimizing query performance for the table-structured file wrapper

You can improve the performance of table structured file queries by having files that are sorted and by creating statistics for your nicknames.

Use the following tips to improve query performance:

- Sort the data in your files. The federated server can search files that are sorted much more efficiently than files that are not sorted.
- For sorted files, specify a value or range for the key column when you submit a query.
- Statistics for nicknames of table-structured files must be updated manually by updating the SYSSTAT and SYSCAT views. Use the Nickname statistics update facility to update the statistics for the table-structured file nicknames.

Chapter 19. Teradata data sources

Configuring access to Teradata data sources

To configure a federated server to access Teradata data sources, you must provide the federated server with information about the data sources and objects that you want to access.

Before you begin

- Teradata client software must be installed and configured on the server that acts as the federated server.
- IBM WebSphere Federation Server must be installed on a server that acts as the federated server.
- Check the setup of the federated server.
- Check the federated parameter to ensure that federation is enabled.

About this task

You can configure a federated server to access data that is stored in Teradata data sources by using the DB2 Control Center or by issuing SQL statements on the DB2 command line. The DB2 Control Center includes a wizard to guide you through the steps that are necessary to configure the required federated objects.

Procedure

To configure access to Teradata data sources:

1. Test the connection to the Teradata server.
2. Verify that the Teradata library is enabled for run-time linking (AIX).
3. Set the environment variables for the Teradata wrapper.
4. Register the wrapper.
5. Register the server definition.
6. Create the user mappings.
7. Test the connection from the federated server to the Teradata server.
8. Register nicknames for Teradata tables and views.

Testing the connection to the Teradata server

Test the connection to the Teradata server to verify that the Teradata client software is properly set up on the federated server.

Before you begin

The Basic Teradata Query (BTEQ) utility and the Teradata Data Connector Application Program Interface (PIOM) must be installed on the federated server. The BTEQ utility and the Teradata Data Connector Application Program Interface are installed on the federated server when you install the Teradata client software.

About this task

You use the BTEQ utility to submit an SQL query to verify that the federated server can connect to the Teradata server. See the Teradata documentation for more information about the BTEQ utility.

Procedure

To test the connection to the Teradata server:

1. Start a BTEQ utility session and log on to the Teradata server.
2. Issue an SQL command to verify that you can successfully connect to the Teradata server.

   For example:

   ```
   select count(*) from dbc.tables;
   ```

   If the connection is successful, you will see the output from the query.

   If the connection is unsuccessful, you will receive an error. Check the Teradata client software to verify that it is properly installed and configured on the federated server.
3. Log off from the Teradata server and end the BTEQ utility session.

After you complete this task, verify that the Teradata library is enabled for run-time linking.

Verifying that the Teradata library is enabled for run-time linking (AIX)

When you add a Teradata data source to your federated server on AIX, you must verify that run-time linking is enabled before you register wrappers or servers.

Procedure

To verify that the Teradata library is enabled for run-time linking:

1. Go to the directory in which the libcliv2.so file resides.

   The libcliv2.so file is installed with the Teradata client software. By default, it is installed in the /usr/lib directory.
2. From a command prompt, issue the following UNIX command to verify if run-time linking is enabled:

   ```
   dump -H libcliv2.so | grep libtli.a
   ```
3. Check the file names that are returned. If the libtli.a file name is returned, the Teradata library is enabled for run-time linking.

   If the libtli.a file name is not returned, open a command window and issue the following UNIX commands to enable run-time linking for the Teradata library:

   ```
   rtl_enable libcliv2.so -F libtli.a
   mv libcliv2.so libcliv2.so.old
   mv libcliv2.so.new libcliv2.so
   chmod a+r libcliv2.so
   ```

After you complete this task, you can set the environment variables.

Setting the Teradata environment variables

The Teradata environment variables must be set in the db2dj.ini file on the federated server.

Restrictions

Review the restrictions for the db2dj.ini file.

About this task

The db2dj.ini file contains configuration information about the Teradata client software that is installed on your federated server.

There are required and optional environment variables for Teradata data sources.

If you installed the Teradata client software before you installed the Teradata wrapper, the required Teradata environment variables are set in the db2dj.ini file.

You must set the environment variables by using the steps in this task if you did not install the Teradata client software before you installed the Teradata wrapper or if you want to set any of the optional environment variables.

Procedure

To set the Teradata environment variables:

1. Use one of the following methods:

Method	Step
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.
Automatically set the environment variables.	Run the WebSphere Federation Server installation wizard. Follow the instructions in the wizard. **Important:** Set the required environment variables by running the installation wizard. The optional environment variables must be set manually.

Method	Step
Manually set the environment variables.	Edit the db2dj.ini file: The db2dj.ini file is located in the directory that the DB2 registry variable DB2_DJ_INI specifies. When the DB2_DJ_INI variable is not set, the db2dj.ini file is in one of the following default paths depending on the operating system: • On UNIX: *instancehome*/sqllib/cfg/db2dj.ini. *instancehome* The home directory of the instance owner. • On Windows: %DB2PATH%\cfg\db2dj.ini **%DB2PATH%** The directory where the DB2 database system is installed, for example, C:\Program Files\IBM\sqllib. If the file does not exist, you can create a file with the name db2dj.ini by using any text editor. In the db2dj.ini file, you must specify the fully qualified path in the value of the environment variables; otherwise you will encounter errors. For example: `COPLIB=/usr/lib`

2. Set the Teradata code page conversion environment variables (as necessary).
3. To ensure that the environment variables are set on the federated server, recycle the federated database instance with these commands:

```
db2stop
db2start
```

After you complete this task, you can register the wrapper.

Teradata environment variables

There are required and optional environment variables for Teradata data sources. These variables are set in the db2dj.ini file.

The following environment variables are valid for Teradata:

- COPLIB
- COPERR
- TERADATA_CHARSET (optional)
- NETRACE (optional)
- COPANOMLOG (optional)

Variable descriptions

COPLIB

Specifies the directory path on the federated server for the libcliv2.so file. Specify the fully qualified path for the COPLIB variable.

For example:

```
COPLIB=/usr/lib
```

The libcliv2.so and errmsg.cat files typically reside in the same directory.

COPERR

Specifies the directory path on the federated server for the errmsg.cat file. Specify the fully qualified path for the COPERR variable.

For example:

```
COPERR=/usr/lib
```

TERADATA_CHARSET

Specifies the code page character set to use with Teradata data sources.

Each time that the federated server connects to a Teradata data source, the Teradata wrapper determines which code page character set to use for that connection. You can have the Teradata wrapper set the code page character set or you can designate a code page by setting the TERADATA_CHARSET environment variable.

If the TERADARA_CHARSET environment variable is set in the db2dj.ini file on the federated server, the wrapper uses the code page character set in the db2dj.ini file. The value in the TERADATA_CHARSET environment variable is not validated, but if the environment variable is not set to a valid value, the Teradata data source returns an error.

If the TERADARA_CHARSET environment variable is not set in the db2dj.ini file on the federated server, the wrapper detects the client character set based on the code page of the database.

On federated servers that run UNIX, the following values are valid for the TERADATA_CHARSET environment variable:

- HANGULKSC5601_2R4
- KanjiEUC_0U
- LATIN1_0A
- LATIN9_0A
- LATIN1252_0A
- SCHGB2312_1T0
- TCHBIG5_1R0
- UTF8
- ASCII

On federated servers that run Windows, the following values are valid for the TERADATA_CHARSET environment variable:

- HANGULKSC5601_2R4
- KanjiSJIS_0S
- LATIN1_0A
- LATIN1252_0A
- SCHGB2312_1T0
- TCHBIG5_1R0
- UTF8
- ASCII

NETRACE

Optional. Enables the tracing feature of the Teradata client software. This variable is needed only for debugging.

COPANOMLOG

Optional. Enables the logging feature of the Teradata client software. This variable is needed only for debugging.

Verifying the character set on the Teradata server

If the correct character set is not specified on the Teradata server, you can receive connection errors. Verify that the character set that you want to use is installed on the Teradata server.

Procedure

To verify that the character set that you want to use is installed on the Teradata server:

1. Log on to the Teradata server by using the BTEQ utility or any other valid logon utility.
2. Issue the following statement to display the dbc.chartranslations table: `select * from dbc.chartranslations;`
3. Check the value in the third column, InstallFlag, of the table that is returned. The value ′Y′ in the third column indicates that the character set is installed and in use on the Teradata server.

   Use the following table to determine if you have the correct character set installed:

Table 83. Character sets for Teradata

Double-byte character set	Single-byte character set	Teradata character set	Language	IBM DB2 code set
941	897	″KanjiSJIS_0S″	Japanese	IBM-943
1362	1126	″HANGULKSC5601_2R4″	Korean	1363
1385	1114	″SCHGB2312_1T0″	Simplified Chinese	GBk
380	1115	″SCHGB2312_1T0″	Simplified Chinese	IBM-1381
947	1114	″TCHBIG5_1R0″	Traditional Chinese	big5
1200	1208	″UTF8″	Unicode	UTF-8
0	819	″Latin1_0A″	English (Latin 1)	ISO8859-1
0	1252	″Latin1252_0A″	English (Win Latin)	ISO8859-1/15
0	819	″ASCII″	English (ASCII)	ISO8859-1

4. If you do not have the required character set installed, install the character set to use the Teradata wrapper.
   - ASCII is enabled on the Teradata server, but it is not cataloged in the dbc.chartranslations table. If all of the values that are returned for InstallFlag are ′N′, ASCII is the only valid character set on the Teradata server and the TERADATA_CHARSET environment variable must be set to ASCII in the db2dj.ini file.
   - If the character set that you want to use is listed in the dbc.chartranslations table, but the InstallFlag value is set to ′N′, issue the following statement to change the InstallFlag to ′Y′:

     ```
     update dbc.chartranslations
          set installflag='Y' where CharSetName= 'character_set_name';
     ```
   - If the character set that you want to use is not listed in the dbc.chartranslations table, contact Teradata customer support.
5. Restart the Teradata server to update the list of character sets. In a Teradata command window, enter: tpareset -f reason_for_restart

Registering the Teradata wrapper

You must register a wrapper to access Teradata data sources. Federated servers use wrappers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.

Procedure

To register the Teradata wrapper:

Use one of the following methods:

Method	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.

Issue the CREATE WRAPPER statement and specify the default name for the Teradata wrapper.	For example:
	`CREATE WRAPPER TERADATA;`
	Remember: When you register the wrapper by using the default name, TERADATA, the federated server automatically uses the appropriate Teradata wrapper library for the operating system that your federated server is running on.
	If the default wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name that you choose. When you use a name that is different from the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement.
	For example, to register a wrapper with the name tera_wrapper on a federated server that uses AIX, issue the following statement:
	`CREATE WRAPPER tera_wrapper`
	`  LIBRARY 'libdb2teradata.a'`
	`  OPTIONS (DB2_FENCED 'Y',`
	`  DB2_SOURCE_CLIENT_MODE '32BIT');`
	The wrapper library file that you specify depends on the operating system of the federated server.
	Mandatory options for AIX and Solaris
	Note: The DB2_FENCED and DB2_SOURCE_CLIENT_MODE wrapper options in the CREATE WRAPPER statement are mandatory for AIX and Solaris because the Teradata wrapper supports only 32-bit clients.
	For example:
	`CREATE WRAPPER TERADATA`
	`  OPTIONS (DB2_FENCED 'Y',`
	`  DB2_SOURCE_CLIENT_MODE '32BIT')`

After you complete this task, you can register the server definition.

Teradata wrapper library files

The Teradata wrapper library files are added to the federated server when you install the wrapper.

When you install the Teradata wrapper, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2teradata.a, libdb2teradataF.a, and libdb2teradataU.a. The default wrapper library file is libdb2teradata.a. The other wrapper library files are used internally by the Teradata wrapper.

If you do not use the default wrapper name when you register the wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

The default directory paths and default wrapper library file names are listed in the following table.

Table 84. Teradata wrapper library locations and file names

Operating system	Directory path	Library file names
AIX	/usr/opt/*install_path*/lib32/ /usr/opt/*install_path*/lib64/	libdb2teradata.a
Solaris	/opt/IBM/db2/*install_path*/lib32 /opt/IBM/db2/*install_path*/lib64	libdb2teradata.so
Windows	%DB2PATH%\bin	db2teradata.dll

install_path is the directory path where IBM WebSphere Federation Server is installed on UNIX.

Registering the server definitions for a Teradata data source

You must register each Teradata server that you want to access in the federated database.

Procedure

To register a server definition for an Teradata data source:

1. Locate the hosts file.
   - On federated servers that run AIX, the hosts file is located in the /etc/hosts directory.
   - On federated servers that run Windows, the hosts file is located in the %WINDIR%\system32\drivers\etc\hosts directory.
2. Search the hosts file for the alias of the remote server.

   This alias begins with an alphabetic string and ends with the suffix COP*n*. The value *n* is the number of the application processor that is associated with the Teradata communications processor.
3. Find the first non-numeric field on the line in the hosts that contains the alias.

   Sample hosts file:

   ```
   127.0.0.1       localhost

   9.22.5.77       nodexyz        nodexyzCOP1      # teradata server

   9.66.111.133   rtplib05.data.xxx.com aap
   9.66.111.161   rtpscm11.data.xxx.com aaprwrt
   9.66.111.161   rtpscm11.data.xxx.com accessm
   ```

   In this example, `nodexyz` is the node name.

4. To create the server, use one of the following methods:

Methods	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.
Issue the CREATE SERVER statement.	For example: `CREATE SERVER server_definition_name TYPE TERADATA VERSION version_number WRAPPER wrapper_name OPTIONS (NODE 'node_name');` Although the *'node_name'* variable is specified as an option in the CREATE SERVER statement, this option is required for Teradata data sources.

Creating the user mapping for a Teradata data source

When you attempt to access an Teradata server, the federated server establishes a connection to the Teradata server by using a user ID and password that are valid for that data source. You must define an association (a user mapping) between each federated server user ID and password and the corresponding data source user ID and password.

About this task

Create a user mapping for each user ID that will access the federated system to send distributed requests to the Teradata data source.

Procedure

To create the user mappings for a Teradata data source:

Issue a CREATE USER MAPPING statement.

For example:

```
CREATE USER MAPPING FOR local_userID SERVER server_definition_name
       OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password')
```

Although the REMOTE_AUTHID and REMOTE_PASSWORD variables are specified as options in the CREATE USER MAPPING statement, these options are required to access Teradata data sources.

After you complete this task, test the connection from the federated server to the Teradata server.

Testing the connection from the federated server to the Teradata server

Test the connection to the Teradata data source server to determine if the federated server is properly configured to access Teradata data sources.

About this task

You can test the connection to the Teradata server by using the server definition and user mappings that you defined.

Procedure

To test the connection to the Teradata server:

Open a pass-through session and issue a SELECT statement on the Teradata system tables. If the SELECT statement returns a count, your server definition and your user mapping are set up properly.

For example:

```
SET PASSTHRU server_definition_name
SELECT count(*) FROM dbc.tables
SET PASSTHRU RESET
```

If the SELECT statement returns an error, you should troubleshoot the connection errors.

After you complete this task, you can register the nicknames for Teradata tables and views.

Registering nicknames for Teradata tables and views

For each Teradata server definition that you register, you must register a nickname for each table or view that you want to access. Use these nicknames, instead of the names of the data source objects, when you query the Teradata servers.

Before you begin

The federated database relies on the data source catalog statistics to optimize query processing. To ensure that the federated database has complete statistics on Teradata tables, use the COLLECT STATISTICS Teradata command before you register a nickname.

From the Teradata server, use the COLLECT STATISTICS Teradata command to collect statistics on one or more columns or indexes in a table.

When you register the nickname with the CREATE NICKNAME statement, the federated database reads the statistics from the Teradata system catalog and updates the local statistics for the nickname.

About this task

When you register a nickname on a Teradata view, the federated database recognizes all columns of the view as nullable, even if the columns in the Teradata view do not allow null values. There is no workaround for this limitation.

Procedure

To register a nickname for a Teradata table or view:

Method	Description
Use the Federated Objects wizard in the DB2 Control Center.	To start the wizard, right-click the Federated Database Objects folder and click **Create Federated Objects**.

Method	Description

Issue the CREATE NICKNAME statement. Nicknames can be up to 128 characters in length.	For example:
	`CREATE NICKNAME nickname`
	` FOR server_definition_name.remote_schema.remote_table;`

When you create the nickname, the federated server queries the data source catalog by using the nickname. This query tests the connection to the data source table or view. If the connection does not work, you receive an error message.

Repeat this step for each Teradata table or view that you want to create a nickname for.

Teradata nicknames on federated servers

When you query a Teradata data source from a federated server, you use a nickname in the query to identify the Teradata table and view that you want to access.

When you create a nickname for a Teradata table or view, the federated server connects to the Teradata server associated with the table or view. The federated server uses the nickname to verify the connection to the Teradata server. The federated database verifies the presence of the table or view at the data source and then attempts to gather statistical data about the Teradata table or view from the catalog on the Teradata server. The statistics that are gathered about the nicknamed object are stored in the global catalog on the federated server.

The federated server relies on the statistics that it collects for the nicknamed objects to optimize query processing. Because some or all of the Teradata catalog information might be used by the query optimizer, you should update the statistics at the Teradata server before you create a nickname. Update the statistics at the Teradata server by using a command or utility that is equivalent to the DB2 RUNSTATS command.

You cannot submit an INSERT, UPDATE, or DELETE statement to a nickname that references an updatable Teradata view unless the SQL statement can be completely pushed down to the Teradata data source.

CREATE SERVER statement - Examples for the Teradata wrapper

Use the CREATE SERVER statement to register server definitions for the Teradata wrapper. This topic includes a complete example with the required parameters, and an example with additional server options.

The following example shows you how to register a server definition for a Teradata wrapper by issuing the CREATE SERVER statement:

```
CREATE SERVER tera_server TYPE TERADATA
     VERSION 2.5 WRAPPER my_wrapper
     OPTIONS (NODE 'tera_node');
```

tera_server
A name that you assign to the Teradata database server. Duplicate server definition names are not allowed.

TYPE *TERADATA*
: Specifies the type of data source server to which you are configuring access. For the Teradata wrapper, the server type must be `TERADATA`.

VERSION *2.5*
: The version of the Teradata database server that you want to access.

WRAPPER *TERADATA*
: The wrapper name that you specified in the CREATE WRAPPER statement.

NODE *'tera_node'*
: The name of the node where the Teradata database server resides. Obtain the node name from the hosts file. This value is case sensitive.

Although the name of the node is specified as an option in the CREATE SERVER statement, it is required for Teradata data sources.

Server options

When you create a server definition, you can specify additional server options in the CREATE SERVER statement. The server options can be general server options and Teradata-specific server options.

The CPU_RATIO and IO_RATIO server options provide the statistical information about the Teradata server to the query optimizer. To specify that the CPU resources of the federated server are twice as powerful as the CPU resources of the Teradata server, set the value of the CPU_RATIO server option to 2.0. To specify that the I/O devices of the federated server process data three times faster than the I/O devices of the Teradata server, set the IO_RATIO server option to 3.0.

The following example shows a Teradata server definition with these options:

```
CREATE SERVER tera_server TYPE TERADATA
     VERSION 2.5 WRAPPER my_wrapper
     OPTIONS (NODE 'tera_node', CPU_RATIO '2.0', IO_RATIO '3.0');
```

CREATE USER MAPPING statement - Examples for the Teradata wrapper

Use the CREATE USER MAPPING statement to map a federated server user ID to an Teradata server user ID and password. This topic includes a complete example with the required parameters, and an example that shows you how to use the DB2 special register USER with the CREATE USER MAPPING statement.

The following example shows how to map a local federated user ID to a Teradata server user ID and password:

```
CREATE USER MAPPING FOR MICHAEL SERVER tera_server
      OPTIONS (REMOTE_AUTHID 'mike', REMOTE_PASSWORD 'passxyz123');
```

MICHAEL
: Specifies the local user ID that you are mapping to a user ID that is defined at the Teradata server.

SERVER *tera_server*
: Specifies the server definition name that you registered in the CREATE SERVER statement for the Teradata server.

REMOTE_AUTHID *'mike'*
: Specifies the user ID at the Teradata database server to which you are

mapping *MICHAEL*. Use single quotation marks to preserve the case of this value unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

Although the remote user ID is specified as an option in the CREATE SERVER statement, it is required for Teradata data sources.

REMOTE_PASSWORD *'passxyz123'*

Specifies the password that is associated with *'mike'*. Use single quotation marks to preserve the case of this value unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

Although the remote password is specified as an option in the CREATE SERVER statement, it is required for Teradata data sources.

DB2 special register USER

You can use the DB2 special register USER to map the authorization ID of the person who is issuing the CREATE USER MAPPING statement to the data source authorization ID that is specified in the REMOTE_AUTHID user option.

The following example shows a CREATE USER MAPPING statement that includes the special register USER:

```
CREATE USER MAPPING FOR USER SERVER tera_server
      OPTIONS (REMOTE_AUTHID 'mike', REMOTE_PASSWORD 'passxyz123');
```

CREATE NICKNAME statement - Examples for the Teradata wrapper

Use the CREATE NICKNAME statement to register a nickname for an Teradata table or view that you want to access. This topic includes a complete example with the required parameters.

This example shows how to create a nickname for a Teradata table or view on the Teradata server:

```
CREATE NICKNAME TERASALES FOR tera_server.vinnie.europe ;
```

TERASALES

A unique nickname that is used to identify the Teradata table or view.

Important: The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authorization ID of the user who registers the nickname. The authorization ID is for the federated server, not the remote Teradata data source.

tera_server.vinnie.europe

A three-part identifier for the remote object:

- *tera_server* is the server definition name that you assigned to the Teradata database server in the CREATE SERVER statement.
- *vinnie* is the user ID of the owner to which the table or view belongs. This value is case sensitive.
- *europe* is the name of the remote table or view that you want to access.

Troubleshooting data source connection errors

A test connection to the data source server might return an error for several reasons. There are actions that you can take to determine why the error occurred.

Symptom

An error is returned when you attempt to connect to the data source.

Cause

There are several possible causes for a connection problem.

Resolving the problem

To troubleshoot data source connection errors, check the following items for problems:

- Verify that the data source is available.
- If applicable, ensure that the data source server is configured for incoming connections.
- Ensure that your user mapping settings for the REMOTE_AUTHID and REMOTE_PASSWORD options are valid for the connections to the data source. Alter the user mapping, or create another user mapping as necessary.
- If applicable, ensure that the data source client software on the federated server is installed and configured correctly to connect to the data source.
- For ODBC data sources, ensure that the ODBC driver on the federated server is installed and configured correctly to connect to the ODBC data source server. On federated servers that run Windows, use the ODBC Data Source Administrator tool to check the driver. On federated servers that run UNIX, consult the ODBC client vendor's documentation.
- Verify that the settings for the variables set on the federated server are correct for the data source. These variables include the system environment variables, the variables in the db2dj.ini file, and the DB2 Profile Registry (db2set) variables.
- Check your server definition. If necessary, drop the server definition and create it again.

Chapter 20. Web services data sources

Configuring access to Web services data sources

To configure the federated system to access Web services data sources, you must provide the federated server with information about the data sources and objects that you want to access, such as a valid Web services description language (WSDL) document.

Before you begin

- IBM WebSphere Federation Server must be installed on a server that will act as the federated server
- A federated database must exist on the federated server

About this task

You can configure the federated server to access Web services data sources by using the DB2 Control Center or the DB2 command line. The DB2 Control Center includes a wizard to guide you through the steps that are required to configure the federated server.

Procedure

To configure access to Web services data sources:

1. Register the Web services wrapper.
2. Register the server definition for Web services data sources.
3. Register user mappings to enable security for HTTP authentication (optional)
4. Register nicknames for Web services data sources:
   - Register nicknames for Web services data sources by using the DB2 command line.
   - Register nicknames for Web services data sources by using the DB2 Control Center.
5. Create federated views for Web services nicknames.

Web services and the Web services wrapper

Web service providers are described by Web Services Description Language (WSDL) documents. You can use the Web services wrapper to access Web service providers.

The Figure 23 on page 334 diagram shows the architecture of Web services.

1. A Web service provider implements a service and publishes the WSDL information to a service broker, such as UDDI.
2. The service consumer can then use the service broker to find a Web service provider.
3. When the service consumer finds a Web service provider, the service consumer binds to the service provider so that the consumer can use the Web service.
4. The consumer invokes the service by exchanging SOAP (simple object access protocol) messages between the requester and provider.

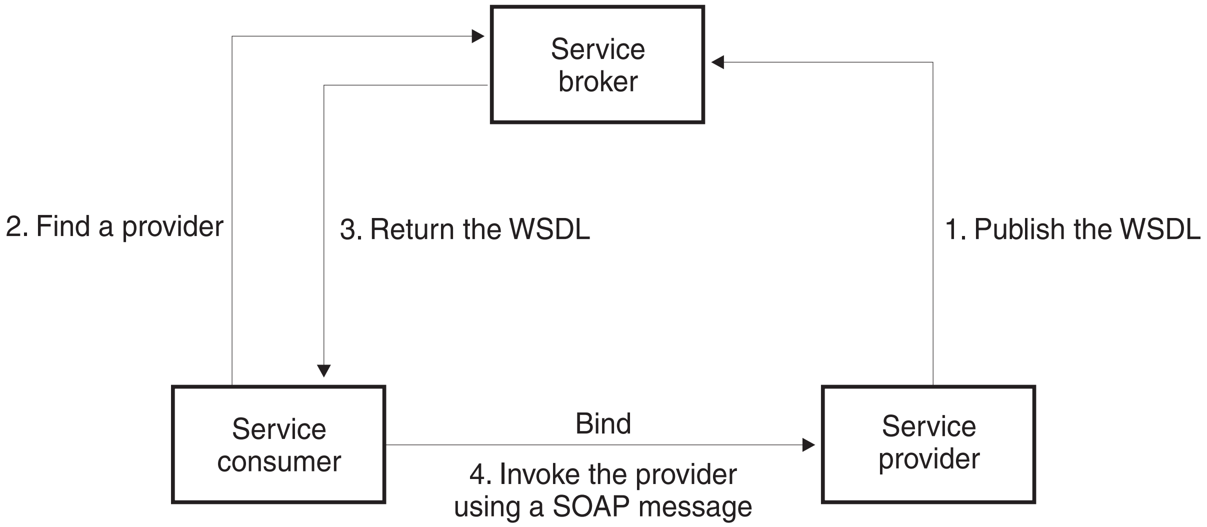


Figure 23. Web services: a service-oriented architecture

The SOAP specification defines the layout of an XML-based message. A SOAP message is contained in a SOAP envelope. The envelope consists of an optional SOAP header and a mandatory SOAP body. The SOAP header can contain information about the message, such as encryption information or authentication information. The SOAP body contains the message. The SOAP specification also defines a default encoding for programming language bindings, which is called the SOAP encoding.

The WSDL document and the Web service

The key to the Web service is the WSDL document. The WSDL document is an XML document that describes Web services in terms of the messages that it sends and receives. Messages are described by using a type system, which is typically the XML schema. A Web service operation associates a message exchange pattern with one or more messages. A message exchange pattern identifies the sequence and cardinality of messages that are sent or received, as well as who the messages are logically sent to or received from. An interface groups together operations without any commitment to the transport or wire format. A WSDL binding specifies transport and wire format details for one or more interfaces. An endpoint associates a network address with a binding. A service groups together endpoints that implement a common interface. The messages can contain document-oriented information or process-oriented information, which is also known as remote procedure calls (RPC). A WSDL document can contain one or more Web services.

The example in Figure 24 on page 335 shows the WSDL definition of a simple service that provides stock quotes. The Web service supports a single operation that is named GetLastTradePrice. The service can be accessed with the SOAP 1.1 protocol over HTTP. The request reads a ticker symbol as input, which is a string data type, and returns the price, which is a float data type. The string and float data types are predefined types in the XML schema standards. A Web service can also define data types and use those user-defined data types in messages. Predefined and user-defined XML data types map to columns of the nicknames. The complete example and the WSDL specification is at the W3C Web site.

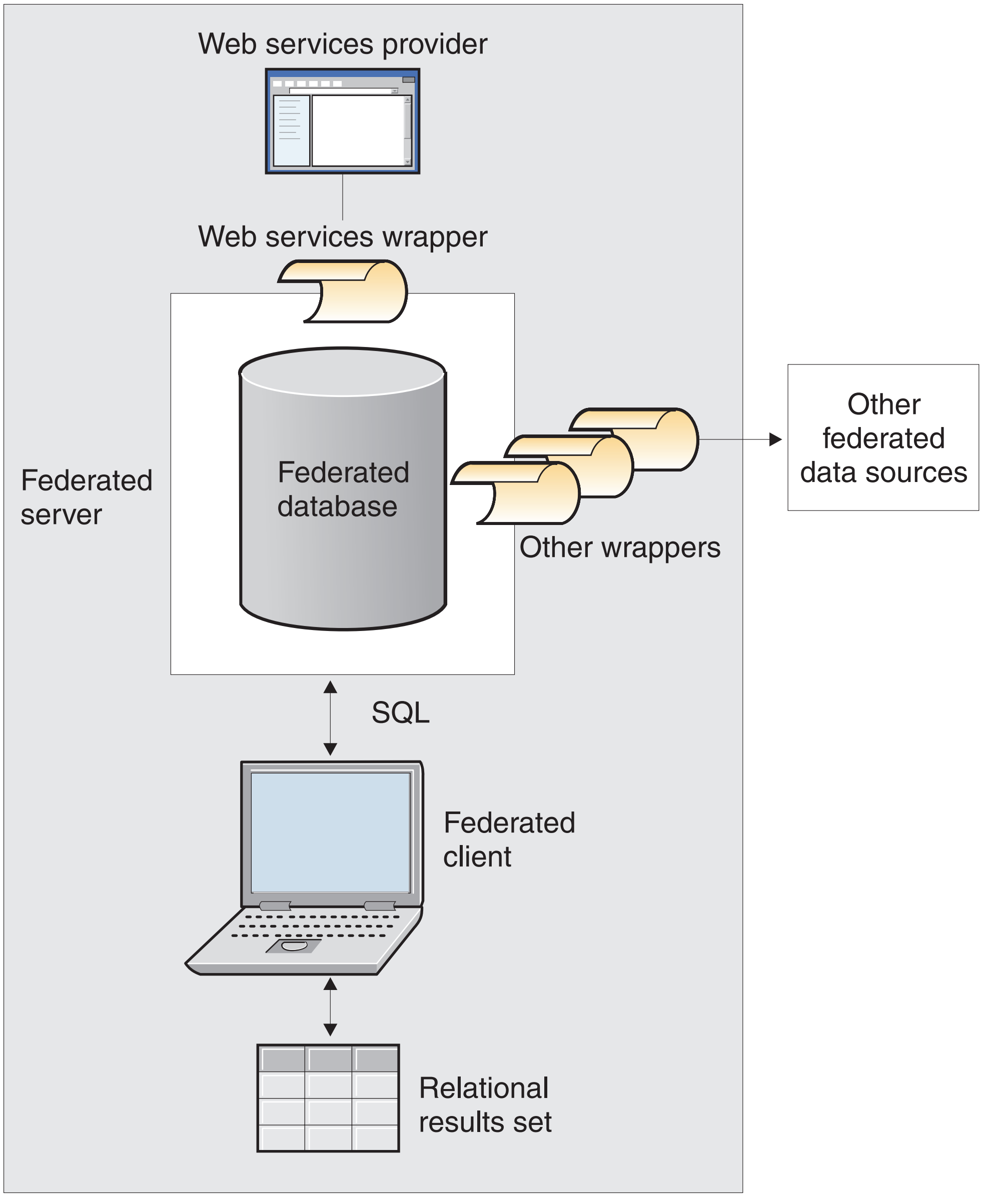
```
<?xml version="1.0"?>
<definitions name="StockQuote"
...


<types>
       <schema targetNamespace="http://example.com/stockquote.xsd"
              xmlns="http://www.w3.org/2000/10/XMLSchema">
           <element name="TradePriceRequest">
              <complexType>
                  <all>
                      <element name="tickerSymbol" type="string"/>
                  </all>
              </complexType>
           </element>
           <element name="TradePrice">
              <complexType>
                  <all>
                      <element name="price" type="float"/>
                  </all>
              </complexType>
           </element>
       </schema>
    </types>

<message name="GetLastTradePriceInput">
...
</message>

    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
           <input message="tns:GetLastTradePriceInput"/>
           <output message="tns:GetLastTradePriceOutput"/>
        </operation>
    </portType>


    <binding name="StockQuoteSoapBinding"
          type="tns:StockQuotePortType">
        <soap:binding style="document"
          transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetLastTradePrice">
           <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
           <input>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
           </output>
        </operation>
    </binding>

   <service name="StockQuoteService">
        <documentation>My first service</documentation>
        <port name="StockQuotePort" binding="tns:StockQuoteBinding">
           <soap:address location="http://example.com/stockquote"/>
        </port>
    </service>
</definitions>
```

Figure 24. Example of a WSDL document

The WSDL document, Web services wrapper, and nicknames

The Web services wrapper uses the operations in a port type that have a SOAP binding with an HTTP transport. The input messages in the operation, and the associated types or elements become columns in the nickname. The output messages in the operation are extracted into the nickname hierarchy. You can create a separate hierarchy of nicknames for each operation in the WSDL document.

By using the Web services wrapper, you can use the federated systems functions to join data from Web services with data on other federated data sources.



The example Figure 25 on page 337 uses a WSDL document that contains a portType with an operation name of GETTEMP. With this Web service, you enter a zip code as input and receive a temperature for that zip code.

```
<?xml version="1.0"?>
<definitions name="TemperatureService" targetNamespace=http://www.xmethods.net/
   sd/TemperatureService.wsdl"
 xmlns:tns="http://www.xmethods.net/sd/TemperatureService.wsdl"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns="http://schemas.xmlsoap.org/wsdl/">
 <message name="getTempRequest">
   <part name="zipcode" type="xsd:string"/>
 </message>
<message name="getTempResponse">
   <part name="return" type="xsd:float"/>
</message>
<portType name="TemperaturePortType">
  <operation name="getTemp">
      <input message="tns:getTempRequest"/>
      <output message="tns:getTempResponse"/>
  </operation>
</portType>
<binding name="TemperatureBinding" type="tns:TemperaturePortType">
  <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="getTemp">
     <soap:operation soapAction="" />
    <input>
     <soap:body use="encoded" namespace="urn:xmethods-Temperature"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:xmethods-Temperature"
         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>
</binding>
<service name="TemperatureService">
  <documentation>
      Returns current temperature in a given U.S. zipcode
  </documentation>
  <port name="TemperaturePort" binding="tns:TemperatureBinding">
     <soap:address
       location="http://services.xmethods.net:80/soap/servlet/rpcrouter" />
  </port>
</service>
</definitions>
```

Figure 25. GETTEMP Web service

The input value is described by the zipcode column of the nickname. The output value is described by the return column of the nickname. In the WSDL document, those columns are identified in the messages element. The messages element represents the logical definition of the data that is sent between the Web service provider and the Web service consumer. If more explanation of the information in the message element is needed, then the WSDL document can also contain a type element. The type element can refer to predefined types that are based on the XML schema specifications or types that are defined by a user.

The example Figure 26 on page 338 shows the nickname that the DB2® Control Center Discovery tool produces from the GETTEMP Web service WSDL document. The zipcode column is a required input column because of the nickname TEMPLATE syntax:

```
CREATE NICKNAME GETTEMP (
  ZIPCODE VARCHAR (48) OPTIONS(TEMPLATE '&column'),
   RETURN VARCHAR (48) OPTIONS(XPATH './return/text()')
   )
  FOR SERVER "EHPWSSERV"
   OPTIONS(URL 'http://services.xmethods.net:80/soap/servlet/rpcrouter',
           SOAPACTION ' ' ,
           TEMPLATE '<soapenv:Envelope>
                       <soapenv:Body>
                          <ns2:getTemp>
                            <zipcode>&zipcode[1,1]</zipcode>
                          </ns2:getTemp>
                       </soapenv:Body>
                     </soapenv:Envelope>',
           XPATH '/soapenv:Envelope/soapenv:Body/*' ,
           NAMESPACES ' ns1="http://www.xmethods.net/sd/TemperatureService.wsdl",
                        ns2="urn:xmethods-Temperature" ,
                          soapenv="http://schemas.xmlsoap.org/soap/envelope/"');
```

Figure 26. GETTEMP nickname

The nickname options in the Web services wrapper, URL and SOAPACTION, provide the ability to override the endpoint, or the address that you specified when you created the nickname. When you use the URLCOLUMN or SOAPACTIONCOLUMN enabled columns in a query, you can use dynamic addresses with the same nicknames. If you define the nickname options URL and SOAPACTION when you create a nickname and enable the URLCOLUMN and SOAPACTIONCOLUMN on the column option, then you are using the late binding functions of Web services wrappers. The value for the SOAPACTION nickname option becomes an attribute in the HTTP header. The value for the URL nickname option is the HTTP URL to which the request is sent.

The URL and SOAPACTION nickname options provide dynamic nickname associations. These dynamic addresses are useful if several companies implement a Web service portType. The Web services wrapper requires that the only differences between the WSDL documents are different URLs and SOAPACTIONS. You can use the late binding function to create and use the same nickname for different service endpoints that different companies might want to use. The URL and SOAPACTION values are derived from the WSDL document.

The following example shows how you can use the URLCOLUMN and SOAPACTIONCOLUMN column options:

```
CREATE NICKNAME GetPartQuote(
  partnumber INTEGER OPTIONS (TEMPLATE'&column'),
  price FLOAT OPTIONS (XPATH './price')),
  urlcol VARCHAR(100) OPTIONS (URLCOLUMN 'Y'),
  soapactioncol VARCHAR(100) OPTIONS (SOAPACTIONCOLUMN 'Y'),
 FOR SERVER myServer
  OPTIONS (
  ...
  SOAPACTION 'http://example.com/GetPartPrice' ,
  URL 'http://mycompany.com:9080/GetPartPrice'',
  ...
   )
```

Figure 27. GetPartQuote nickname

The following example uses the columns URLCOL and SOAPACTIONCOL that were defined with the URLCOLUMN column option enabled and the SOAPACTIONCOLUMN column option enabled:

```
SELECT * FROM supplier_endpoints p,
    GetPartQuote q
 WHERE partnumber=1234 AND
       p.url=q.urlcol AND
       p.soapaction=q.soapactioncol;
```

The SQL application can defer choosing which endpoints to use until the time that a query is run, instead of defining a specific endpoint at the time that the nickname is created.

The Web services wrapper can separate a large amount of WSDL document data into fragments to decrease the total memory that is used. Specify the **STREAMING** option in the DB2 Control Center in the Settings page of the Properties window, when you create a Web services nickname. The Web services wrapper processes the resulting stream of XML data and then extracts the information that is requested by a query fragment. The Web services wrapper parses one fragment at a time. Use the **STREAMING** option to parse large XML documents only.

Registering the Web services wrapper

You must register a wrapper to access Web services data sources. Wrappers are used by federated servers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.

Before you begin

See the list of Web services wrapper library files for the correct name to specify in the CREATE WRAPPER statement.

About this task

The name of the wrapper library file that you specify depends on the operating system of the federated server.

Procedure

To register a wrapper:

Issue a CREATE WRAPPER statement with the name of the wrapper and the name of the wrapper library file. For example, to register a wrapper with the name websr_wrapper on a federated server that uses Windows, issue the following statement:

```
CREATE WRAPPER websr_wrapper LIBRARY 'db2ws.dll';
```

Web services wrapper library files

The Web services wrapper library files are added to the federated server when you install the federated server.

When you install IBM WebSphere Federation Server, library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files added to the directory path are libdb2ws.a, libdb2wsF.a, and libdb2wsU.a. The default wrapper library file is libdb2ws.a. The other wrapper library files are used with specific wrapper options.

You must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

The default directory paths and default wrapper library file names are listed in the following table.

Table 85. Library locations and file names for CREATE WRAPPER

Operating system	Directory path	Wrapper library file
AIX	/usr/opt/<install_path>/lib32/ /usr/opt/<install_path>/lib64/	libdb2ws.a
Linux	/opt/IBM/db2/<install_path>/lib32 /opt/IBM/db2/<install_path>/lib64	libsb2ws.so
Solaris	/opt/IBM/db2/<install_path>/lib32 /opt/IBM/db2/<install_path>/lib64	libdb2ws.so
Windows	%DB2PATH%\bin	db2ws.dll

- <install_path> is the directory path where IBM WebSphere Federation Server is installed on UNIX or Linux.

  %DB2PATH% is the environment variable that is used to specify the directory path where IBM WebSphere Federation Server is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

Registering the server definition for Web services data sources

You can register a server definition by using the DB2 Control Center or by using the DB2 command line. This task describes how to register a Web services server definition from the DB2 command line.

About this task

A server definition must be registered for each Web service that you want to access.

Procedure

To register a server definition to the federated system for the Web services wrapper, issue the CREATE SERVER statement.

For example, to register a Web services server definition named ws_server on Windows, issue the following statement:

```
CREATE SERVER ws_server WRAPPER websr_wrapper;
```

You can set optional timeout and proxy server parameters for the CREATE SERVER statement.

CREATE SERVER statement - example for Web services wrapper

Use the CREATE SERVER statement to register server definitions for the Web services wrapper with time out and proxy server settings.

Even if you are not using a proxy server to access Web services documents, you must still register a server definition. The hierarchy of federated objects requires that the Web services files are associated with a specific server definition object. The statement that you issue to register a server definition is:

```
CREATE SERVER my_server WRAPPER my_wrapper
   OPTIONS (TIMEOUT '60');
```

my_server
A name that you assign to the Web services server definition. Duplicate server definition names are not allowed.

WRAPPER my_wrapper
The wrapper name that you specified in the CREATE WRAPPER statement.

TIMEOUT '60'
Specifies the time, in minutes, that the federated server should wait for a network transfer and the computation of a result.

Server definitions when a proxy server is used

You must use the proxy server options in the CREATE SERVER statement if all of the following conditions are true:

- You want to retrieve data using a URI
- The URI used will retrieve data from behind a firewall, through a proxy
- The firewall or proxy used is HTTP

The options that you specify depend on the type of proxy server that you want to access.

Check with your network administrator for information about the type of proxy that you use, and the settings that you should specify in the proxy options.

Registering a server definition for an HTTP proxy server

To register a server definition and specify an HTTP proxy server, use the following statement:

```
CREATE SERVER ws_server_http
    WRAPPER ws_wrapper
    OPTIONS (PROXY_TYPE 'HTTP', PROXY_SERVER_NAME 'proxy_http',
        PROXY_SERVER_PORT '8080');
```

ws_server_http
A name that you assign to the Web services server definition. Duplicate server definition names are not allowed.

WRAPPER *ws_wrapper*
The wrapper name that you specified in the CREATE WRAPPER statement.

PROXY_TYPE *'HTTP'*
Specifies the proxy type that is used to access the Internet when behind a firewall.

PROXY_SERVER_NAME *'proxy_http'*
Specifies the proxy server name or IP address.

PROXY_SERVER_PORT *'8080'*
Specifies the proxy server port number.

Registering a server definition for a HTTP proxy server with authentication information

To register a server definition and specify a HTTP proxy server with authentication information, use the following statement:

```
CREATE SERVER ws_server_http
    WRAPPER ws_wrapper
    OPTIONS (PROXY_TYPE 'HTTP', PROXY_SERVER_NAME 'proxy_http',
        PROXY_SERVER_PORT '1081', PROXY_AUTHID 'Sampson',
        PROXY_PASSWORD 'sailing4me');
```

PROXY_AUTHID *'Sampson'*
Specifies the user name on the proxy server.

PROXY_PASSWORD *'sailing4me'*
Specifies the password on the proxy server that is associated with the user name *Sampson*.

Enabling security through the Web services wrapper

The Web services wrapper supports HTTP authentication by using the CREATE USER MAPPING statement.

The Web services wrapper supports HTTPS as a transport protocol for SOAP messages. The WSDL document that is generated by the Web services provider contains https:// in the URL. Thus, the SOAP messages in the HTTP request or HTTP response are encrypted.

If the Web service uses HTTPS as a transport protocol, you can configure the wrapper to validate the SSL certificates that the server sends for identification by using the SSL_VERIFY_SERVER_CERTIFICATE wrapper or server option. The Web service wrapper can call Web services with self-signed certificates.

The user mapping is optional. If you do not specify a user mapping, you might see an error if the Web service provider expects authentication information. Some servers might use authentication to restrict access to a service. The need for authentication is not apparent from the information in the WSDL document.

Procedure

To map a federated server user ID to a Web services user ID and password:

Issue the CREATE USER MAPPING statement.

For example, with the following CREATE USER MAPPING statement, when the Web services nickname on the S1 server is accessed, the HTTP request is sent with SYSTEM as the user ID and MANAGER as the password.

```
CREATE USER MAPPING
  FOR RSPALTEN SERVER S1
  OPTIONS ( REMOTE_AUTHID 'SYSTEM', REMOTE_PASSWORD 'MANAGER'
   PROXY_AUTHID 'ID' PROXY_PASSWORD 'PWD'
   SSL_CLIENT_CERTIFICATE_LABEL 'LABEL');
```

Registering nicknames for Web services data sources

For each Web services server definition that you register, you must register a nickname for each data source that you want to access. You can register a nickname by using the command line or the DB2 Control Center.

Procedure

To register nicknames for Web services data sources:

Select one of the following methods:
- Registering nicknames for Web services data sources (DB2 command line)
- Registering nicknames for Web services data sources (DB2 Control Center)

CREATE NICKNAME statement – examples for the Web services wrapper

When you create a nickname to access a Web service, you create an input column for each value in the input message of a Web service operation and an output column for each value in the output message of a Web service operation. You control the input and output column definitions with the nickname column option definitions.

The TEMPLATE column option specifies that a column is an input column. The XPATH column option specifies that a column is an output column. When the TEMPLATE nickname option contains a bracketed notation (`[1,1]`), the column is a required input column. When the TEMPLATE nickname option contains a bracketed notation (`[0,1]`), the column is an optional input column.

The NAMESPACES nickname option is a comma-separated list of name-value pairs that a federated system uses to resolve the namespaces for elements in input and output XML documents. The namespaces are used in the message request so that the prefixes in the TEMPLATE nickname option are defined. The NAMESPACES nickname option resolves the prefixes in XPath expressions with the namespace URIs that are defined in the WSDL or XML schemas. The XPath expressions are applied on the XML document that is returned from the Web service.

Example 1: Required input columns

The following example shows a nickname for a Web service named getQuote. The Web service reads a stock ticker symbol and returns a trading price. The following DDL statement is created by the Discovery tool in the DB2 Control Center.

```
CREATE NICKNAME "stockquote.stockquoteport_getquote_nn" (
 symbol VARCHAR (48) OPTIONS(TEMPLATE '&column'),
 result VARCHAR (48) OPTIONS(XPATH './Result/text()'))
FOR SERVER "xmethods_server"  OPTIONS(
 URL 'http://66.28.98.121:9090/soap' ,
 SOAPACTION 'urn:xmethods-delayed-quotes#getQuote' ,
 TEMPLATE '<soapenv:Envelope>
                  <soapenv:Body>
                   <ns2:getQuote>
                     <symbol>&symbol[1,1]</symbol>
                   </ns2:getQuote>
                  </soapenv:Body>
                 </soapenv:Envelope>',
   XPATH '/soapenv:Envelope/soapenv:Body/*' ,
   NAMESPACES 'ns2="urn:xmethods-delayed-quotes" ,
             ns1="http://www.example.com/wsdl/
                 net.xmethods.services.stockquote.StockQuote/" ,
                 soapenv="http://schemas.xmlsoap.org/soap/envelope/" ');
```

The nickname TEMPLATE option specifies the column SYMBOL as a required input column, because the column contains the [1,1] designation. In the nickname TEMPLATE option, the complete SOAP envelope is specified for the Web service. The getQuote input value is contained in the SOAP envelope and body elements. The XPATH nickname option contains the information to find the trading price value through the SOAP envelope and body tags.

Use the stockquote.stockquoteport_getquote_nn nickname to access the Web service, such as in the following query:

```
SELECT * FROM "stockquote.stockquoteport_getquote_nn"
  WHERE symbol='IBM';
```

You must use the predicate, `symbol='IBM'`, in this statement because `symbol` is a required input column. The equality predicate is the only valid predicate on input columns. Each of the equality predicates sets a value in the input message. If the input column is optional, an equality predicate on that column is not necessary. If the input column is required, then you must issue the query with an equality predicate. You can use a literal value such as `IBM` in an equality expression or a value from a joined table or nickname.

Example 2: Repeating elements and child nicknames

The following example uses a Web service named getZooReport that produces a report for zoos. The input value is a zoo identifier. The output value is a report that is described by the following schema:

```
<wsdl:definitions name="Name"
    targetNamespace="http://myzoo.com"
...
<wsdl:types>
 <xsd:schema elementFormDefault="qualified" targetNamespace="http://myzoo.com"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="Animal">
      <xsd:complexType>
         <xsd:sequence>
          <xsd:element ref="tns:Name"/>
          <xsd:element ref="tns:Species"/>
          <xsd:element ref="tns:Lot"/>
         </xsd:sequence>
      </xsd:complexType>
  </xsd:element>
  <xsd:element name="AnimalCareList">
     <xsd:complexType>
       <xsd:sequence>
         <xsd:element maxOccurs="unbounded" minOccurs="1" ref="tns:Animal"/>
       </xsd:sequence>
     </xsd:complexType>
  </xsd:element>
  <xsd:element name="Count" type="xsd:string"/>
  <xsd:element name="LastModified" type="xsd:string"/>
  <xsd:element name="Lot" type="xsd:string"/>
  <xsd:element name="Name" type="xsd:string"/>
  <xsd:element name="NumberOfCages" type="xsd:string"/>
  <xsd:element name="Species" type="xsd:string"/>
  <xsd:element name="Zoo">
     <xsd:complexType>
       <xsd:sequence>
         <xsd:element ref="tns:ZooName"/>
         <xsd:element ref="tns:Count"/>
         <xsd:element ref="tns:LastModified"/>
         <xsd:element maxOccurs="unbounded" minOccurs="0" ref="tns:Zookeeper"/>
       </xsd:sequence>
       <xsd:attribute name="id" type="xsd:string" use="optional"/>
      </xsd:complexType>
 </xsd:element>
 <xsd:element name="ZooName" type="xsd:string"/>
 <xsd:element name="Zookeeper">
     <xsd:complexType>
       <xsd:sequence>
         <xsd:element ref="tns:Name"/>
         <xsd:element ref="tns:NumberOfCages"/>
         <xsd:element ref="tns:AnimalCareList"/>
       </xsd:sequence>
      <xsd:attribute name="id" type="xsd:string" use="optional"/>
    </xsd:complexType>
</xsd:element>
</xsd:schema>
</wsdl:types>
...
```

Figure 28. getZooReport Web service

The following DDL statement is generated by the DB2 Control Center Discovery tool that is based on the WSDL that contains the schema:

```
CREATE NICKNAME zooport_getzooreport_nn (
 zooid VARCHAR (48) OPTIONS(TEMPLATE '&column'),
 zoo_id VARCHAR (48) OPTIONS(XPATH './ns1:Zoo/@ns1:id'),
 report_zooname VARCHAR (48) OPTIONS(XPATH './ns1:Zoo/ns1:ZooName/text()'),
 report_count VARCHAR (48) OPTIONS(XPATH './ns1:Zoo/ns1:Count/text()'),
 report_lastmodified VARCHAR (48)
       OPTIONS(XPATH './ns1:Zoo/ns1:LastModified/text()'),
 zooport_getzooreport_pkey VARCHAR (16) FOR BIT DATA NOT NULL
       OPTIONS(PRIMARY_KEY 'YES'))
FOR SERVER "zooserver"  OPTIONS(
 URL 'http://localhost:9080/MaelstromTest/services/ZooPort' ,
 SOAPACTION 'http://myzoo.com/getZooReport' ,
 TEMPLATE '<soapenv:Envelope>
              <soapenv:Body>
                <zooId>&zooId[1,1]</zooId>
              </soapenv:Body>
             </soapenv:Envelope>' ,
 XPATH '/soapenv:Envelope/soapenv:Body' ,
     NAMESPACES ' soapenv="http://schemas.xmlsoap.org/soap/envelope/" ,
                  ns1="http://myzoo.com " ');
```

Figure 29. Parent nickname – zooport_getzooreport_nn

```
CREATE NICKNAME zooport_getzooreport_report_nn (
 zooport_getzooreport_fkey VARCHAR (16)
        FOR BIT DATA NOT NULL
           OPTIONS(FOREIGN_KEY 'ZOOPORT_GETZOOREPORT_NN'),
   zookeeper_id VARCHAR (48) OPTIONS(XPATH './ns1:Zookeeper/@ns1:id'),
   report_name VARCHAR (48) OPTIONS(XPATH './ns1:Zookeeper/ns1:Name/text()'),
   report_numberofcages VARCHAR (48)
           OPTIONS(XPATH './ns1:Zookeeper/ns1:NumberOfCages/text()'),
   zooport_getzooreport_pkey VARCHAR (16)
           FOR BIT DATA NOT NULL OPTIONS(PRIMARY_KEY 'YES'))
  FOR SERVER "zooserver"  OPTIONS(
        XPATH './ns1:Zoo' ,
        NAMESPACES ' soapenv="http://schemas.xmlsoap.org/soap/envelope/" ,
               ns1="http://myzoo.com" ');
```

Figure 30. Child of nickname zooport_getzooreport_nn

```
CREATE NICKNAME zooport_getzooreport_report_report_nn (
        zooport_getzooreport_fkey VARCHAR (16) FOR BIT DATA NOT NULL
             OPTIONS(FOREIGN_KEY 'zooport_getzooreport_report_nn'),
        report_name VARCHAR (48)
             OPTIONS(XPATH './ns1:Animal/ns1:Name/text()'),
        report_species VARCHAR (48)
             OPTIONS(XPATH './ns1:Animal/ns1:Species/text()'),
        report_lot VARCHAR (48) OPTIONS(XPATH './ns1:Animal/ns1:Lot/text()'))
 FOR SERVER "zooserver"  OPTIONS(
        XPATH './ns1:Zookeeper/ns1:AnimalCareList' ,
        NAMESPACES ' soapenv="http://schemas.xmlsoap.org/soap/envelope/" ,
                  ns1="http://myzoo.com" ');
```

Figure 31. Child of zooport_getzooreport_report_nn

The schema includes some elements that are repeated, known as sequence elements. These repeated elements become child nicknames of the parent nickname, as shown in Figure 29, Figure 30, and Figure 31. For example, zooname, count, lastmodified, and zookeeper are all elements of zoo. The element zoo contains 0 or more zookeeper elements. The root nickname, zoo, contains the columns zooname, count, and lastmodified. A child nickname, zookeeper, is created by the DB2 Control Center Discovery tool to describe the repeating elements of zookeeper. The third element in the zookeeper column, animalcarelist, also contains 0 or more elements and so it becomes a child nickname, zooport_getzooreport_report_report_nn. The following figure shows the nickname hierarchy:

Root nickname: zooport_getzooreport_nn

Zoo (parent):

- ZooName
- Count
- LastModified
- ZooKeeper (there are 0 or more ZooKeeper elements)

  Child nickname: zooport_getzooreport_report_nn

  ZooKeeper elements

  – Name
  – NumberOfCages
  – AnimalCareList (there are 0 or more Animal elements)

    Child nickname: zooport_getzooreport_report_report_nn

    Animal

    - Name
    - Species
    - Lot

Figure 32. Parent —> Child —> nickname hierarchies

The following statement is a typical query that you might issue on the nicknames to access the zoo report Web service. When you issue this statement, you retrieve the information from the zoo report based on a specific identifier and on where the primary and foreign keys of the child nickname zoo reports match.

```
SELECT * FROM zooport_getzooreport_nn ,
       zooport_getzooreport_report_nn zk ,
       zooport_getzooreport_report__report__nn a
   WHERE zooid='1'AND zooport_getzooreport_pkey=zk.zooport_getzooreport_fkey
   and zk.zooport_getzooreport_pkey=a.zooport_getzooreport_fkey;
```

Example 3: Late binding

The following example shows how you can use the late binding option. You can use this option from the DB2 Control Center or from a DB2 command line. If you define the nickname options URL and SOAPACTION and if you enable the column options URLCOLUMN and SOAPACTIONCOLUMN when you create a nickname, you are using the late binding functions. The DB2 Control Center creates the column options, URLCOLUMN and SOAPACTIONCOLUMN, and sets the values of the columns to `yes`.

The following example is for a Web service that provides price quotes for parts that is implemented by all suppliers for a company. Here is the CREATE NICKNAME statement that includes the URLCOLUMN and SOAPACTIONCOLUMN definitions:

```
CREATE NICKNAME GetPartQuote(
  partnumber INTEGER OPTIONS (TEMPLATE'&column'),
  price FLOAT OPTIONS (XPATH './price')),
  urlcol VARCHAR(100) OPTIONS (URLCOLUMN 'Y'),
  soapactioncol VARCHAR(100) OPTIONS (SOAPACTIONCOLUMN 'Y'),
 FOR SERVER myServer
  OPTIONS (
  ...
  SOAPACTION 'http://example.com/GetPartPrice' ,
  URL 'http://mycompany.com:9080/GetPartPrice'',
  ...
   )
```

To get price quotes from all of the suppliers with a single query, the values that each supplier uses for the SOAPACTION and URL column options are needed. The query looks like this:

```
SELECT * FROM supplier_endpoints p,
    GetPartQuote q
 WHERE partnumber=1234 AND
       p.url=q.urlcol AND
       p.soapaction=q.soapactioncol;
```

Local table supplier_endpoints contains all of the URLs and SOAP addresses with which you can call the Web service. You can include an ORDER BY price clause to determine the least expensive supplier for this part.

Example 4: ESCAPE_INPUT column option

You can include XML fragments as input values in your query. When you register a nickname, include the column option ESCAPE_INPUT=N. This option maintains the special characters, such as angle brackets (< and >) in XML fragments in the input values.

When a schema contains repeating input values that requires you to send XML as part of the SOAP message, you can use the ESCAPE_INPUT column option to build the output message with the correct XML.

For example, the zoo Web service includes an operation to add a new zoo keeper and the animals that are associated with that zoo keeper. In the schema for this example, an AnimalCareList can have multiple animals.

```
CREATE NICKNAME add_zookeeper(
 zookeeper_id VARCHAR(48)  OPTIONS(TEMPLATE '...'),
 name VARCHAR(48) OPTIONS(TEMPLATE '...'),
 numberofcages VARCHAR(48) OPTIONS(TEMPLATE '...'),
 animals VARCHAR(3000) OPTIONS( TEMPLATE '...' , ESCAPE_INPUT 'N')
...
```

To add a new zoo keeper with two animals, issue a query such as the following example:

```
SELECT * FROM add_zookeeper
  WHERE zookeeper_ID='37' AND
        name='Amit Kapoor' AND
        numberofcages='3'  AND
        animals='<AnimalCareList xmlns="http://myzoo.com">
                  <Animal>
                    <Name>Larry</Name>
                    <Species>Gorilla</Species>
                    <Lot>7</Lot>
                  </Animal>
                  <Animal>
                    <Name>Bill</Name>
```

```
              <Species>Chimpanzee</Species>
              <Lot>8H</Lot>
            </Animal>
          </AnimalCareList>';
```

The add_zookeeper nickname is a Web service operation that can change the state of the Web service, or update information. Although nonrelational wrappers cannot be updated, the SELECT statement in this example updates the zoo information to add a new zoo keeper.

You can also use the ESCAPE_INPUT column option for a schema with an element such as xsd:anyType. In this case, the type of the element is unknown. You can use the ESCAPE_INPUT column option on the input column for that element so that you can specify arbitrary XML fragments for your input.

The TEMPLATE option for the Web services wrapper

The Web services wrapper needs the TEMPLATE option on the CREATE NICKNAME statement at the time that the nickname is created.

Web services wrapper

For the Web services wrapper, the required and optional attributes vary according to the definitions in the WSDL document and how a column is derived. A column can be derived from either an element or an attribute of an element.

- If the column is derived from an element, then the minOccurs value determines if a column is optional.
- If the value of minOccurs equals 0, then the column is optional.
- If the value of minOccurs equals 1, then the column is required.
- If the column is derived from an attribute of an element, then the value of use on the attribute determines if a column is optional.
- If an attribute contains the value use=optional, then the column is optional.
- If an attribute contains the value use=required, then the column is required.

The following example is an attribute in a schema definition that is associated with a column:

```
<xsd:complexType>
  <xsd:sequence>
    <xsd:element ref="tns:ZooName"/>
    <xsd:element ref="tns:Count"/>
    <xsd:element ref="tns:LastModified"/>
    <xsd:element maxOccurs="unbounded" minOccurs="0" ref="tns:Zookeeper"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string" use="optional"/>
</xsd:complexType>
```

Creating federated views for Web services nicknames

Create federated views for Web services nicknames to ensure that queries that join pieces of a Web services nickname hierarchy run properly.

About this task

You can define federated views for the hierarchy of nicknames that describe a Web services document. A federated view ensures that the queries that join pieces of a Web services nickname hierarchy can run properly.

The Web services wrapper requires joins from any child nickname to include all nicknames back to the parent nickname. A SELECT statement on only a child nickname fails if the parent nickname is not included in the statement. The preferred method of including all required nicknames in a query is to define a federated view.

Procedure

To create federated views for Web services nicknames:

1. Define a view that includes all of the nicknames that are related to an operation in the Web service if you want to join those nicknames.
2. In the WHERE clause of the view, use join predicates for all columns that are related by the PRIMARY_KEY and FOREIGN_KEY column options.

In the following example, the primary key is on column ooport_getzooreport_pk in nickname zooport_getzooreport_report_nn. The foreign key is on column ooport_getzooreport_fkey in nickname zooport_getzooreport_report_report_nn.

```
CREATE VIEW zooreport
  (zooid, zooname, number_of_zookeeper,
   lastmodified,zookeeper_id, zookeeper_name,
   cages, animal_name, animal_species, animal_lot)
AS ( SELECT zooid, report_zooname,
      report_count, report_lastmodified,
      zookeeper_id, zk.report_name, report_numbercages,
      a.report_name, report_species,
      report_lot
     FROM zooport_getzooreport_nn ,
          zooport_getzooreport_report_nn as zk,
          zooport_getzooreport_report_report_nn as a
     WHERE zk.ooport_getzooreport_pkey=a.ooport_getzooreport_fkey
      AND zooport_getzooreport_pkey=zk.ooport_getzooreport_fkey);
```

You can get information from all of the nicknames with the following SELECT statement:

```
SELECT * FROM zooreport WHERE zooid='1';
```

Web services data sources – example queries

Examples of SQL queries for the Web services wrapper are shown.

Example 1: Using materialized query tables

You use materialized query tables to locally cache the results of a query and to improve the performance of queries. You can use nicknames from Web services data sources to create materialized query tables. For some queries, the database can automatically determine whether the materialized query table can answer a query without accessing the base tables. The following procedure shows how to create and populate a materialized query table:

1. Create a local or base table:

   ```
   CREATE TABLE mystocks(ticker VARCHAR(10));
   ```

   You can use the local table to maintain all the values that you want to cache.
2. Insert all of the values that you want to cache into the table:

   ```
   INSERT INTO mystocks VALUES('IBM');
   INSERT INTO mystocks VALUES('MSFT');
   ...
   ```

3. Create a Web services nickname:

   ```
   CREATE NICKNAME stockquote_nn (
           ticker VARCHAR(40) OPTIONS (TEMPLATE   '&column'),
           price VARCHAR(16) OPTIONS (XPATH   './Result/text()')
           )
    FOR SERVER stock_server
    OPTIONS (TEMPLATE '<ticker>&column</ticker>'
               XPATH './Result/text()' );
   ```

4. Create a view that consists of the nickname and the local table:

   ```
   CREATE VIEW  stock_quote_view (ticker, price)
     AS (
       SELECT nn.ticker, nn.price
        FROM stockquote_nn nn, mystocks s
        WHERE nn.ticker=s.ticker
   );
   ```

5. Create a materialized query table:

   ```
   CREATE TABLE stockquote_MQT (ticker, ticker2, price)
           as (SELECT nn.ticker,s.ticker as ticker2, nn.price
     FROM stockquote_nn nn, mystocks s
     WHERE nn.ticker=s.ticker )
     DATA INITIALLY DEFERRED REFRESH DEFERRED;
   ```

   Include all of the VARCHAR columns in the join predicate (nn.ticker and s.ticker) in the materialized query table output list to maximize the opportunities that the materialized query table is used by the federated database.

   To defer the refresh of the materialized query table, specify the REFRESH DEFERRED keyword. Materialized query tables that are specified with the REFRESH DEFERRED keyword do not reflect changes to the underlying base table. Use the clause DATA INITIALLY DEFERRED so that your data is not inserted into the table as part of the CREATE TABLE statement.

6. Issue a REFRESH TABLE statement to populate the table. The data in the table reflects the result of the query as a snapshot at the time that you issue the REFRESH TABLE statement. The following example populates the stockquote_MQT table and sets a value for the special register with the current refresh age.

   ```
   REFRESH TABLE stockquote_MQT;

   SET CURRENT REFRESH AGE any;
   ```

The queries that run on the data in the materialized query table are faster than the queries that run on the data in a base table. When you want to use the materialized query table, you refer to the view and not the nickname:

```
SELECT * FROM stock_quote_view
  WHERE  ticker='IBM';
```

If you issue a query to select a value that has not been cached, 0 rows are returned.

Example 2: Issuing joins using the primary and foreign keys

The PRIMARY_KEY and FOREIGN_KEY columns define relationships between the parent and child nicknames. Each parent nickname must have a primary key column option. You define the children of a parent nickname with the foreign key column option that references the primary key column of a parent nickname. A nickname can have multiple children, but a nickname can have only one parent.

Because these columns contain only binary data, the columns are defined with the FOR BIT DATA NOT NULL keywords. The DB2 Control Center generates this definition when you create the nickname. You can explicitly define the PRIMARY_KEY and FOREIGN_KEY columns as FOR BIT DATA NOT NULL when you create the nickname.

The following example shows how the Web services wrapper uses the PRIMARY_KEY and FOREIGN_KEY columns to associate parent and child nicknames.

```
CREATE NICKNAME zooport_getzooreport_nn (
   zooid VARCHAR (48) OPTIONS(TEMPLATE '&column'),
   zoo_id VARCHAR (48) OPTIONS(XPATH './ns1:Zoo/@id'),
   report_zoo_zooname VARCHAR (48)
       OPTIONS(XPATH './ns1:Zoo/ns1:ZooName/text()'),
   report_zoo_count VARCHAR (48)
       OPTIONS(XPATH './ns1:Zoo/ns1:Count/text()'),
   report_zoo_lastmodified VARCHAR (48)
       OPTIONS(XPATH './ns1:Zoo/ns1:LastModified/text()'),
   nn_pk VARCHAR (16) NOT NULL OPTIONS(PRIMARY_KEY 'YES'),
   url VARCHAR (256) OPTIONS(URLCOLUMN 'Y'),
   soapaction VARCHAR (256) OPTIONS(SOAPACTIONCOLUMN 'Y')
) FOR SERVER "mytestsrvr"
  OPTIONS(
    URL 'http://localhost:9080/MaelstromTest/services/ZooPort',
    SOAPACTION 'http://myzoo.com/getZooReport' ,
    TEMPLATE '<soapenv:Envelope>
               <soapenv:Body>
                 <zooId>&zooId[1,1]</zooId>
               </soapenv:Body>
              </soapenv:Envelope>',
    XPATH '/soapenv:Envelope/soapenv:Body' ,
    NAMESPACES ' soapenv="http://schemas.xmlsoap.org/soap/envelope/",
                ns1="http://myzoo.com" ');
```

```
CREATE NICKNAME zooport_getzooreport_report_zookeeper_nn (
  nn_fk VARCHAR (16) NOT NULL
      OPTIONS(FOREIGN_KEY 'ZOOPORT_GETZOOREPORT_NN'),
  zookeeper_id VARCHAR (48) OPTIONS(XPATH './@id'),
  report_zookeeper_name VARCHAR (48) OPTIONS(XPATH './ns1:Name/text()'),
  zookeeper_numbercages VARCHAR(48)
       OPTIONS(XPATH './ns1:NumberOfCages/text()'),
  nn_pk VARCHAR (16) NOT NULL OPTIONS(PRIMARY_KEY 'YES')
 )
FOR SERVER "MYTESTSRVR" OPTIONS(
  XPATH './ns1:Zoo/ns1:Zookeeper' ,
  NAMESPACES ' soapenv="http://schemas.xmlsoap.org/soap/envelope/",
     ns1="http://myzoo.com" ');
```

The foreign key, nn_fk, in nickname zooport_getzooreport_report_zookeeper_nn, refers to the parent nickname, zooport_getzooreport_nn in the foreign key option. The designated primary and foreign key nickname columns do not correspond to data in your WSDL document because these nickname columns contain keys that are generated by the wrapper. These keys identify a relationship between the parent and child nicknames that is unique only within a query. If the child nickname contains an input column, the parent nickname option template refers to that child nickname in the template structure with the nickname option.

The following SQL statement joins the parent and child nicknames:

```
SELECT *
FROM   zooport_getzooreport_nn a,
       zooport_getzooreport_report_zookeeper_nn z,
WHERE   a.nn_pk  = z.nn_fk
    AND a.zooid  = 100
    ;
```

The following description explains how the Web services wrapper uses the TEMPLATE and XPATH nickname and column options during query execution. It is not intended as an example of a specific implementation.

When you join the primary and foreign key columns, the Web services wrapper sends a message to the Web services provider, and a set of rows is returned from the Web services provider. The wrapper generates a message for the parent nickname by substituting the values of the input column (a.zooid = 100) from the query for the reference in the column option template (ZOOID VARCHAR (48) OPTIONS(TEMPLATE ’&column’)), and then all of the column references in the nickname template option (<zooId>&zooId[1,1]</zooId>). The nickname template option can include column references or child nickname references. The message is then sent to the Web service.

The wrapper generates the rows for a nickname by applying the nickname option XPATH on the document that the Web service returns. If the nickname option XPATH returns multiple XML fragments, then the nickname contains multiple rows. The column XPATH option is applied on the resulting XML fragments that represent the rows to get the column values. If a nickname has one or more indirect parents, all of the parent nickname XPATH expressions are applied in the order from the top of the hierarchy down before the nickname option XPATH and the column option XPATH are applied for this nickname.

Chapter 21. XML data sources

Configuring access to XML data sources

You can integrate the data that is in XML data sources with information from other sources by using a federated system.

Procedure

To configure a federated server to access XML data sources, you must provide the federated server with information about the data sources and objects that you want to access. After you configure the federated server, you can create queries to access the XML data sources.

XML wrapper

The Extensible Markup Language (XML) is a universal format for structured documents and data. XML uses tags for structuring the data in documents.

XML files use the `.xml` file extension. Like HTML, XML uses words that are enclosed in angle brackets (< >) as tags. The tags structure the data that is in the document.

The following document is a sample XML document.

```
<?xml version="1.0" encoding=UTF-8"?>
<doc>
   <customer id='123'>
      <name>...</name>
      <address>...</address>
       ...
      <order>
         <amount>...</amount>
            <date>...</date>
         <item quant='12'>
            <name>...</name>
         </item>
         <item quant='4'>...</item>
          ...
      </order>
      <order>...</order>
       ...
      <payment>
         <number>...</number>
         <date>...</date>
      </payment>
      <payment>>...</payment>
       ...
   </customer>
   <customer id='124'>...</customer>
</doc>
```

Figure 33. Sample XML document

How the XML wrapper works

The XML wrapper enables you to use SQL to query the following types of data:

- External XML documents that are stored in a single file
- Multiple XML files in a directory path
- Remote XML files that are referenced with a Uniform Reference Identifier (URI)
- XML documents stored in relational columns

The following figure shows how the XML wrapper works with your federated system.

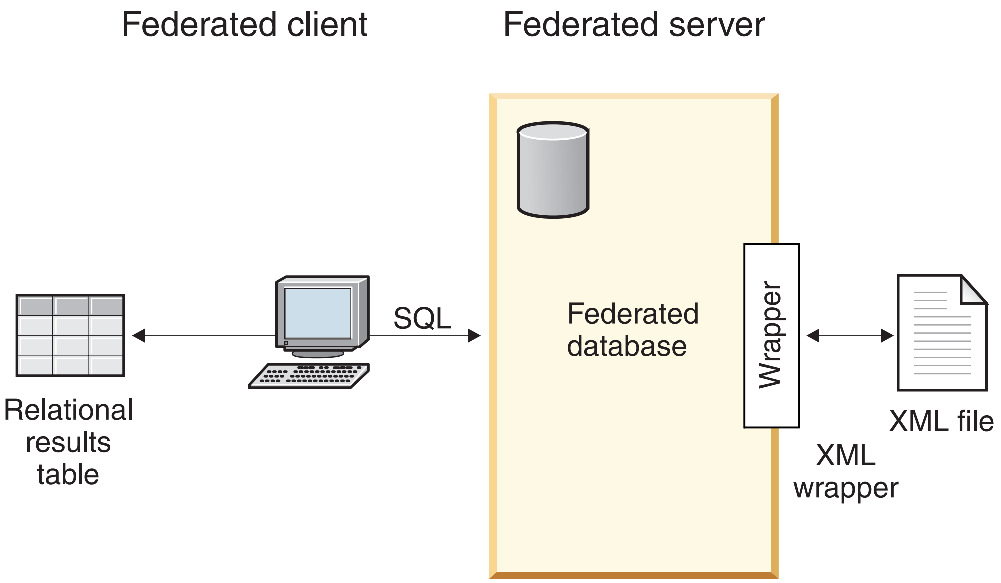


Figure 34. How the XML wrapper works

With the XML wrapper, you can map XML data from an external data source into a relational schema that is composed of a set of nicknames. The structure of an XML document is logically equivalent to a relational schema in which the nested and repeating elements are modeled as separate tables with foreign keys.

The nicknames that correspond to an XML document are organized into a tree structure in which the child nicknames map to elements that are nested within the element that corresponds to the parent nickname.

A root nickname is a nickname at the top-level of a nickname hierarchy. A nonroot nickname is a nickname that has a parent nickname in a nickname hierarchy. You can have root nicknames that are not the top level element in an XML document.

When nested elements are repeated or have distinct identities with complex structures, you can provide separate nicknames for each nested element.

Child and parent nicknames are connected by primary and foreign keys that are generated by the wrapper.

XPath expressions are used to map an XML document into a relational schema that is composed of a set of nicknames. XPath is an addressing mechanism for identifying the parts of an XML file (for example, the groups of nodes and attributes within an XML document tree). The basic XPath syntax is similar to file system addressing.

Each nickname is defined by an XPath expression that identifies the XML elements representing individual tuples, and a set of XPath expressions that specifies how to extract the column values from each element.

An example of XML document mapping

The following example illustrates how:

- The sample XML document is mapped into a set of nicknames
- Parent and child relationships are established by using primary and foreign keys
- XPath expressions are used to define individual tuples and columns within each element of the document
- A query can run on the XML document after the document is registered to your federated system

The sample XML document contains a set of customer elements. Each element encloses several order and payment elements.

The order elements enclose several item elements.

The relationship among the elements is shown in the following figure.

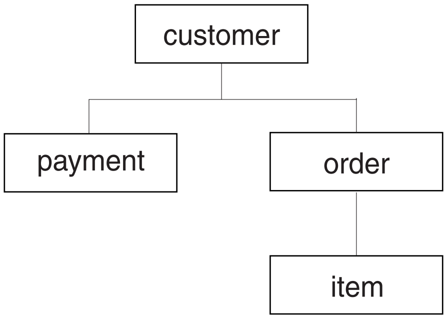


Figure 35. Tree structure of the sample XML document

From this structure, you can use the CREATE NICKNAME statement to map the XML document into a relational schema that includes four nicknames:

- customers
- orders
- payments
- items

You define relationships between the nicknames by specifying each nickname as a parent nickname or a child nickname by using special nickname column options that specify primary and foreign keyss. Each parent nickname must have a special column that is designated with a primary key column option. You define the children of a parent nickname with the foreign key column option that references the primary key column of a parent nickname. The designated primary and foreign nickname columns do not correspond to data in your XML document because these nickname columns will contain keys that are generated by the wrapper. A nickname can have multiple children, but a nickname can have only one parent. The root nickname has no parent.

For the sample XML document, the customers nickname has a defined primary key, and the orders, payments, and items nicknames have defined foreign keys that point to the parent nickname. The foreign keys of the orders and payments nicknames point to the customers nickname, and the foreign key of the items nickname points to the orders nickname.

To identify the XML elements that represent individual tuples, you create one XPath expression. In this example, all the customer elements are referenced by using the `'/doc/customer'` XPath expression, and all the order elements are referenced by using the `'./order'` XPath expression. The period in the `'./order'` XPath expression indicates that the tuples of each order element are nested within the tuples of the corresponding customer element.

You create a set of XPath expressions to specify how to extract the column values from each element. In this example, the `id` attribute of the customer elements, now a column defined in the nickname, is referenced by using the `'./@id'` XPath expression. The name element of the customer elements is referenced by using the `'./name'` XPath expression.The address element of the customer elements is referenced by using the `'./address/@street'` XPath expression.

After you map the XML document into a set of nicknames by using the CREATE NICKNAME statement, you define each nickname as a parent or child by using primary and foreign keys. You specify XPath expressions on these primary and foreign keys that define individual tuples and columns within each element of the document. You can then run SQL queries on the XML document.

Adding XML to a federated system

To configure a federated server to access XML data sources, you must provide the federated server with information about the data sources and objects that you want to access.

Before you begin

- IBM WebSphere Federation Server must be installed on a server that will act as the federated server
- A federated database must exist on the federated server

About this task

You can configure a federated server to access data that is stored in XML data sources by using the Control Center or by issuing SQL statements on the command line. The Control Center includes a wizard to guide you through the steps that are necessary to configure the required federated objects.

Procedure

To add XML data sources to a federated server:

1. Register the XML wrapper.
2. Register the XML server definition.
3. Register nicknames for the XML data sources.

4. Create federated views for non-root nicknames.

Registering the XML wrapper

You must register a wrapper to access XML data sources.

About this task

Wrappers are used by federated servers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.

You can register a wrapper by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps that are necessary to register the wrapper.

If you use a proxy server and a keystore to access XML files, you can specify the proxy server information as options when you register the wrapper or server definition. If you specify the proxy server information when you register the server definition, those settings override the proxy server settings that you specify when you register the wrapper.

Procedure

To register the XML wrapper:

Choose the method that you want to use to register the XML wrapper:

Method	Procedure
Using the Control Center	Start the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**.

From the command line	Issue the CREATE WRAPPER statement.
	`CREATE WRAPPER wrapper_name`
	`LIBRARY library_name;`
	If you use a proxy server or a keystore to access XML files, you must specify several options when you register either the XML wrapper or the server definition. To specify the proxy server information when you register the XML wrapper, the statement that you issue is:
	`CREATE WRAPPER wrapper_name`
	`LIBRARY library_name`
	`OPTIONS (PROXY_TYPE 'type',`
	`PROXY_SERVER_NAME 'server_name',`
	`PROXY_SERVER_PORT 'port_number');`

You must specify the LIBRARY parameter in the CREATE WRAPPER statement. The name of the wrapper library file that you specify depends on the operating system of the federated server. See the list of XML wrapper library files for the correct library name to specify in the CREATE WRAPPER statement.

If you use a proxy server or a keystore to access XML files, you must specify several options when you register either the XML wrapper or the server definition.

XML wrapper library files

The XML wrapper library files are added to the federated server when you install IBM WebSphere Federation Server.

When you install IBM WebSphere Federation Server, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2lsxml.a, libdb2lsxmlF.a, and libdb2lsxmlU.a. The default wrapper library file is libdb2lsxml.a. The other wrapper library files are used with specific wrapper options.

You must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

The default directory paths and default wrapper library file names are listed in the following table.

Table 86. XML wrapper library locations and file names

Operating system	Directory path	Wrapper library file name
AIX	/usr/opt/<install_path>/lib32/ /usr/opt/<install_path>/lib64/	libdb2lsxml.a
Linux	/opt/IBM/db2/<install_path>/lib32 /opt/IBM/db2/<install_path>/lib64	libdb2lsxml.so
Solaris	/opt/IBM/db2/<install_path>/lib32 /opt/IBM/db2/<install_path>/lib64	libdb2lsxml.so
Windows	%DB2PATH%\bin	db2lsxml.dll

`<install_path>` is the directory path where IBM WebSphere Federation Server is installed on UNIX or Linux.

%DB2PATH% is the environment variable that is used to specify the directory path where IBM WebSphere Federation Server is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

CREATE WRAPPER statement - examples for the XML wrapper

Use the CREATE WRAPPER statement to register the XML wrapper. The examples show the parameters that are required to access XML documents with and without a proxy server.

Registering a wrapper

If you are not using a proxy server to access XML documents, the statement that you issue to register the wrapper is:

```
CREATE WRAPPER xml_wrapper LIBRARY 'libdb2lsxml.a';
```

xml_wrapper
A name that you assign to the XML wrapper. Duplicate wrapper names are not allowed.

LIBRARY *'libdb2lsxml.a'*
The name of the wrapper library file for federated servers that use AIX operating systems.

Registering a wrapper for an HTTP proxy server

To register a wrapper and specify an HTTP proxy server, use the following statement:

```
CREATE WRAPPER xml_wrapper LIBRARY 'libdb2lsxml.a'
    OPTIONS (PROXY_TYPE 'HTTP', PROXY_SERVER_NAME 'proxy_http',
        PROXY_SERVER_PORT '8080');
```

PROXY_TYPE *'HTTP'*
: Specifies the proxy type that is used to access the Internet when behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy_http'*
: Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'8080'*
: Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Registering a wrapper for a SOCKS proxy server

To register a wrapper and specify a SOCKS proxy server without authentication information, use the following statement:

```
CREATE WRAPPER xml_wrapper LIBRARY 'libdb2lsxml.so'
    OPTIONS (PROXY_TYPE 'SOCKS', PROXY_SERVER_NAME 'proxy_socks',
        PROXY_SERVER_PORT '1081');
```

LIBRARY *'libdb2lsxml.so'*
: The name of the wrapper library file for federated servers that use Linux and Solaris operating systems.

PROXY_TYPE *'SOCKS'*
: Specifies the proxy type that is used to access the Internet when behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy_socks'*
: Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'1081'*
: Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Registering the server definition for an XML data source

You must register a server definition to retrieve XML documents. The server definition that you register depends on if you use a proxy server to access the XML documents.

About this task

To retrieve XML documents that are behind a firewall by using a proxy and a Uniform Resource Identifier (URI), you must include the proxy server options in the CREATE SERVER statement.

If you are not using a proxy server, you must still register a server definition because the hierarchy of federated objects requires that the XML files are associated with a specific server definition object.

You can register a server definition by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps to register the server definition.

Procedure

To register a server definition for an XML data source:

Choose the method that you want to use to register the server definition:

Method	Procedure
Using the Control Center	Use the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**. Follow the steps in the wizard.

From the command line	Issue the CREATE SERVER statement.
	If you do not use a proxy server to access XML documents, the statement that you issue is:
	`CREATE SERVER server_definition_name WRAPPER wrapper_name;`
	If you use a proxy server to access XML documents, the statement that you issue is:
	`CREATE SERVER server_definition_name WRAPPER wrapper_name OPTIONS (PROXY_TYPE 'type', PROXY_SERVER_NAME 'server_name', PROXY_SERVER_PORT 'port_number');`

If you use a proxy server to access XML documents, the protocol that you use might require that you specify an authorization ID and a password for the proxy server. You specify the authentication information as options when you register the server definition.

CREATE SERVER statement - examples for the XML wrapper

Use the CREATE SERVER statement to register server definitions for the XML wrapper. The examples show the parameters that are required to access XML documents with and without a proxy server.

Registering a server definition

Even if you are not using a proxy server to access XML documents, you must still register a server definition. The hierarchy of federated objects requires that the XML files are associated with a specific server definition object. The statement that you issue to register a server definition is:

```
CREATE SERVER xml_server WRAPPER xml_wrapper;
```

xml_server
A name that you assign to the XML server definition. Duplicate server definition names are not allowed.

WRAPPER *xml_wrapper*
The wrapper name that you specified in the CREATE WRAPPER statement.

Server definitions when a proxy server is used

You must use the proxy server options in the CREATE SERVER statement if all of the following conditions are true:

- You want to retrieve data using a URI
- The URI used will retrieve data from behind a firewall, through a proxy
- The firewall or proxy used is HTTP or SOCKS

The options that you specify depend on the type of proxy server that you want to access.

Check with your network administrator for information about the type of proxy that you use, and the settings that you should specify in the proxy options.

Registering a server definition for an HTTP proxy server

To register a server definition and specify an HTTP proxy server, use the following statement:

```
CREATE SERVER xml_server_http
    WRAPPER xml_wrapper
    OPTIONS (PROXY_TYPE 'HTTP', PROXY_SERVER_NAME 'proxy_http',
        PROXY_SERVER_PORT '8080');
```

xml_server_http
A name that you assign to the XML server definition. Duplicate server definition names are not allowed.

WRAPPER *xml_wrapper*
The wrapper name that you specified in the CREATE WRAPPER statement.

PROXY_TYPE *'HTTP'*
Specifies the proxy type that is used to access the Internet when behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy_http'*
Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'8080'*
Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Registering a server definition for a SOCKS proxy server

To register a server definition and specify a SOCKS proxy server when authentication information is not required, use the following statement:

```
CREATE SERVER xml_server_socks
    WRAPPER xml_wrapper
    OPTIONS (PROXY_TYPE 'SOCKS', PROXY_SERVER_NAME 'proxy_socks',
        PROXY_SERVER_PORT '1081');
```

xml_server_socks
A name that you assign to the XML server definition. Duplicate server definition names are not allowed.

WRAPPER *xml_wrapper*
The wrapper name that you specified in the CREATE WRAPPER statement.

PROXY_TYPE *'SOCKS'*
: Specifies the proxy type that is used to access the Internet when behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.

PROXY_SERVER_NAME *'proxy_socks'*
: Specifies the proxy server name or IP address. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

PROXY_SERVER_PORT *'1080'*
: Specifies the proxy server port number. This field is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Registering a server definition for a SOCKS proxy server with authentication information

To register a server definition and specify a SOCKS proxy server with authentication information, use the following statement:

```
CREATE SERVER xml_server_socks
    WRAPPER xml_wrapper
    OPTIONS (PROXY_TYPE 'SOCKS', PROXY_SERVER_NAME 'proxy_socks',
        PROXY_SERVER_PORT '1081', PROXY_AUTHID 'Martin',
        PROXY_PASSWORD 'not4me');
```

PROXY_AUTHID *'Martin'*
: Specifies the user name on the proxy server. This server option is required when the value for the PROXY_TYPE server option is 'SOCKS'.

PROXY_PASSWORD *'not4me'*
: Specifies the password on the proxy server that is associated with the user name *'Martin'*. This server option is required when the value for the PROXY_TYPE server option is 'SOCKS'.

Specifying a time limit on proxy server responses

In addition to the required server options for proxy servers, you can also specify the SOCKET_TIMEOUT server option when you register a server definition. The SOCKET_TIMEOUT server option specifies the maximum amount of time, in minutes, that the federated server will wait for the results from the proxy server. If you do not specify the SOCKET_TIMEOUT server option, there is no time limit. The federated server will wait indefinitely for the results from the proxy server.

To register a server definition and specify how long the federated server waits for a response from the proxy server, use the following statement:

```
CREATE SERVER xml_server_http
    WRAPPER xml_wrapper
    OPTIONS (PROXY_TYPE 'HTTP', PROXY_SERVER_NAME 'proxy_http',
        PROXY_SERVER_PORT '8080', SOCKET_TIMEOUT '12');
```

SOCKET_TIMEOUT *12*
: Specifies that the federated server will wait 12 minutes for a response from the proxy server.

Access XML files using a proxy server

If your network uses a proxy server, you must specify information about the proxy server when you register the wrapper or server definition for XML data sources.

The options that you specify depend on the type of proxy server that you want to access, and whether you are using Secure Socket Layer (SSL) or Transport Layer Security (TLS) protocols.

You can specify the proxy and SSL options when you register a wrapper or a server definition:

- If you specify these options when you register the wrapper, the nicknames that are associated with that wrapper will use the options that are set for the wrapper.
- If you specify these options when you register a server definition, the nicknames that are associated with that server definition will use the options that are set for the server definition.
- If you specify different values for these options when you register the wrapper and the server definition, the values that are set for the server definition take precedence over the values that are set for the wrapper.

The XML wrapper has a validation feature that might have limitations when it is used with the proxy server. The conditions in which you will see this limitation are:

- You are using the proxy feature, at the server level, and you have set the various proxy options
- The XML instance document contains a reference to an external XML schema that is located outside the firewall

If you have one of these conditions, try to change the location of your XML schema to a location inside the firewall. If you change the XML schema location, you must update the XML instance document with the new location of the XML schema.

Proxy server options

The following table lists the options that must you specify for each type of proxy server:

Table 87. Options that are required with proxy servers

Type of proxy server	Required wrapper or server options
HTTP or SOCKS without authentication	PROXY_TYPE PROXY_SERVER_NAME PROXY_SERVER_PORT
HTTP or SOCKS with authentication	PROXY_TYPE PROXY_SERVER_NAME PROXY_SERVER_PORT PROXY_AUTHID PROXY_PASSWORD

SSL server options

The following table lists the options that you must specify when you use the SSL protocols:

Table 88. Options that are required with SSL protocols

Federated object	Required options
Wrapper	SSL_KEYSTORE_FILE SSL_KEYSTORE_PASSWORD SSL_VERIFY_SERVER_CERTIFICATE
Server definition	SSL_KEYSTORE_FILE SSL_KEYSTORE_PASSWORD SSL_VERIFY_SERVER_CERTIFICATE SSL_CLIENT_CERTIFICATE_LABEL

Nicknames for XML data sources

You must register a nickname for each XML document that you want to access. Use these nicknames, instead of the document names, when you query XML data sources.

You can register nicknames for XML data sources by using the Control Center or the command line. The Control Center simplifies the process for creating the XML nicknames. Before you register the nicknames, you should understand:

Data associations between nicknames and XML documents

Nicknames correspond to the tree structure of your XML document data. Parent nicknames and child nicknames correspond to the root structure and nested elements of the data tree structure. These parent and child nicknames are connected by primary and foreign keys that are specified with the CREATE NICKNAME statement.

Each nickname is defined by XPath expressions that perform the following functions:

- Identifies the XML elements that represent individual tuples
- Specifies how to extract the column values from each element

The XML wrapper uses XPath expressions to establish a correspondence between the data in the XML document and the rows in a relational table. These XPath expressions identify the values within the XML document and determine how these values correspond to the columns of each row. The XML wrapper reads the XML document data only. The XML wrapper does not update this data.

When you create a nickname, you choose options that specify the association between the nickname and the XML document. Nicknames are associated with your XML documents either in a fixed manner or with source names that you specify.

With a fixed association, the nickname represents data from specific XML documents. These XML documents include:

One local file
You specify one XML file as your XML document.

Multiple local files in a directory path
You specify a directory path in which multiple XML files reside. The XML files in this directory path provide the XML document data to the nickname. All of the XML files must have the same configuration. If any

XML file in the directory has a configuration that is different from the configuration of the nickname, the XML wrapper returns null values when it processes that XML data file. The directory must be either local to the federated server or accessible from a shared file system.

When scanning the directory, the XML wrapper retains and parses only those files with a .xml extension. The XML wrapper ignores all other files, including files with a .txt extension, files with a .xsd extension, and files without extensions.

Use the FILE_PATH option in the CREATE NICKNAME statement to specify fixed data from a file. Use the DIRECTORY_PATH option to specify fixed data from a directory.

When the source data is specified while the query is running, you can use the nickname to represent data from any XML document source whose schema matches the nickname definition. These XML documents include:

Uniform Reference Identifiers
A remote XML file that a URI refers to supplies the XML document data to the nickname. Specify this document source by using the DOCUMENT ’URI’ nickname column option.

Relational columns
Columns from a relational table, view, or nickname are used as input to your XML document. Specify this document source by using the DOCUMENT ’COLUMN’ nickname column option.

File A single file that contains XML data is supplied as input while the query runs. Specify this document source by using the DOCUMENT ’FILE’ nickname column option.

Directory
Multiple XML files under a specified directory path supply the data while the query runs. Specify this document source by using the DOCUMENT ’DIRECTORY’ nickname column option.

You specify the DOCUMENT column option to indicate that the source data is supplied at query time. Specify either URI, COLUMN, FILE, or DIRECTORY with the DOCUMENT column to indicate the type of XML document source.

You cannot specify a FILE_PATH option or a DIRECTORY_PATH option with a DOCUMENT column option.

Use the STREAMING option to separate the XML document data into fragments. You can use the STREAMING option with data that is in a fixed format or data that is from source names that you specify when you run a query. The XML wrapper processes the resulting stream of XML data and extracts the information that is requested by a query fragment. The XML wrapper parses one fragment at a time. Because fragments are parsed one at a time, total memory use decreases. The processing time required to run the entire query increases depending on the memory capacity of your server. Therefore, use the STREAMING option to only parse large XML documents that are 50 megabytes or more.

You can also choose nickname option values that help you optimize queries that retrieve large amounts of XML data or data that contains multiple nested elements. These options include:

- INSTANCE_PARSE_TIME

- XPATH_EVAL_TIME
- NEXT_TIME

You can set values for these options to test and optimize the XML query. These option values control the processing time that is needed to locate elements and to parse the data in the rows of the XML document.

The cost model facility for the XML wrapper

The XML wrapper provides a cost model facility to optimize queries on nicknames that correspond to your XML source documents.

When you create a nickname by using the CREATE NICKNAME statement, you can specify the following nickname options to support the cost model facility:

- INSTANCE_PARSE_TIME
- XPATH_EVAL_TIME

You can use the default values for these nickname options. Or you can set the values for these nickname options to optimize queries on the root and nonroot nicknames that you create.

The INSTANCE_PARSE_TIME nickname option is the amount of time, in milliseconds, that is required to read and parse one row-producing root element of the root nickname. The parsing time includes all contained row-producing nonroot elements. For example if the root nickname is customers, the nonroot elements are all of the elements that correspond to the orders, payments, and items of each customer. The XML wrapper builds a structure in memory to represent these row-producing root and nonroot elements.

The XPATH_EVAL_TIME nickname option is the amount of time, in milliseconds, that is required to evaluate the XPath expressions that locate the data corresponding to a row of the nickname. The XPath expressions that are evaluated include the XPath expressions that locate the actual rows and the XPath expressions that locate column values within these rows.

Namespaces for XML data sources

Use the NAMESPACES nickname column option to identify the elements or attributes that are part of a namespace.

You can specify the NAMESPACES nickname option when you register nicknames. The value of the NAMESPACES nickname column option is a comma-separated list of name and value pairs. The XML wrapper uses the name and value pairs to resolve the namespace prefixes that are in the nickname XPath expressions. The prefixes that are used in the XPath expressions are processed by the XPath processor.

In the following example, the XML document includes the name, code, and description information for three products. The XML document declares two namespaces, `http://www.one.com` and `http://www.two.com`, and has one default namespace `http://www.default.com`. The `product` element is associated with the `ns1` namespace. The `product` element contains the `name` and `code` attributes and the `desc` element. The `name` attribute is not associated with a namespace. The `code` attribute is associated with the `ns2` namespace. The `desc` element is associated with the `default` namespace.

```
<?xml version="1.0" encoding="UTF-8"?>
<doc xmlns:ns1="http://www.one.com" xmlns:ns2="http://www.two.com"
    xmlns="http://www.default.com">
<ns1:product name="Computer" ns2:code="ABC123"
    <desc>"The Computer product description"</desc>
<ns1:product name="Keyboard" ns2:code="EFG456"
    <desc>"The Keyboard product description"</desc>
<ns1:product name="Mouse" ns2:code="HIJ789"
    <desc>"The Mouse product description"</desc>
</ns1:product>
</doc>
```

The following table shows the namespace that is associated with each element and attribute in the XML document.

Table 89. The elements, attributes, and namespaces in the XML document

Element or attribute	Namespace in XML document	Notes
product: An element in the XML document.	ns1=″http://www.one.com″	None
name: An attribute of the product element in the XML document.	None	The name attribute is not associated with a namespace.
code: An attribute of the product element in the XML document.	ns2=″http://www.two.com″	None
desc: An element within the product element in the XML document.	″http://www.default.com″	The desc element uses the default namespace, which does not contain a prefix.

When you register the nickname for the XML document, you define three columns to correspond to the elements and attributes in the XML document. You specify the namespace information in the NAMESPACES nickname option. For example:

```
CREATE NICKNAME products
   (name VARCHAR(16) OPTIONS (XPATH '@name'),
    code VARCHAR(16) OPTIONS (XPATH '@pre2:code')
    description VARCHAR (256) OPTIONS (XPATH './default:desc'))
    FOR SERVER xml_server
    OPTIONS (FILE_PATH '/home/mbreining/sql/xml/namespaces.xml',
        XPATH '//pre1:name',
        NAMESPACES 'pre1="http://www.one.com", pre2="http://www.two.com",
           default="http=//www.default.com"');
```

The following table shows how the XML document namespaces correspond to the values that you specify in the CREATE NICKNAME statement.

Table 90. XML document namespaces and corresponding values from the CREATE NICKNAME statement.

Namespace in XML document	Column name in the CREATE NICKNAME statement	Value in the XPATH column option	Value in the NAMESPACES nickname option
The namespace is ns1="http://www.one.com".	None	None	The value is pre1="http://www.one.com". The value is a prefix that you define (pre1) and the unique identifier for the namespace (″http://www.one.com″).

Table 90. XML document namespaces and corresponding values from the CREATE NICKNAME statement. (continued)

Namespace in XML document	Column name in the CREATE NICKNAME statement	Value in the XPATH column option	Value in the NAMESPACES nickname option
None. The attribute is not associated with a namespace.	Name	The value is `@name`. The value is an attribute that is in the XML document (name).	None
The namespace is `ns2="http://www.two.com"`	Code	The value is `@pre2:code`. The value is a prefix that you define (pre2) and an attribute that is in the XML document (code).	The value is `pre2="http://www.two.com"`. The value is a prefix that you define (pre2) and the unique identifier for the namespace (″http://www.two.com″).
The namespace is `"http://www.default.com"` The default namespace does not contain a prefix.	Description	The value is `./default:desc`. The value is a prefix that you define (default) and an element that is in the XML document (desc).	The value is `default="http=//www.default.com"`. The value is a prefix that you define (default) and the unique identifier for the namespace (″http://www.default.com″).

The NAMESPACES nickname option uses packed descriptors to support strings that are greater than 256 characters.

For more information about XML namespaces, see the explanation for namespaces on the W3C Web site.

Registering nicknames for XML data sources

For each XML server definition that you register, you must register a nickname for each XML document that you want to access. Use these nicknames, instead of the XML document names, when you query the XML data sources.

Restrictions

If you attempt to access XML data sources that are on a shared drive from a federated server that runs Windows 2003, your queries might fail. This is a limitation of Windows 2003. You can avoid this problem by specifying the absolute path in the FILE_PATH option in the CREATE NICKNAME statement.

About this task

You must create nicknames that correspond to the tree structure of your XML data source. Parent nicknames correspond to the root structure of the tree. Child nicknames correspond to the elements that are nested within the element for the parent nickname.

You can register nicknames by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps to register the server definition.

Procedure

To register nicknames for XML data sources:

Choose the method that you want to use to register the server definition:

Method	Procedure
Using the Control Center	Use the Federated Objects wizard. Right-click the **Federated Database Objects** folder and click **Create Federated Objects**. Follow the steps in the wizard.
From the command line	Issue the CREATE NICKNAME statement. Nicknames can be up to 128 characters in length.

CREATE NICKNAME statement - examples for XML wrapper

Use the CREATE NICKNAME statement to register nicknames for XML documents. There is a complete example, which shows how to create parent and child nicknames, and examples for specific column options.

Recommendation: Do not use the self or descendant operator // when you specify XPATH columns and nickname options in your queries. The self or descendant operator is an XPath operator. Using self or descendant operator can decrease federated server performance.

Complete example

The following example shows how to create nicknames for XML data sources by using the sample XML file shown in the following sample XML file.

```
<?xml version="1.0" encoding="UTF-8"?>
<doc>
   <customer id='123'>
      <name>...</name>
      <address>...</address>
       ...
      <order>
         <amount>...</amount>
            <date>...</date>
         <item quant='12'>
            <name>...</name>
         </item>
         <item quant='4'>...</item>
          ...
      </order>
      <order>...</order>
       ...
      <payment>
         <number>...</number>
         <date>...</date>
      </payment>
      <payment>>...</payment>
       ...
   </customer>
   <customer id='124'>...</customer>
</doc>
```

Figure 36. Sample XML file

The parent nickname

The first step is to create the parent nickname, customers. To create the nickname, issue the following statement:

```
CREATE NICKNAME customers
(
   id        VARCHAR(5)    OPTIONS(XPATH './@id')
   name      VARCHAR(16)   OPTIONS(XPATH './name'),
   address   VARCHAR(30)   OPTIONS(XPATH './address/@street'),
   cid       VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(PRIMARY_KEY 'YES'))
   FOR SERVER xml_server
   OPTIONS(DIRECTORY_PATH '/home/db2user',
         XPATH '/doc/customer', STREAMING 'YES');
```

This statement creates the customers nickname over multiple XML files under the specified directory path, /home/db2user.

The STREAMING nickname option indicates that the XML source data is separated and processed by node, in this example by customer record. When the STREAMING nickname option is used, the wrapper does not storing the entire XML document into memory. Instead, the XML wrapper divides the document into multiple sections which are parsed individually and sequentially. The STREAMING nickname option should be used only with large XML documents. The performance of your queries is impacted when you use this option.

The child nicknames

The next step is to create the child nicknames for the orders, payments, and items elements.

Issue the following statement to create the orders child nickname:

```
CREATE NICKNAME orders
(
   amount INTEGER       OPTIONS(XPATH './amount'),
   date   VARCHAR(10)   OPTIONS(XPATH './date'),
   oid    VARCHAR(16)   OPTIONS(PRIMARY_KEY 'YES'),
   cid    VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(FOREIGN_KEY 'CUSTOMERS'))
   FOR SERVER xml_server
   OPTIONS( XPATH './order');
```

Issue the following statement to create the payments child nickname:

```
CREATE NICKNAME payments
(
   number INTEGER       OPTIONS(XPATH './number'),
   date   VARCHAR(10)   OPTIONS(XPATH './date'),
   cid    VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(FOREIGN_KEY 'CUSTOMERS'))
   FOR SERVER xml_server
   OPTIONS( XPATH './payment');
```

Issue the following statement to create the items child nickname:

```
CREATE NICKNAME items
(
   name      VARCHAR(20)   OPTIONS(XPATH './name'),
   quantity  INTEGER       OPTIONS(XPATH './@quant'),
   oid       VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(FOREIGN_KEY 'ORDERS'))
   FOR SERVER xml_server
   OPTIONS( XPATH './item');
```

Nickname column option examples

The following examples show you how to include the DOCUMENT nickname column options when you create nicknames. The examples also show you how those options are used in queries.

DOCUMENT *'FILE'* example

The following examples show you how to include the DOCUMENT nickname column options when you create nicknames. The examples also show you how to use those options in queries.

The following CREATE NICKNAME example shows the use of the DOCUMENT 'FILE' nickname column option:

```
CREATE NICKNAME customers
(
   doc      VARCHAR(100)   OPTIONS(DOCUMENT 'FILE'),
   name     VARCHAR(16)    OPTIONS(XPATH './name'),
   address  VARCHAR(30)    OPTIONS(XPATH './address/@street'),
   cid      VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(PRIMARY_KEY 'YES'))
   FOR SERVER xml_server
   OPTIONS(XPATH '/doc/customer');
```

You can then run the following query on the customers nickname, specifying the location of the XML document in the WHERE clause:

```
SELECT * FROM customers WHERE doc = '/home/db2user/Customers.xml';
```

DOCUMENT *'DIRECTORY'* example

The following CREATE NICKNAME example shows the use of the DOCUMENT 'DIRECTORY' nickname column option:

```
CREATE NICKNAME customers
(
   doc      VARCHAR(100)   OPTIONS(DOCUMENT 'DIRECTORY'),
   name     VARCHAR(16)    OPTIONS(XPATH './name'),
   address  VARCHAR(30)    OPTIONS(XPATH './address/@street'),
   cid      VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(PRIMARY_KEY 'YES'))
   FOR SERVER xml_server
   OPTIONS(XPATH '/doc/customer');
```

You can then run the following query on the customers nickname:

```
SELECT name FROM customers WHERE doc = '/home/data/xml';
```

This query retrieves the XML documents that are located under the directory path /home/data/xml, which is specified in the WHERE clause.

DOCUMENT *'URI'* example

The following CREATE NICKNAME example shows the use of the DOCUMENT 'URI' nickname column option:

```
CREATE NICKNAME customers
(
   doc      VARCHAR(100)   OPTIONS(DOCUMENT 'URI'),
   name     VARCHAR(16)    OPTIONS(XPATH './name'),
   address  VARCHAR(30)    OPTIONS(XPATH './address/@street'),
   cid      VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(PRIMARY_KEY 'YES'))
   FOR SERVER xml_server
   OPTIONS(XPATH '/doc/customer');
```

You can then run the following query on the customers nickname to retrieve the XML data from the remote location:

```
SELECT * FROM customers WHERE doc = 'http://www.lg-mv.org/foo.xml';
```

DOCUMENT *'COLUMN'* example

The following CREATE NICKNAME example shows the use of the DOCUMENT 'COLUMN' nickname column option:

```
CREATE NICKNAME emp
(
   doc      VARCHAR(500)  OPTIONS(DOCUMENT 'COLUMN')
   fname    VARCHAR(16)   OPTIONS(XPATH '@first'),
   lname    VARCHAR(16)   OPTIONS(XPATH '@last'))
   FOR SERVER xml_server
   OPTIONS(XPATH '/doc/name');
```

You can then run one of the following queries on the emp nickname to retrieve the XML data:

```
SELECT * FROM emp WHERE doc = '<?xml version="1.0" encoding="UTF-8"?>
        <doc>
        <title>  employees </title>
        <name first="David"  last="Marston"/>
        <name first="Donald" last="Leslie"/>
        <name first="Emily"  last="Farmer"/>
        <name first="Myriam" last="Midy"/>
        <name first="Lee"    last="Tran"/>
        <name first="Lili"   last="Farmer"/>
        <name first="Sanjay" last="Kumar"/>
        </doc>';
```

or

```
SELECT * FROM emp WHERE doc = (SELECT * FROM xml_tab);
```

The xml_tab table contains one column that is populated with the XML data.

Queries for XML data sources

Before you create queries to access XML data sources, there are actions that you can take to optimize the query performance.

Federated views

You can use federated views to ensure that the queries that join the pieces of an XML nickname hierarchy run correctly.

Avoid the self or descendant operator

Do not use the self or descendant operator // when you specify XPATH columns and nickname options when you create the XML nicknames. The self or descendant operator is an XPath operator, and using the self or descendant operator can decrease federated server performance.

Windows 2003 federated servers

If you attempt to access XML data sources that are on a shared drive from a federated server that runs Windows 2003, your query might fail with the following error message:

```
SQL1822N  Unexpected error code "ERRNO = 2" received from data source
"XML_SERVER". Associated text and tokens are "Unable to read file".
SQLSTATE=560BD
```

This is a limitation of Windows 2003. You can avoid this problem by specifying the absolute path in the FILE_PATH option in the CREATE NICKNAME statement.

The following example shows a CREATE NICKNAME statement with an abbreviated path specified in the FILE_PATH option:

```
CREATE NICKNAME customers
(
  id      VARCHAR(5)  OPTIONS(XPATH './@id'),
  name    VARCHAR(16) OPTIONS(XPATH './name'),
  address VARCHAR(30) OPTIONS(XPATH './address/@street'),
  cid     VARCHAR(16) FOR BIT DATA NOT NULL
     OPTIONS(PRIMARY_KEY 'YES'))
     FOR SERVER xml_server
        OPTIONS(DIRECTORY_PATH '\home\db2user',
           XPATH '/doc/customer', STREAMING 'YES');
```

Queries that use this nickname might fail because you specified the abbreviated path.

For federated servers that run Windows 2003, specify the absolute path in the FILE_PATH option in the CREATE NICKNAME statement.

For example:

```
CREATE NICKNAME customers
(
  id      VARCHAR(5)  OPTIONS(XPATH './@id'),
  name    VARCHAR(16) OPTIONS(XPATH './name'),
  address VARCHAR(30) OPTIONS(XPATH './address/@street'),
  cid     VARCHAR(16) FOR BIT DATA NOT NULL
     OPTIONS(PRIMARY_KEY 'YES'))
```

```
                FOR SERVER xml_server
                   OPTIONS(DIRECTORY_PATH '\\host.svl.ibm.com\D$\home\db2user',
                      XPATH '/doc/customer', STREAMING 'YES');
```

Creating federated views for the XML wrapper nicknames

You can create federated views over the hierarchy of nicknames that describe an XML document. Defining federated views ensures that the queries that join the pieces of an XML nickname hierarchy run properly.

About this task

A *federated view* is a view in the federated database that references a nickname, instead of a data source table.

In the XML nickname hierarchy the root nickname and queries that join columns, other than the special PRIMARY_KEY and FOREIGN_KEY columns, are not impacted by using federated views.

When you create federated views for XML nicknames, you should include all of the required predicates and a full path to the root directory.

Procedure

To create federated views for XML nicknames:

Use the CREATE VIEW statement to define a view for each nonroot nickname. The view should be a join of all the nicknames on the path to the root nickname.

1. In the WHERE clause of the view, define the PRIMARY_KEY and FOREIGN_KEY columns as the join predicates.
2. In the SELECT list, include all the columns of the nonroot nickname except the column that is designated with the FOREIGN_KEY nickname column option. In the SELECT list, include the column of the parent nickname designated with the PRIMARY_KEY option.

CREATE VIEW statement - examples for the XML wrapper

Use the CREATE VIEW statement to create federated views for nonroot nicknames. This example includes a sample XML file, the statements that you use to create the views, and shows how you can use the views in a query.

You can create federated views over the hierarchy of nicknames that describe an XML document to ensure that the queries that join pieces of an XML nickname hierarchy run correctly. When you specify a federated view in a query, data is retrieved from the remote data source.

The following examples show you how to create federated views for nonroot nicknames to describe XML source documents.

Sample XML file

The following examples are based this sample XML file.

```
<?xml version="1.0" encoding="UTF-8"?>
<doc>
   <customer id='123'>
      <name>...</name>
      <address>...</address>
       ...
      <order>
         <amount>...</amount>
            <date>...</date>
         <item quant='12'>
            <name>...</name>
         </item>
         <item quant='4'>...</item>
          ...
      </order>
      <order>...</order>
       ...
      <payment>
         <number>...</number>
         <date>...</date>
      </payment>
      <payment>...</payment>
       ...
   </customer>
   <customer id='124'>...</customer>
</doc>
```

Figure 37. Sample XML file

CREATE VIEW statements for the nonroot nicknames

The following example shows how to create a federated view for the nonroot nickname order:

```
CREATE VIEW order_view AS
   SELECT o.amount, o.date, o.oid, c.cid
   FROM customers c, orders o
   WHERE c.cid = o.cid;
```

The following example shows how to create a federated view for the nonroot nickname payment:

```
CREATE VIEW payment_view AS
   SELECT p.number, p.date, c.cid
   FROM customers c, payments p
   WHERE c.cid = p.cid;
```

The following example shows how to create a federated view for the nonroot nickname item:

```
CREATE VIEW item_view AS
   SELECT i.quantity, i.name, o.oid
   FROM customers c, orders o, items i
   WHERE c.cid = o.cid AND o.oid = i.oid;
```

A query that uses the federated views

Queries that are submitted to the federated views are processed correctly because the join path to the root directory is specified in the WHERE clause.

For example, the following query uses the customer identification number and the date that an order was placed to return the amount that was ordered and the amount of the payment due. Instead of using the nicknames in the query, the views are specified in the FROM clause.

```
SELECT o.amount, p.amount
FROM order_view o, payment_view p
WHERE p.date = o.date AND
   p.cid = o.cid;
```

Query optimization tips for the XML cost model facility

The cost model facility for the XML wrapper helps optimize queries on the nicknames that you create.

The cost model facility uses the following nickname options of the CREATE NICKNAME statement:

- INSTANCE_PARSE_TIME
- XPATH_EVAL_TIME

You can specify values for these nickname options when you issue a CREATE NICKNAME statement to register a nickname for an XML data source.

The cost model facility uses these parameter values to determine the amount of time required to parse the data in each row of an XML source document. The parameter values are also used to evaluate the XPath expression for the nickname.

You can use the default values for these nickname options. However, if you want to optimize queries on large or complex XML source structures for the nicknames that you create, use the following example as a guide.

An example of optimizing a large query

Your XML document has a relational schema with four nicknames:

- customers
- orders
- payments
- items

The root nickname is customers.

Run queries on each nickname. Run each query on a sample of the XML data that is typical for your environment.

For example:

```
SELECT * from customers;
SELECT * from orders;
SELECT * from payments;
SELECT * from items;
```

Make a note of the amount of time (in milliseconds) that is required to run each query by using the db2batch command, or an equivalent command or utility. You can use the db2batch command to obtain an output file that contains the time required to run queries. Make a note of the number of tuples that are returned.

For each nickname, use the following formulas to determine the optimal values for the INSTANCE_PARSE_TIME and XPATH_EVAL_TIME nickname options:

```
INSTANCE_PARSE_TIME = (75%  X  run time of SELECT * query) ÷ number of tuples returned
XPATH_EVAL_TIME     = (25%  X  run time of SELECT * query) ÷ number of tuples returned
```

For the root nickname (in this example, `customers`), use the calculated values for the INSTANCE_PARSE_TIME and XPATH_EVAL_TIME nickname options.

For nonroot nicknames, (in this example, `orders`, `payments`, and `items`), use only the calculated value for the XPATH_EVAL_TIME parameter. The INSTANCE_PARSE_TIME parameter value is not applicable for nonroot nicknames.

You can use these formulas as a guide for tuning your queries. The optimal values for these nickname options also depend on the complexity of your XML source documents and on the speed of the processor that you are using.

XML data source - example queries

Examples of queries using XML nicknames.

These examples use the customers, orders, and items nicknames.

A query that returns a specific value from the XML documents

When the following SELECT statement is run, the wrapper returns the names of all of the customers:

```
SELECT name FROM customers;
```

A query that returns all of the records for a specific customer

When the following SELECT statement is run, the wrapper returns all of the records in which the customer name is Chang:

```
SELECT * FROM customers
    WHERE name='Chang';
```

A query that returns specific values based on a join condition between a parent and child nickname

When the following SELECT statement is run, the wrapper returns the customer names and amounts for each order placed by each customer. You must specify the join, c.cid=o.cid, to indicate the parent-child relationship between the customers nickname and the orders nickname.

```
SELECT c.name, o.amount FROM customers c, orders o
    WHERE c.cid=o.cid;
```

A query that shows how to specify join conditions between a parent and several child nicknames

When the following SELECT statement is run, the wrapper returns the customer addresses, order amounts, and item names for each order and item of each customer. You must specify the two joins to maintain the parent-child relationships.

```
SELECT c.address, o.amount, i.name FROM customers c, orders o, items i
    WHERE c.cid=o.cid AND o.oid=i.oid;
```

Queries that show how specify an XML document in a query

The following examples show you how to write queries by using a nickname that specifies a DOCUMENT nickname column option rather than a FILE_PATH nickname option.

The CREATE NICKNAME statement that is used to create the customers nickname is:

```
CREATE NICKNAME customers
(
   doc      VARCHAR(100)   OPTIONS(DOCUMENT 'FILE'),
   name     VARCHAR(16)    OPTIONS(XPATH './name'),
   address  VARCHAR(30)    OPTIONS(XPATH './address/@street'),
   cid      VARCHAR(16)    OPTIONS(PRIMARY_KEY 'YES'))
   FOR SERVER xml_server
   OPTIONS(XPATH '/doc/customer');
```

The following query selects all of the data from the XML file Customers.xml with a path of /home/db2user/Customers.xml:

```
SELECT * FROM customers
   WHERE doc='/home/db2user/Customers.xml';
```

The following query selects the names of customers and the dates of their orders from the Customers.xml file for each order with an amount over 1000. The path /home/db2user/Customers.xml specifies the location of the Customers.xml file.

```
SELECT c.name, o.date FROM customers c, orders o
   WHERE c.doc='/home/db2user/Customers.xml' AND o.amount > 1000;
```

Chapter 22. KEGG user-defined functions

KEGG user-defined functions - overview

The Kyoto Encyclopedia of Genes and Genomes (KEGG) is a suite of databases that contain genomic information. The KEGG user-defined functions are a set of functions provided with IBM WebSphere Federation Server to access the genomic information in the KEGG databases.

The Pathway database and Sequence Similarity Database (SSDB) are the only two databases in the KEGG suite that IBM WebSphere Federation Server can access through the KEGG web services interface. The Pathway database is a collection of data about molecular interaction networks in biological processes, including metabolic pathways, regulatory pathways, and molecular complexes. The SSDB is a collection of data about protein-coding genes in the complete genomes.

The KEGG user-defined functions use version 4.1 of the KEGG API to access these databases.

Many of the KEGG methods return lists of values, such as genes or pathways. To facilitate the composition of complex operations from multiple methods, most of the KEGG user-defined function exist in both table and scalar formats. The table functions return a table of single values. The scalar functions return values as a space-delimited list that is contained in a single string.

The KEGG user-defined functions are installed on the federated server when you select **Life sciences user-defined functions** under the Nonrelational wrappers component in the IBM WebSphere Federation Server installation wizard.

You must ensure that the XML Extender client and the SOAP user-defined functions are installed on the federated server. After the KEGG user-defined functions and other prerequisite software are installed, you must register the functions.

To avoid conflicts with namespaces, all of the KEGG user-defined functions are registered in the DB2LS schema.

KEGG user-defined functions by functional category

IBM WebSphere Federation Server includes a set of KEGG user-defined functions that you can use to access data in the Pathway database and Sequence Similarity (SSDB) database.

The following table lists the user-defined functions that you can use to retrieve data from the Pathway database.

Table 91. Pathway user-defined functions

Description	Function types	Function names
Compounds by pathway	scalar table	CompoundsByPathwyS CompoundsByPathwyT
Enzymes by pathways	scalar table	EnzymesByPathwyS EnzymesByPathwyT

Table 91. Pathway user-defined functions (continued)

Description	Function types	Function names
Genes by pathways	scalar table	GenesByPathwyS GenesByPathwyT
Pathways by compound	scalar table	PathwysByCompndsS PathwysByCompndsT
Pathways by enzymes	scalar table	PathwysByEnzymesS PathwysByEnzymesT
Pathways by genes	scalar table	PathwysByGenesS PathwysByGenesT

The following table lists the user-defined functions that you can use to retrieve data from the SSDB.

Table 92. SSDB user-defined functions

Description	Function types	Function names
Best neighbors or best homologous genes for a gene	scalar table	BestNbrsByGeneS BestNbrsByGeneT
Best-best neighbors or best-best homologous genes for a gene	scalar table	BstBstNbrsByGeneS BstBstNbrsByGeneT
Reverse best neighbors for a gene	scalar table	RevBestNbrsByGeneS RevBestNbrsByGeneT
Paralogous genes for a gene	scalar table	ParalogsByGeneS ParalogsByGeneT
Genes by motifs	table	GenesByMotifsT
Motifs by gene	scalar table	MotifsByGenesS MotifsByGenesT

All of the table user-defined functions that are for the SSDB database return a fixed set of output columns, except for the GenesByMotifsT function. The GenesByMotifsT function returns the *gene_id* VARCHAR(100) and the *definition* VARCHAR(1000) for each gene.

Function arguments for the KEGG user-defined functions

The KEGG user-defined functions use a common set of arguments. The arguments for the KEGG user-defined function are described in the following list.

compound_id

The unique identifier for a compound. The format that you use for the *compound_id* argument is `cpd:`*`compound`*. `cpd` is the abbreviation for the compound database. *compound* is the compound number that is in the KEGG LIGAND composite database.

For example, the compound identifier for Methylparabanic acid is:

```
cpd:C11116
```

compound_id_list

A space-delimited list of compound identifiers.

entrylist

A comma separated list of genes identifiers. The maximum number of entries in the *entrylist* argument is 100.

For example, an entry list for the M. loti, P. abyssi, and X. fastidiosa genes is:

```
mlo:mll1558, pab:PAB1288, xfa:XF2434
```

enzyme_id
The unique identifier for an enzyme. The format that you use for the *enzyme_id* argument is `ec:`*`enzyme`*. The `ec` is the abbreviation for the enzyme database. The *enzyme* is the enzyme code that is in the enzyme database.

For example, the compound identifier for Sulfite oxidase is:

```
ec:1.8.3.1
```

enzyme_id_list
A space-delimited list of enzyme identifiers.

genes_id
A unique identifier for a gene. The format that you use for the *genes_id* argument is *`org`*`:`*`gene_name`*. The *org* is the three-letter KEGG code for an organism. The *gene_name* is the name of the gene.

For example, the genes identifier for the E. coli gene is:

```
eco:b0001
```

gene_id_list
A space-delimited list of gene identifiers.

org A three-letter KEGG code for an organism. If this argument is not specified, the entire organism list is searched.

Each organism in the KEGG databases is assigned a code. The list of organisms changes frequently. Check the current list of genomes at http://www.genome.ad.jp/kegg/kegg2.html#genes for the correct codes.

max_result
An integer that is used to specify the number of results that are returned. The *max_results* argument is used with the *start* argument to control the results that are returned. For example, if 5 is specified for *start* and 12 is specified for *max_results*, the results that are returned start with result 5 and end with result 16.

motif_id
A unique identifier for a motif. The format that you use for the *motif_id* argument is *motif_database_identifier*:*motif_name*. The *motif_database_identifier* is the 2-letter code for a motif database. The *motif_name* is the name of the motif.

The valid motif database identifiers are listed in the following table.

Table 93. Identifiers for the motif databases

Identifier	Motif database
bl	Blocks
pd	Prodom
pf	Pfam
pr	Prints
ps	Prosite

For example, the motif identifier for the DnaJ protein entry in the pfam database is:

```
pf:DnaJ
```

pathway_id
A unique identifier for a pathway. The format that you use for the *pathway_id* argument is `path:pathway_number`. The `path` is the abbreviation for the pathway database. The *pathway_number* consists of a prefix and a number. The valid prefixes are *map* and *org*. The *map* prefix indicates that you want to search for the reference pathway. The *org* prefix is a 3-letter KEGG code for an organism.

For example, the pathway identifier for the reference pathway for the citrate cycles is:

```
path:map00020
```

For example, the pathway identifier for the specific pathways for the E.coli genes is:

```
path:eco00020
```

start
An integer that is used to specify the first result that is returned. The *start* argument is used with the *max_results* argument to control the results that are returned. For example, if 5 is specified for *start* and 12 is specified for *max_results*, the results that are returned start with result 5 and end with result 16.

Registering the KEGG user-defined functions

You must enable the XML Extender client, the SOAP user-defined functions, and the KEGG user-defined functions to register and use the KEGG user-defined functions.

Before you begin

- The **Life sciences user-defined functions** under the Nonrelational Wrappers component in the IBM WebSphere Federation Server installation wizard must be installed.
- The XML Extender client in the DB2 for Linux, UNIX, and Windows installation wizard must be installed.
- If you installed a version of the KEGG user-defined functions before WebSphere Information Integrator Version 8.2 Fix Pack 9, you must disable the version 2.3 KEGG user-defined functions and enable the version 4.1 KEGG user-defined functions.

Procedure

To register the KEGG user-defined functions:

1. Run the dxxadm enable_db *database_name* command to enable the XML Extender client.
2. Run the db2enable_soap_udf -n *database_name* command to enable the SOAP user-defined functions.
3. Run the enable_KEGGFunctions command to enable the KEGG user-defined functions.
   - On federated servers that run Windows, this command is in the sqllib\bin directory
   - On federated servers that run UNIX, this command is in the sqllib/bin directory

Syntax:

```
enable_KEGGFunctions -n dbName -u userID -p password [-force] [-url endpointURL]
```

-n *dbName*
The name of the federated database that you are registering the functions in.

-u *userID*
A valid user ID for the federated database.

-p *password*
A valid password for the user ID.

[-force]
An optional flag that you can use to remove the functions and register them again. Use this flag if the functions get corrupted or dropped accidentally.

[-url *endpointURL***]**
An optional flag that specifies the endpoint URL for the KEGG Web services API. The current API is Version 4.1. The default URL is set to http://soap.genome.jp/keggapi/request_v4.1.cgi. Use the *endpointURL* flag if the KEGG Web services API changes the name or location of the default URL.

The enable_KEGGFunctions command registers all of the KEGG user-defined functions in the federated database. The functions are registered with the schema name DB2LS.

Pathway database functions

With the KEGG user-defined functions you can search the pathway database by molecular complexes or by pathways.

For example, you can search for:

- Compounds, enzymes, and genes by pathway
- Pathways by compounds, enzymes, and genes

CompoundsByPathwyS user-defined function

Use the CompoundsByPathwyS function to search for all of the compounds on a pathway.

The CompoundsByPathwyS function is a scalar function that returns a space-delimited list of compound identifiers. The list is a character string with a data type of VARCHAR and an actual length that is not greater than 32767 bytes.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.CompoundsByPathwyS(pathway_id)
```

The schema name that you use with this user-defined function is DB2LS.

Example of searching for all of the compounds on a specific pathway

You want to search for all of the compound identifiers on the reference pathway map 00020.

You can specify either one of the following clauses. The first clause searches for the reference pathway. The second clause searches for a specific pathways for the E.coli genes.

```
VALUES CAST(DB2LS.CompoundsByPathwyS
    ('path:map00020')
    AS VARCHAR(1000));
```

```
VALUES CAST(DB2LS.CompoundsByPathwyS
    ('path:eco00020')
    AS VARCHAR(1000));
```

CompoundsByPathwyT user-defined function

Use the CompoundsByPathwyT function to search for all of the compound identifiers on a pathway.

The CompoundsByPathwyT function is a table function that returns a VARCHAR(100) column with the compound identifiers that are on the pathway. The name of the column that is returned is `compound_id`.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.CompoundsByPathwyT(pathway_id)
```

The schema name that you use with this user-defined function is DB2LS.

Example of searching for all of the compounds on a specific pathway

You want to search for all of the compound identifiers on the reference pathway map 00020.

The SELECT statement that you use is:

```
SELECT * FROM TABLE(DB2LS.CompoundsByPathwyT
    ('path:map00020'))
    AS t;
```

EnzymesByPathwyS user-defined function

Use the EnzymesByPathwyS function to search for all of the enzymes on a pathway.

The EnzymesByPathwyS function is a scalar function that returns a space-delimited list of enzymes identifiers. The list is a character string with a data type of VARCHAR and an actual length that is not greater than 32767 bytes.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.EnzymesByPathwyS(pathway_id)
```

The schema name that you use with this user-defined function is DB2LS.

Example of searching for all of the enzymes on a specific pathway

You want to search for all of the enzyme identifiers that are on the same pathway as the E.coli genes.

The clause that you use is:

```
VALUES CAST(DB2LS.EnzymesByPathwyS
    ('path:eco00020')
    AS VARCHAR(1000));
```

EnzymesByPathwyT user-defined function

Use the EnzymesByPathwyT function to search for all of the enzymes on a pathway.

The EnzymesByPathwyT function is a table function that returns a VARCHAR(100) column with the names of the enzymes on the pathway. The name of the column that is returned is enzyme_id.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.EnzymesByPathwyT(pathway_id)
```

The schema name that you use with this user-defined function is DB2LS.

Example of searching for all of the enzymes on a specific pathway

You want to search for all of the enzymes on the reference pathway map 00020.

The SELECT statement that you use is:

```
SELECT * FROM TABLE(DB2LS.EnzymesByPathwyT
    ('path:map00020'))
    AS t;
```

GenesByPathwyS user-defined function

Use the GenesByPathwyS function to search for all of the genes on a pathway.

The GenesByPathwyS function is a scalar function that returns a space-delimited list of gene identifiers. The list is a character string with a data type of VARCHAR and an actual length that is not greater than 32767 bytes.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.GenesByPathwyS(pathway_id)
```

The schema name that you use with this user-defined function is DB2LS.

Example of searching for all of the genes on a specific pathway

You want to search for all of the enzymes on the same pathway as the E.coli genes.

The clause that you use is:

```
VALUES CAST(DB2LS.GenesByPathwyS
    ('path:eco00020')
    AS VARCHAR(1000));
```

GenesByPathwyT user-defined function

Use the GenesByPathwyT function to search for all of the genes on a pathway.

The GenesByPathwyT function is a table function that returns a VARCHAR(100) column with the names of the genes on the pathway. The name of the column that is returned is gene_id.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.GenesByPathwyT(pathway_id)
```

The schema name that you use with this user-defined function is DB2LS.

Example of searching for all of the genes on a specific pathway

You want to search for all of the genes on the reference pathway map 00290.

The SELECT statement that you use is:

```
SELECT * FROM TABLE(DB2LS.GenesByPathwyT
    ('path:map00290'))
    AS t;
```

PathwysByCompndsS user-defined function

Use the PathwysByCompndsS function to search for all of the pathways that contains the compounds that you specify.

The PathwysByCompndsS function is a scalar function that returns a space-delimited list of pathway identifiers. The list is a character string with a data type of VARCHAR and an actual length that is not greater than 32767 bytes.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.PathwysByCompndsS(compound_id_list)
```

The schema name that you use with this user-defined function is DB2LS.

Example of searching for the pathways that contains a specific compound

You want to search for all of the pathway identifiers that contain the compound C00158.

The clause that you use is:

```
VALUES CAST(DB2LS.PathwysByCompndsS
    ('cpd:C00158')
    AS VARCHAR(1000));
```

PathwysByCompndsT user-defined function

Use the PathwysByCompndsT function to search for all of the pathways that contains the compounds that you specify.

The PathwysByCompndsT function is a table function that returns a VARCHAR(100) column with the names of the pathways. The name of the column that is returned is `pathway_id`.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.PathwysByCompndsT(compound_id_list)
```

The schema name that you use with this user-defined function is DB2LS.

Example of searching for the pathways that contains a specific compound

You want to search for all of the pathways that contain the compound C00033.

The SELECT statement that you use is:

```
SELECT * FROM TABLE(DB2LS.PathwysByCompndsT
    ('cpd:C00033'))
    AS t;
```

PathwysByEnzymesS user-defined function

Use the PathwysByEnzymesS function to search for all of the pathways that contains the enzymes that you specify.

The PathwysByEnzymesS function is a scalar function that returns a space-delimited list of pathways. The list is a character string with a data type of VARCHAR and an actual length that is not greater than 32767 bytes.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.PathwysByEnzymesS(enzyme_id_list)
```

The schema name that you use with this user-defined function is DB2LS.

Example of searching for all of the pathways that contains a specific enzyme

You want to search for all of the pathways that contain the enzyme 1.3.99.1.

The clause that you use is:

```
VALUES CAST(DB2LS.PathwysByEnzymesS
    ('ec:1.3.99.1')
    AS VARCHAR(1000));
```

PathwysByEnzymesT user-defined function

Use the PathwysByEnzymesT function to search for all of the pathways that contains the enzyme that you specify.

The PathwysByEnzymesT function is a table function that returns a VARCHAR(100) column with the names of the pathways. The name of the column that is returned is pathway.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.PathwysByEnzymesT(enzyme_id_list)
```

The schema name that you use with this user-defined function is DB2LS.

Example of searching for all of the pathways that contains a specific enzyme

You want to search for all of the pathways that contain the enzyme 1.3.99.1.

The SELECT statement that you use is:

```
SELECT * FROM TABLE(DB2LS.PathwysByEnzymesT
    ('ec:1.3.99.1'))
   AS t;
```

PathwysByGenesS user-defined function

Use the PathwysByGenesS function to search for all of the pathways that contains the genes that you specify.

The PathwysByGenesS function is a scalar function that returns a space-delimited list of pathways. The list is a character string with a data type of VARCHAR and an actual length that is not greater than 32767 bytes.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.PathwysByGenesS(gene_id_list)
```

The schema name that you use with this user-defined function is DB2LS.

Example of searching for all of the pathways that contains a specific gene

You want to search for all of the pathways that contain the E. coli gene b0077.

The clause that you use is:

```
VALUES CAST(DB2LS.PathwysByGenesS
    ('eco:b0077')
    AS VARCHAR(1000));
```

PathwysByGenesT user-defined function

Use the PathwysByGenesT function to search for all of the pathways that contains the genes that you specify.

The PathwysByGenesT function is a table function that returns a VARCHAR(100) column with the names of the pathways for the genes. The name of the column that is returned is `pathway_id`.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.PathwysByGenesT(genes_id_list)
```

The schema name that you use with this user-defined function is DB2LS.

Example of searching for all of the pathways that contains a specific gene

You want to search for all of the pathways that contain the E. coli gene b0078.

The SELECT statement that you use is:

```
SELECT * FROM TABLE(DB2LS.PathwysByGenesT
    ('eco:0078'))
   AS t;
```

Sequence Similarity Database functions

With the KEGG user-defined functions you can search the Sequence Similarity database (SSDB) for information about the amino acid sequence similarities among all protein-encoded genes in the complete genome.

For example, you can search for:

- Homologous and Paralogous genes
- Best-best and reverse neighbors of a gene
- Motifs in a gene

Columns that are returned from SSDB database queries (table functions)

Many of the user-defined table functions for the Sequence Similarity Database (SSDB) return a set of fixed output columns. When you use a KEGG user-defined table function to search the SSDB, you can specify that only a subset of the columns are returned.

The columns that are returned depend on the function that you use and the type of row that is returned. There are three types of rows that can be returned:

- SSDB rows
- Motif rows
- Definition rows

The names and data types of the columns for each type of row are listed in the following tables.

SSDB rows

Table 94. SSDB rows from the SSDB database

Column name	Column data type	Description
genes_id1	VARCHAR (100)	The identifier for the gene that is specified in the query.
genes_id2	VARCHAR (100)	The identifier for the gene that is returned from the query.
sw_score	DOUBLE	The Smith-Waterman score between genes_id1 and genes_id2.
bit_score	DOUBLE	The bit score between genes_id1 and genes_id2.
identity	DOUBLE	The identity percent between genes_id1 and genes_id2.
overlap	INTEGER	The overlap length between genes_id1 and genes_id2.
start_position1	INTEGER	The start position of the alignment in genes_id1.
end_position1	INTEGER	The end position of the alignment in genes_id1.
start_position2	INTEGER	The start position of the alignment in genes_id2.
end_position2	INTEGER	The end position of the alignment in genes_id2.
best_flag_1to2	INTEGER	The flag that indicates the best hit from genes_id1 to genes_id2.

Table 94. SSDB rows from the SSDB database (continued)

Column name	Column data type	Description
best_flag_2to1	INTEGER	The flag that indicates the best hit from genes_id2 to genes_id1.
definition1	VARCHAR (1000)	The definition for the genes_id1 column.
definition2	VARCHAR (1000)	The definition for the genes_id2 column.
length1	INTEGER	The length of the amino acid in the genes_id1 column.
length2	INTEGER	The length of the amino acid in the genes_id2 column.

Motif rows

Table 95. Motif rows from the SSDB database

Column name	Column data type	Description
motif_id	VARCHAR (100)	The identifier for the motif.
definition	VARCHAR (1000)	The definition for the motif.
genes_id	VARCHAR (100)	The identifier for the gene that contains the motif.
start_position	INTEGER	The start position of the motif_id that matches the query.
end_position	INTEGER	The end position of the motif_id that matches the query.
score	DOUBLE	The Smith-Waterman score of the motif in the TIGRFAM and PROSITE databases that matches the query. The datatype of the score in the PROSITE database is FLOAT.
evalue	DOUBLE	The e-value of the motif in the Pfam database that matches the query.

Definition rows

Table 96. Definition rows from the SSDB database

Column name	Column data type	Description
entry_id	VARCHAR (100)	The identifier for the database entry.
definition	VARCHAR (1000)	The definition for the entry.

BestNbrsByGeneS user-defined function

Use the BestNbrsByGeneS function to search for the best neighbors or the best homologous neighbors of a gene in all organisms.

The BestNbrsByGeneS function is a scalar function that returns a space-delimited list of target identifiers. The list is a character string with a data type of VARCHAR and an actual length that is not greater than 32767 bytes.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.BestNbrsByGeneS('genes_id', start, max_result)
```

The schema name for this user-defined function is DB2LS.

Example of searching the entire organism list

You want to search for all of the organisms that are the best neighbors of the E. coli gene. You want the results that are returned to start with the first result, with a maximum of 15 results returned.

The clause that you use is:

```
VALUES CAST(DB2LS.BestNbrsByGeneS
    ('eco:b0002', 1, 15)
    AS VARCHAR(1000));
```

The KEGG code for the E. coli gene is `eco`. The gene name is `b0002`.

BestNbrsByGeneT user-defined function

Use the BestNbrsByGeneT function to search for the best neighbors or the best homologous neighbors of the gene in all organisms.

The BestNbrsByGeneT function is a table function that returns a fixed set of output columns.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.BestNbrsByGeneT('genes_id', start, max_result)
```

The schema name for this user-defined function is DB2LS.

Example of searching the entire organism list

You want to search for all of the organisms that are the best neighbors of the E. coli gene. You want the results that are returned to start with the first result, with a maximum of 15 results returned.

The SELECT statement that you use is:

```
SELECT * FROM TABLE(DB2LS.BestNbrsByGeneT
    ('eco:b0002', 1, 15))
    AS t;
```

The KEGG code for the E. coli gene is `eco`. The gene name is `b0002`.

BstBstNbrsByGeneS user-defined function

Use the BstBstNbrsByGeneS function to search for the best-best neighbors or the best-best homologous neighbors of a gene in all organisms.

The BstBstNbrsByGeneS function is a scalar function that returns a space-delimited list of target identifiers. The list is a character string with a data type of VARCHAR and an actual length that is not greater than 32767 bytes.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.BstBstNbrsByGeneS('genes_id', start, max_result)
```

The schema name for this user-defined function is DB2LS.

Example of searching the entire organism list

You want to search for all of the organisms that are the best-best neighbors of the E. coli gene. You want the results that are returned to start with the first result, with a maximum of 12 results returned.

The clause that you use is:

```
VALUES CAST(DB2LS.BstBstNbrsByGeneS
    ('eco:b0002', 1,12)
    AS VARCHAR(1000));
```

The KEGG code for the E. coli gene is `eco`. The gene name is `b0002`.

BstBstNbrsByGeneT user-defined function

Use the BstBstNbrsByGeneT function to search for the best-best neighbors or the best-best homologous neighbors of a gene in all organisms.

The BstBstNbrsByGeneT function is a table function that returns a fixed set of output columns.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.BstBstNbrsByGeneT('genes_id', start, max_result)
```

The schema name for this user-defined function is DB2LS.

Example of searching the entire organism list

You want to search for all of the organisms that are the best-best neighbors of the E. coli gene. You want the results that are returned to start with the first result, with a maximum of 12 results returned.

The SELECT statement that you use is:

```
SELECT * FROM TABLE(DB2LS.BstBstNbrsByGeneT
    ('eco:b0002', 1,12))
    AS t;
```

The KEGG code for the E. coli gene is `eco` and the gene name is `b0002`.

BtitS user-defined function

Use the BtitS function to retrieve the definitions for the genes that you specify.

The BtitS function is a scalar function that returns a semi-colon delimited list of definitions. The list is a character string with a data type of VARCHAR and an actual length that is not greater than 32767 bytes.

The maximum number of gene identifiers that you can specify for the *entrylist* argument is 100.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.BtitS(entrylist)
```

The schema name that you use with this user-defined function is DB2LS.

Example of retrieving the definitions for a list of specified genes

You want to retrieve the definitions for the M. loti, P. abyssi, and X. fastidiosa genes.

The clause that you use is:

```
VALUES CAST(DB2LS.BtitS
    ('mlo:mll1558', 'pab:PAB1288', 'xfa:XF2434'))
    AS VARCHAR(1000));
```

GenesByMotifsT user-defined function

Use the GenesByMotifsT function to search for all of the genes that contain all of the motifs in a list of motifs that you specify.

The GenesByMotifsT function is a table function that returns a definition row.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.GenesByMotifsT('motif_id_list', start, max_results)
```

The schema name for this user-defined function is DB2LS.

Example of searching for all of the genes that contain all of the motifs in a list

You want to search for all of the genes that contain both the `Pfam 'DnaJ'` and the `Prosite 'DNAJ_2'` motifs. You want the results that are returned to start with the second result, with a maximum of 10 results returned.

The SELECT statement that you use is:

```
SELECT * FROM TABLE(DB2LS.GenesByMotifsT
    ('pf:DnaJ ps:DNAJ_2', 2, 10))
    AS t;
```

MotifsByGenesS user-defined function

Use the MotifsByGenesS function to search for all of the motifs for the gene that you specify.

The MotifsByGenesS function is a scalar function that returns a space-delimited list of motif identifiers. The list is a character string with a data type of VARCHAR and an actual length that is not greater than 32767 bytes.

You can search for the motifs in a single database or in all of the valid databases. To search all of the valid databases, you specify *`all`* in the `database` parameter of the function.

The databases that you can specify are listed in the following table.

Table 97. Databases that contain motif information

Database name	Abbreviation for the database
Pfam	pfam
TIGRFAM	tfam
PROSITE pattern	pspt
PROSITE profile	pspf

You can use this function in a SELECT statement.

Syntax

```
DB2LS.MotifsByGenesS('genes_id', 'database')
```

The schema name for this user-defined function is DB2LS.

Example of searching for all of the motifs for a gene in a specific database

You want to search for all of the motifs for the E. coli gene in the PROSITE profile database.

The clause that you use is:

```
VALUES CAST(DB2LS.MotifsByGenesS
    ('eco:b0002', 'pspf'))
    AS VARCHAR(1000));
```

The KEGG code for the E. coli gene is *eco* and the gene name is *b0002*. The KEGG name for the PROSITE profile database is *pspf*.

Example of searching for all of the motifs for a gene in all of the valid databases

You want to search for all of the motifs for the E. coli gene in all four of the valid databases.

The clause that you use is:

```
VALUES CAST(DB2LS.MotifsByGenesS
    ('eco:b0002', 'all'))
    AS VARCHAR(1000));
```

The KEGG code for the E. coli gene is *eco* and the gene name is *b0002*.

MotifsByGenesT user-defined function

Use the MotifsByGenesT function to search for all of the motifs for the gene that you specify.

The MotifsByGenesT function is a table function that returns the motif row.

You can search for the motifs in a single database or in all of the valid databases. To search all of the valid databases, you specify `all` in the *database* argument of the function.

The databases that you can specify are listed in the following table.

Table 98. Databases that contain motif information

Database name	Abbreviation for the database
Pfam	pfam
TIGRFAM	tfam
PROSITE pattern	pspt
PROSITE profile	pspf

You can use this function in a SELECT statement.

Syntax

```
DB2LS.MotifsByGenesT('genes_id', 'database')
```

The schema name for this user-defined function is DB2LS.

Example of searching for all of the motifs for a gene in a specific database

You want to search for all of the motifs for the E. coli gene in the PROSITE profile database.

The SELECT statement that you use is:

```
SELECT * FROM TABLE(DB2LS.MotifsByGeneT
    ('eco:b0002', 'pfam'))
    as t;
```

The KEGG code for the E. coli gene is `eco` and the gene name is `b0002`. The KEGG name for the Pfam profile database is `pfam`.

Example of searching for all of the motifs for a gene in all of the valid databases

You want to search for all of the motifs for the E. coli gene in all four of the valid databases.

The SELECT statement that you use is:

```
SELECT * FROM TABLE(DB2LS.MotifsByGeneT
    ('eco:b0002', 'all')
    AS t;
```

The KEGG code for the E. coli gene is `eco` and the gene name is `b0002`.

ParalogsByGeneS user-defined function

Use the ParalogsByGeneS function to search for the genes that are paralogous to the gene that you specify.

The ParalogsByGeneS function is a scalar function that returns a space-delimited list of target identifiers. The list is a character string with a data type of VARCHAR and an actual length that is not greater than 32767 bytes.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.ParalogsByGeneS('genes_id', start, max_result)
```

The schema name for this user-defined function is DB2LS.

Example of searching for paralogous genes

You want to search for all of the paralogous genes for the E. coli gene. You want the results that are returned to start with the second result, with a maximum of 10 results returned.

The clause that you use is:

```
VALUES CAST(DB2LS.ParalogsByGeneS
    ('eco:b0002', 2, 10)
    AS VARCHAR(1000));
```

The KEGG database name for the E. coli organism is `eco`. The organism name is `b0002`.

ParalogsByGeneT user-defined function

Use the ParalogsByGeneT function to search for the genes that are paralogous to the gene that you specify.

The ParalogsByGeneT function is a table function that returns a fixed set of output columns.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.ParalogsByGeneT('genes_id', start, max_result)
```

The schema name for this user-defined function is DB2LS.

Example of searching for paralogous genes

You want to search for all of the paralogous genes for the E. coli gene. You want the results that are returned to start with the second result, with a maximum of 10 results returned.

The SELECT statement that you use is:

```
SELECT * FROM TABLE(DB2LS.ParalogsByGeneT
    ('eco:b0002', 2, 10))
    AS t;
```

The KEGG database name for the E. coli organism is `eco`. The organism name is `b0002`.

RevBestNbrsByGeneS user-defined function

Use the RevBestNbrsByGeneS function to search for the reverse best neighbors of a gene in all organisms.

The RevBestNbrsByGeneS function is a scalar function that returns a space-delimited list of target identifiers. The list is a character string with a data type of VARCHAR and an actual length that is not greater than 32767 bytes.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.RevBestNbrsByGeneS('genes_id', start, max_result)
```

The schema name for this user-defined function is DB2LS.

Example of searching the entire organism list

You want to search for all of the organisms that are the reverse best neighbors of the E. coli gene. You want the results that are returned to start with the first result, with a maximum of 15 results returned.

The clause that you use is:

```
VALUES CAST(DB2LS.RevBestNbrsByGeneS
    ('eco:b0002', 1, 15)
    AS VARCHAR(1000));
```

The KEGG code for the E. coli gene is `eco`. The gene name is `b0002`.

RevBestNbrsByGeneT user-defined function

Use the RevBestNbrsByGeneT function to search for the reverse best neighbors of a gene in all organisms.

The RevBestNbrsByGeneT function is a table function that returns a fixed set of output columns.

You can use this function in a SELECT statement.

Syntax

```
DB2LS.RevBestNbrsByGeneT('genes_id', start, max_result)
```

The schema name for this user-defined function is DB2LS.

Example of searching the entire organism list

You want to search for all of the organisms that are the reverse best neighbors of the E. coli gene. You want the results that are returned to start with the first result, with a maximum of 15 results returned.

The SELECT statement that you use is:

```
SELECT * FROM TABLE(DB2LS.RevBestNbrsByGeneT
    ('eco:b0002', 1, 15))
    AS t;
```

The KEGG code for the E. coli gene is `eco`. The gene name is `b0002`.

Disabling the KEGG user-defined functions

You can temporarily disable the KEGG user-defined functions or permanently remove the functions from your federated database.

About this task

If you disable the KEGG user-defined functions, you can enable the functions again by registering the functions in the federated database. You must uninstall the functions to permanently remove the functions from the federated database.

Procedure

To disable the KEGG user-defined functions:

Run the disable_KEGGFunctions command.

- On federated servers that run Windows, this command is in the sqllib\bin directory
- On federated servers that run UNIX, this command is in the sqllib/bin directory

Syntax

```
disable_KEGGFunctions -n dbName -u userID -p password [-version]
```

-n *dbName*
The name of the federated database that you want to disable the functions from.

-u *userID*
A valid user ID for the federated database.

-p *password*
A valid password for the user ID.

[-*2.3***]**
Removes all of the Version 2.3 KEGG user-defined functions from the *dbName* that you specify. You only need to use this parameter to remove Version 2.3 KEGG user-defined functions. To remove any other version of the KEGG user-defined functions you should omit this parameter.

For example, to disable the version 2.3 KEGG user-defined functions from the CHINA federated database, use the following command:

```
disable_KEGGFunctions -n CHINA -u LI -p panda ver2.3
```

After you disable the KEGG user-defined functions, you can disable XML Extender and the SOAP user-defined functions:

1. Run the db2disable_soap_udf -n *database_name* command.
2. Run the dxxadm disable_db *database_name* command.

Chapter 23. Life sciences user-defined functions

Life sciences user-defined functions

The life sciences user-defined functions provide you with algorithms that you commonly use to analyze data.

The life sciences user-defined functions use the standard single-letter codes and the IUPAC-IUB ambiguity codes to represent amino acids and nucleotides.

The life sciences user-defined functions are installed with the Life Sciences User-Defined Functions component of the nonrelational wrappers. After the life sciences user-defined functions are installed, you must register the functions.

To avoid conflicts with namespaces, all of the life sciences user-defined functions are registered in the DB2LS schema.

Life sciences user-defined function library files

Some of the life sciences user-defined functions that are included with IBM WebSphere Federation Server require library files. These library files are required when you register the life sciences user-defined functions on the federated server.

Most of the life sciences user-defined functions use the same library file. The LSGeneWise user-defined function requires a separate library file.

When you install IBM WebSphere Federation Server for nonrelational data sources, the following user-defined library files are installed on your federated server.

Life sciences user-defined function libraries

Table 99. Life sciences user-defined function library locations and file names

Function type	Operating system	Directory path	Library file name
Life sciences user-defined functions	AIX	/SQLLIB/function	libdb2lsudfs.a
Life sciences user-defined functions	Linux	/SQLLIB/function	libdb2lsudfs.so
Life sciences user-defined functions	Solaris	/SQLLIB/function	libdb2lsudfs.so
Life sciences user-defined functions	Windows	%DB2PATH%\bin	db2lsudfs.dll

%DB2PATH% is the environment variable that is used to specify the directory path where IBM WebSphere Federation Server is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

LSGeneWise user-defined function library

The LSGeneWise user-defined function requires a separate library file.

Table 100. LSGeneWise function library location and file name

Function type	Operating system	Directory path	Library file name
LSGeneWise function	UNIX	/SQLLIB/lib	libdb2lsSTgenewise.a

Life sciences user-defined functions by functional category

IBM WebSphere Federation Server includes a set of user-defined functions that you can use to access life sciences data. There are functions that can convert and align sequences, parse elements in a definition line, and perform pattern matching.

The following table lists the life sciences user-defined functions by functional category. It also provides a brief description of each category.

Table 101. Life sciences user-defined functions

Functional category	User-defined functions	Description
Back translate	LSPep2AmbNuc, LSPep2ProbNuc	Converts an amino acid sequence into a nucleotide sequence.
Defline parsing	LSDeflineParse	Parses elements of a definition line, such as that returned by the BLAST wrapper or present in a FASTA format data file.
Generalized pattern matching	LSPatternMatch, LSPrositePattern	Identifies areas of interest in a given string, such as a nucleotide or peptide sequence.
GeneWise	LSGeneWise	Aligns a protein sequence to a genomic sequence.
Motifs	LSMultiMatch, LSMultiMatch3, LSBarCode	Matches patterns in nucleotide or amino acid sequences.
Reverse	LSRevNuc, LSRevPep, LSRevComp	Reverses a nucleotide or amino acid sequence.
Translate	LSNuc2Pep, LSTransAllFrames	Converts a nucleotide sequence into a peptide sequence.

Registering life sciences user-defined functions

You must register the life sciences user-defined functions before you can use the functions.

Before you begin

The life sciences user-defined functions must be installed. The **Life sciences user-defined functions** option is listed under the Nonrelational Wrappers component in the IBM WebSphere Federation Server installation wizard.

Procedure

To register the life sciences user-defined functions:

Run the enable_LSFunctions command to register the life sciences user-defined functions.

- On federated servers that run Windows, this command is in the sqllib\bin directory
- On federated servers that run UNIX, this command is in the sqllib/bin directory

Syntax:

```
enable_LSFunctions -n dbName -u userID -p password [-force]
```

-n *dbName*
The name of the federated database that you are registering the functions in.

-u *userID*
A valid user ID for the federated database.

-p *password*
A valid password for the user ID.

[-force]
An optional flag that you can use to remove the functions and register them again. Use this flag if the functions get corrupted or dropped accidentally.

The enable_LSFunctions command registers all of the life sciences user-defined functions in the federated database. The functions are registered with the schema name DB2LS.

Disabling the life sciences user-defined functions

You can temporarily disable the life sciences user-defined functions or permanently remove the functions from your federated database.

About this task

If you disable the life sciences user-defined functions, you can enable the functions again by registering the functions in the federated database. You must uninstall the functions to permanently remove the functions from the federated database.

Procedure

To disable the life sciences user-defined functions:

Run the disable_LSFunctions command.

- On federated servers that run Windows, this command is in the sqllib\bin directory
- On federated servers that run UNIX, this command is in the sqllib/bin directory

Syntax:

```
disable_LSFunctions -n dbName -u userID -p password
```

-n *dbName*
The name of the federated database that you want to disable the functions from.

-u *userID*
A valid user ID for the federated database.

-p *password*
A valid password for the user ID.

Back translation user-defined functions - overview

Use the back translation user-defined functions to convert a peptide sequence to a nucleotide sequence. Back translation is the opposite of translation.

Because the mapping from amino acids to nucleotide triplet codons is one-to-many, the back translation produces two results:

most ambiguous
Simple text conversion and lookup. Use the LSPep2AmbNuc user-defined function to do the most ambiguous translation.

most probable
Requires additional information from a codon frequency table. Use the LSPep2ProbNuc user-defined function to do the most probable translation.

LSPep2AmbNuc user-defined function

Use the LSPep2AmbNuc function to produce the most ambiguous nucleotide sequence, according to a translation table, from a peptide sequence.

```
DB2LS.LSPep2AmbNuc(input peptide sequence,filepath to external translation table)
```

input peptide sequence
A valid character string representation describing a peptide sequence. A character string representation must have a data type of VARCHAR and an actual length that is no greater than 10890 bytes. The input data uses the standard amino acid symbols and ambiguity codes.

filepath to external translation table
If you use a customized translation table, include the file path information to find the translation table. The string value of the path must be no greater than 255 characters.

The schema name that you use with this user-defined function is DB2LS.

The result of the function is a character string with a data type of VARCHAR and an actual length that is not greater than 32672 bytes. The result represents the most ambiguous nucleotide sequence, according to a translation table, either built-in or specified by you.

If you do not specify a translation table, the function uses the following default translation table.

Table 102. Default translation table

Amino acid symbol	Abbreviation	Codon
A	Ala	GCX
B	Asx	RAY
C	Cys	TGY
D	Asp	GAY
E	Glu	GAR
F	Phe	TTY
G	Gly	GGX
H	His	CAY

Table 102. Default translation table (continued)

Amino acid symbol	Abbreviation	Codon
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

LSPep2AmbNuc user-defined function - example:

Example for the LSPep2AmbNuc user-defined function.

Back translation of a simple peptide

You can invoke the function with a values statement. The single input is a peptide sequence, as in the following example:

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

The above example transforms a peptide into a nucleotide using the ambiguous translations and the built-in default translation table. The result of the above statement is a nucleotide sequence created from the standard amino acid symbols:

```
CAYMGX
```

Back translation of a simple peptide into a nucleotide

The following example transforms a peptide into a nucleotide using the ambiguous translations and the built-in table:

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

The result of this values statement is the following nucleotide sequence. (The sequence has been split to fit on the page.)

```
WSXMGXGGXTTYGGXTTYATHACXTAYWSXCAYWSXWSXATGATHGAYGARGCXCARA
ARWSXMGXCCXCAYAARATHGAYGGXMGXGTXGTXGARCCXAARMGXGCX
```

Back translation using input from a table or nickname column

The next example shows the function applied to a set of values extracted from a table or nickname:

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

The data in column peptide_seq of table protein_table looks like the following:

Table 103. Data in the peptide_seq column

peptide_seq
GIKEDTEEHHLRDYFE
QKYHTVNGHNCEVRKA
.....

The result of the select statement is:

```
GGXATHAARGARGAYACXGARGARCAYCAYYTXMGXGAYTAYTTYGAR
CARAARTAYCAYACXGTXAAYGGXCAYAAYTGYGARGTXMGXAARGCX
...
```

Back translation using a user-defined translation table

The following example transforms a peptide into a nucleotide using the ambiguous translations and a user-defined table. Usually, the differences between translation tables are small. There might be just one or two symbols that are unique. They might occur because some species have more codons or some species have fewer codons. For example, the codon AGG is absent in Drosophila.

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

Assuming that the input translation table is for Drosophila, the result of the values statement is shown in the following example:

```
MGRGGXAAYATGGGXGGXGGXAAYTAYGGXAAYTARAAYGGXGGXGGXAAYTGGAAYAAYGGX
```

LSPep2ProbNuc user-defined function

Use the LSPep2ProbNuc function to generate the most probable nucleotide sequence, from a peptide sequence, based on a codon frequency table that you specify.

```
DB2LS.LSPep2ProbNuc(input peptide sequence,filepath to codon frequency table)
```

input peptide sequence
A valid character string representation describing a peptide sequence. The character string representation must have a data type of VARCHAR and an actual length that is no greater than 10890 bytes. The input data uses the standard amino acid symbols.

filepath to codon frequency table
This is the codon frequency table. Include the file path information to find the frequency table. The string value of the path must be no greater than 255 characters.

The schema name that you use with this user-defined function is DB2LS.

The result of the function is a character string with a data type of VARCHAR and an actual length that is not greater than 32672 bytes representing the most probable nucleotide sequence using the codon frequency table.

LSPep2ProbNuc user-defined function - example:

Example for the LSPep2ProbNuc user-defined function.

The following example shows how you can transform a peptide sequence into a nucleotide sequence using the most probable translations defined in the yeast_high.cod frequency table.

```
VALUES db2lsLSPep2ProbNuc('RDNNDDDN', '\data\yeast_high.cod')
```

The result of the above values statement is:

```
AGAGACAATAACGACGATGATAAC
```

A second execution of the same statement produces the following string:

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

A third execution of the same statement produces the following string with random values:

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

Codons in bold codons highlight the differences between the current and previous transformations.

The results from the single values statement shows that function LSPep2ProbNuc chooses one of the possible symbols based on pervious statistics. This is different from function LSPep2AmbNuc which uses ambiguous symbols where there are more possible translations.

Function LSPep2ProbNuc picks up the most probable translations for each symbol and then replaces every symbol with a random translation from the set previously picked. Assume that you have the following data in a frequency table:

Table 104. Sample frequency table data

Amino acid	Codon	Frequency
Ala	GCG	0.17
Ala	GCA	0.13
Ala	GCT	0.17
Ala	GCC	0.53

Assume that the peptide sequence contains four “A” symbols (Ala). The function translates A twice to GCC; once to GCG and once to GCT. However, the order that the function produces the translations is random. The query could translate the first A to each of translations from the set {GCC, GCC, GCG, GCT}. The result is always two occurrences of GCC, one occurrence of GCG and one occurrence of GCT in the output DNA sequence. Multiple executions of the function on the same sequence might return DNA sequences with the values interchanged.

Defline parsing user-defined functions - overview

The defline parsing user-defined functions parse elements of a definition line.

The defline parsing functions cover the most common definition line formats. Examples include definition line elements that the BLAST wrapper returns or that are present in a FASTA format data file.

The defline parsing user-defined functions can:

- Enable joins with other data sources on sequence identifiers parsed out of the defline
- Evaluate predicates on portions of the defline, such as `'species = "human"'`

LSDeflineParse user-defined functions

Definition lines that are compound definitions are output on multiple rows, with each row containing a single component definition.

Each LSDeflineParse function parses out the fields of the NCBI standard FASTA sequence identifier (NSID) and the description into columns in a table.

The schema name that you use with the defline parsing user-defined functions is DB2LS.

The defline parsing user-defined functions are:

```
DB2LS.LSDeflineParse2(definition line)
DB2LS.LSDeflineParse3(definition line)
DB2LS.LSDeflineParse2_2(definition line)
DB2LS.LSDeflineParse2_3(definition line)
DB2LS.LSDeflineParse3_3(definition line)
```

definition line

A valid string representation of a definition line in FASTA format. The string must have a data type of VARCHAR and an actual length that is no greater than 1024 bytes.

The LSDeflineParse2 user-defined function

LSDeflineParse2 parses a defline that has an NSID with two fields. The result of the function is a table with four columns:

Table 105. LSDeflineParse2 user-defined function result table column descriptions

Column name	Description
ROWID	An integer that numbers the rows returned from the function.
TAG	A VARCHAR of up to three characters that represents the NSID tag.
IDENTIFIER	A VARCHAR of up to 20 characters and represents the second identifier field in the NSID.
DESCRIPTION	A VARCHAR of up to 1019 characters.

The LSDeflineParse3 user-defined function

LSDeflineParse3 parses a defline that has an NSID with three fields. The result of the function is a table with five columns:

Table 106. LSDeflineParse3 user-defined function result table column descriptions

Column name	Description
ROWID	An integer that numbers the rows returned from the function.
TAG	A VARCHAR of up to three characters that represents the NSID tag.
ACCESSION	A VARCHAR of up to 20 characters and represents the second identifier field in the NSID.
LOCUS	A VARCHAR of up to 20 characters and represents the third identifier field in the NSID.
DESCRIPTION	A VARCHAR of up to 1017 characters.

The LSDeflineParse2_2 user-defined function

LSDeflineParse2_2 parses a defline that has a compound identifier that consists of two concatenated NSIDs, with two fields each. The result of the function is a table with six columns:

Table 107. LSDeflineParse2_2 user-defined function result table column descriptions

Column name	Description
ROWID	An integer that numbers the rows returned from the function.
TAG1	A VARCHAR of up to three characters that represents the NSID tag of the first identifier.
IDENTIFIER1	A VARCHAR of up to 20 characters and represents the second identifier field of the first NSID.
TAG2	A VARCHAR of up to three characters that represents the NSID tag of the first identifier.
IDENTIFIER2	A VARCHAR of up to 20 characters and represents the second identifier field of the second NSID.
DESCRIPTION	A VARCHAR of up to 1015 characters.

The LSDeflineParse2_3 user-defined function

LSDeflineParse2_3 parses a defline that has a compound identifier consisting of an NSID with two fields concatenated with an NSID with three fields. The order of concatenation in the input defline is not important. The result of the function is a table with seven columns:

Table 108. LSDeflineParse2_3 user-defined function result table column descriptions

Column name	Description
ROWID	An integer that numbers the rows returned from the function.
TAG1	VARCHAR of up to three characters that represents the NSID tag of the two-field identifier.
IDENTIFIER	A VARCHAR of up to 20 characters and represents the second identifier field of the two-field NSID.
TAG2	A VARCHAR of up to three characters that represents the NSID tag of the three-field identifier.
ACCESSION	A VARCHAR of up to 20 characters and represents the second identifier field of the three-field NSID.
LOCUS	A VARCHAR of up to 20 characters and represents the third identifier field of the three-field NSID.
DESCRIPTION	A VARCHAR of up to 1013 characters.

The LSDeflineParse3_3 user-defined function

LSDeflineParse3_3 parses a defline that has a compound identifier that consists of a set of two NSID, with three fields each. The result of the function is a table with eight columns:

Table 109. LSDeflineParse3_3 user-defined function result table column descriptions

Column name	Description
ROWID	An integer that numbers the rows returned from the function.

Table 109. LSDeflineParse3_3 user-defined function result table column descriptions (continued)

Column name	Description
TAG1	A VARCHAR of up to three characters that represents the NSID tag of the first identifier.
ACCESSION1	A VARCHAR of up to 20 characters and represents the second identifier field of the first NSID.
LOCUS1	A VARCHAR of up to 20 characters and represents the third identifier field of the first NSID.
TAG2	A VARCHAR of up to three characters that represents the NSID tag of the first identifier.
ACCESSION2	A VARCHAR of up to 20 characters and represents the second identifier field of the second NSID.
LOCUS2	A VARCHAR of up to 20 characters and represents the third identifier field of the second NSID.
DESCRIPTION	A VARCHAR of up to 1014 characters.

LSDeflineParse user-defined function - examples:

Examples for the LSDeflineParse user-defined function. This topic contains examples that show how the LSDeflineParse user-defined functions parse definition lines into result tables.

Parsing a definition line that contains a two-field NSID

The following example query and results table shows how the LSDeflineParse2 user-defined function parses a definition line containing a two-field NSID:

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

The result table contains the following data:

Table 110. LSDeflineParse2 user-defined function result data

Column name	Data
ROWID	1
TAG	gi
IDENTIFIER	12346
DESCRIPTION	hypothetical protein 185 – wheat chloroplast

Parsing a definition line that contains a three-field NSID

The following example query and results table shows how the LSDeflineParse3 user-defined function parses a definition line containing a three-field NSID:

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

The result table contains the following data:

Table 111. LSDeflineParse3 user-defined function result data

Column name	Data
ROWID	1
TAG	gb
ACCESSION	U37104
LOCUS	APU37104
DESCRIPTION	Aethia pusilla cytochrome b gene

Parsing a definition line that contains a compound identifier that consists of two 2-field NSID

The following example query and results table shows how the LSDeflineParse2_2 user-defined function parses a definition line containing a compound identifier that consists of two NSIDs that contain 2 fields:

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

The result table contains the following data:

Table 112. LSDeflineParse2_2 user-defined function result data

Column name	Data
ROWID	1
TAG1	gb
IDENTIFIER1	U37104
TAG2	gim
IDENTIFIER2	73401A
DESCRIPTION	Aethia pusilla cytochrome b gene

Parsing a definition line that contains a compound identifier that consists of a 2-field NSID followed by a 3-field NSID

The following example query contains a definition line with a compound identifier that consists of an NSID with two fields concatenated with an NSID with three fields. The example shows how the LSDeflineParse2_3 function parses the definition line.

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

The result table contains the following data:

Table 113. LSDeflineParse2_3 user-defined function result data

Column name	Data
ROWID	1
TAG1	gi
IDENTIFIER	12346

Table 113. LSDeflineParse2_3 user-defined function result data (continued)

Column name	Data
TAG2	gp
ACCESSION	CAA44030.1
LOCUS	CHTAHSRA_4
DESCRIPTION	hypothetical protein 185 – wheat chloroplast

Parsing a definition line that contains a compound identifier the consists of a 3-field NSID followed by a 2-field NSID

The following example query contains a definition line with a compound identifier that consists of an NSID with three fields concatenated with an NSID with two fields. The example shows how the LSDeflineParse2_3 function parses the definition line.

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

The result table contains the following data:

Table 114. LSDeflineParse2_3 user-defined function result data

Column name	Data
ROWID	1
TAG1	gi
IDENTIFIER	12346
TAG2	gp
ACCESSION	CAA44030.1
LOCUS	CHTAHSRA_4
DESCRIPTION	hypothetical protein 185 – wheat chloroplast

Parsing a definition line that contains a compound identifier that consists of two 3-field NSIDs

The following example query and results table shows how the LSDeflineParse3_3 user-defined function parses a definition line that contains a compound identifier with two NSIDs that have two fields:

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

The result table contains the following data:

Table 115. LSDeflineParse3_3 user-defined function result data

Column name	Data
ROWID	1
TAG1	dbj
ACCESSION1	AAD55586.1
LOCUS1	AF055084_1

Table 115. LSDeflineParse3_3 user-defined function result data (continued)

Column name	Data
TAG2	gp
ACCESSION2	CAA44030.1
LOCUS2	CHTAHSRA_4
DESCRIPTION	hypothetical protein 185 – wheat chloroplast

Parsing a compound definition line

You can use any of the defline user-defined functions to parse a compound definition line. The following example query contains a compound definition line with multiple definitions that are separated by a Control-A character. You can find this type of definition line in NCBI's non-redundant protein database nr. The example shows how the LSDeflineParse2_3 function parses the definition line.

```
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
```

The result table contains the following data:

Table 116. LSDeflineParse2_3 user-defined function result data

Column name	Data	Data
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

Generalized pattern matching user-defined functions - overview

The generalized pattern matching user-defined functions identify areas of interest in a given string, such as a nucleotide or peptide sequence.

There are two generalized pattern matching user-defined functions:
- LSPatternMatch
- LSPrositePattern

The LSPatternMatch user-defined function is used to search the input nucleotide or peptide sequence for a pattern that you specify.

The LSPrositePattern user-defined function is used to convert a pattern that is in the PROSITE syntax to the Perl syntax.

LSPatternMatch user-defined function

You can use the LSPatternMatch user-defined function to search the input nucleotide or peptide sequence for a pattern you specify.

```
DB2LS.LSPatternMatch(input character sequence, pattern)
```

input character sequence
: The character string representation must have a data type of VARCHAR and an actual length that is no greater than 32672 bytes.

pattern
: The pattern as specified in any valid Perl regular expression. The character string representation must have a data type of VARCHAR and an actual length that is no greater than 32672 bytes.

The schema name that you use with this user-defined function is DB2LS.

You can use the LSPatternMatch user-defined function to search the input nucleotide or peptide sequence for a pattern you specify.

The result of the function is an integer representing the position of the first match of the pattern in the sequence. The function returns a value of zero if there is no match.

If you have patterns written with the PROSITE syntax, you can convert them to Perl syntax with the LSPrositePattern user-defined function. You can then use the converted syntax with the LSPatternMatch user-defined function.

LSPatternMatch user-defined function – example:

Example for the LSPatternMatch user-defined function.

Find matches that start with specific characters

In the following example, look for the beginning position of the string that matches “coward”, “cowage”, “cowboy”, or “cowl”.

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

The function searches by characters, and in this example, returns a value of nine. The string “cowboy” begins at position nine, assuming that the first position is one.

Find matches that start with specific characters and exclude specific words

In the next example, look for the beginning position of the string that matches “not ” or “non ”:

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

The function searches by characters, and in this example, returns a value of seven. The string “not ” begins at position seven, assuming that the first position is one.

SELECT statements to filter using Perl syntax

LSPatternMatch is useful in SELECT statements to filter the results using the Perl syntax, which is a more powerful syntax than the SQL LIKE statement. In the following example, use LSPatternMatch on a blast output to filter the genes that match a certain pattern:

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

SELECT statements to filter using PROSITE syntax

If you are more familiar with the PROSITE syntax, you can use the LSPrositePattern function with the above query. Change the query to the following:

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

SELECT statements to search other types of text

The pattern matching functions are useful for searching other types of text, as well as the nucleotide or peptide sequences. Consider using the SQL LIKE statement when performance might be a concern.

The following example shows a query that filters BLAST hsp alignments based on protein motifs found in the subject or target line of the alignment. The example is adapted from 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.

The following query returns only alignments in which the subject sequence includes the P-loop ATPase domain [GA]xxxxGK[ST]. The query uses CED4, the *Caenorhabditis elegans* regulator of cell death, as a query sequence against NCBI's non-redundant protein sequence database. The database retrieves the blast query sequence from the translation of the CDS feature of GenBank entry 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;
```

SELECT statements to find HSPs

You can use the next example query to find HSPs in a genomic sequence that contain putative single nucleotide polymorphisms (SNPs) with respect to a canonical query sequence. It is adapted from 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.

The query uses the pattern matching on the blast hsp midline to find a pattern of ≥20 perfect matches followed by a single mismatch followed by ≥20 perfect matches. That is, 20 ″|″ characters, a single space, and then 20 ″|″ characters in the midline of the alignment.

This example also shows the usage of the LSPatternMatch user-defined function on strings that are not nucleotide or peptide sequences.

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

You can rewrite the previous query as:

```
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
```

This second query will return the blast rows that have a match together with the matched string and their position in the sequence.

BlastOutput is a view over a BlastN nickname.

LSPrositePattern user-defined function

Use the LSPrositePattern user-defined function to convert a pattern from the PROSITE syntax to the Perl syntax.

After you use the LSPrositePattern user-defined function to convert from the PROSITE syntax to the Perl syntax, you can then use the converted syntax with the LSPatternMatch, LSMultiMatch, and LSMultiMatch3 user-defined functions.

```
DB2LS.LSPrositePattern(pattern)
```

pattern
The pattern matching syntax specified by the Prosite syntax. The character string representation must have a data type of VARCHAR and an actual length that is no greater than 32672 bytes.

The schema name that you use with this user-defined function is DB2LS.

The result of the function is a character string representing a regular expression in the Perl syntax. The character string representation must have a data type of VARCHAR and an actual length that is no greater than 32672 bytes.

LSPrositePattern user-defined function - example:

Example for the LSPrositePattern user-defined function.

Convert patterns from PROSITE syntax into Perl syntax

In the following example, convert a pattern from PROSITE syntax into Perl syntax.

```
VALUES db2lsLSPrositePattern('[AC]-x-V-x(4)-{ED}.');
```

The function converts the input pattern in PROSITE syntax into an equivalent pattern in Perl syntax, as shown in the following example:

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

The next example converts another syntax pattern from PROSITE into the Perl syntax:

```
VALUES db2lsLSPrositePattern('<A-x-[ST](2)-x(0,1)-V.');
```

The function translate the string from the PROSITE syntax based on the input pattern and returns the following:

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

Convert patterns corresponding to a PROSITE database entry into a Perl pattern

The next example converts the pattern corresponding to the PROSITE database entry with the ID number of PS01205 into a Perl pattern that is used as input by the pattern matching functions.

```
VALUES db2lsLSPrositePattern('R-P-L-[IV]-x-[NS]-F-G-S-[CA]-T-C-P-x-F.')
```

The result of this query is:

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

Use LSPrositePattern UDF in SELECT statements

The next example shows how you can use the function in a query. The query prints out only sequences that match the PROSITE pattern specified.

```
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;
```

Use LSPrositePattern UDF in a VALUES clause

The next example converts the pattern corresponding to the PROSITE entry whose ID is PS00261:

```
VALUES db2lsLSPrositePattern('C-[STAGM]-G-[HFYL]-C-x-[ST].')
```

The result of this query is:

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

GeneWise user-defined function - overview

The GeneWise user-defined function aligns a protein sequence with a genomic sequence.

GeneWise is a commonly-used component which aligns a protein sequence with a genomic DNA sequence, allowing for introns and frame-shift errors.

Linking to GeneWise

This topic describes the procedure for linking to the GeneWise library.

Procedure

To link to the GeneWise library:

1. Download the Wise2 package version 2.1.20c from http://www.ebi.ac.uk/Wise2
2. Expand the archive into a folder of your preference.
3. Compile the package with pthread support. For more information on this step, see the Wise2 documentation.
4. On HP-UX federated servers, you need to add the option `+z` to the compiler options before you compile the source code.

   To add this option, open the file makefile from the root directory of the Wise2 package and change the line as shown in the following example:

   ```
   CFLAGS = -c -O -DPTHREAD to: CFLAGS = -c -O -DPTHREAD  +z
   ```
5. Run `make api` in root directory of the Wise2 package.
6. Set the WISE2_HOME environment variable to point to the Wise2 package root directory. For example: `export WISE2_HOME=/usr/wise2.1.20c`

7. Set the WISECONFIGDIR variable in the sqllib/cfg/db2dj.ini file to point to the wisecfg subdirectory.

   For example, if the Wise2 package is installed in the /usr/wise2.1.20c/ directory, add the following variable to the db2dj.ini file:

   ```
   WISECONFIGDIR=/usr/wise2.1.20c/wisecfg/
   ```

   On UNIX federated servers, the default path for the db2dj.ini file is INSTHOME/sqllib/cfg, where INSTHOME is the home directory of the instance owner. On Windows federated servers, the default path to the file is x:\SQLLIB\cfg, where x:\SQLLIB is the drive and directory specified in the DB2PATH registry variable or environment variable.
8. Run the djxlinkLSGeneWise script. This script is located in the stable/bin directory. Check the output from the script in the djxlinkLSGeneWise.out file. The output file is located in the sqllib/function directory. If no errors are reported in the output file, then the library was successfully built.

LSGeneWise user-defined function

Use the LSGeneWise user-defined function to align a protein sequence with a genomic DNA sequence, allowing for introns and frameshifting errors.

Syntax

```
DB2LS.LSGeneWise(protein sequence, DNA_sequence)
```

protein sequence
A valid character string representation describing a peptide sequence. The character string representation must have a data type of VARCHAR and an actual length that is no greater than 32672 bytes.

DNA_sequence
A valid character string representation describing a nucleotide sequence. The character string representation must have a data type of VARCHAR and an actual length that is no greater than 32672 bytes.

The schema name that you use with this user-defined function is DB2LS

The following table shows the one row output table the LSGeneWise function returns.

Table 117. Column names, types, and descriptions for the output table from the LSGeneWise function

Column name	Type	Description
PROTEIN_OFFSET	INTEGER	Represents the starting offset in the input protein sequence at which an alignment was found.
DNA_OFFSET	INTEGER	Represents the starting offset in the input DNA sequence at which an alignment was found.
PROTEIN	VARCHAR(32672)	A fragment from the input sequence representing the aligned sequence.
SIMILARITY	VARCHAR(32672)	Shows the matching between the protein and the dna sequences. Perfect matches are marked with the corresponding symbol letter. Non-perfect matches with a positive score are indicated with the ″+″ sign, and mismatches are indicated with a space.

Table 117. Column names, types, and descriptions for the output table from the LSGeneWise function (continued)

Column name	Type	Description
TRANSLATED_DNA	VARCHAR(32672)	The translated DNA sequence. The sequence might contain dashes and special symbols like deletions and introns.
DNA	VARCHAR(32672)	The DNA sequence with special markers like frameshifting and introns.

The output of the GeneWise program corresponds to the output of the LSGeneWise user-defined function:

- The protein and DNA offsets that are returned by the GeneWise program match the PROTEIN_OFFSET and DNA_OFFSET columns.
- The protein sequence returned on the first line by GeneWise program matches the PROTEIN column of the UDF.
- The second line in GeneWise output, the similarity line, matches the SIMILARITY column.
- The third line in the GeneWise output matches the TRANSLATED_DNA column.
- The fourth, fifth and sixth lines in the GeneWise output match the DNA column. These lines are combined, by reading the lines vertically.

For more information on the LSGeneWise user-defined function output, see http://www.ebi.ac.uk/Wise2.

LSGeneWise user-defined function – example:

Example for the LSGeneWise user-defined function.

The following example shows a query using the LSGeneWise user-defined function and the resulting data.

```
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;
```

Table 118. Results table

Column	Data
PROTEIN_OFFSET	23
DNA_OFFSET	14
PROTEIN	KLFVGALKDDHDEQSIRDYFQHFGNIVDINIVIDKETGKKRGFAFVEFDDYDPVDKVVLQKQHQLNGKMVDVKKALPKQNDQQGGGGGRGGPGGRAGGNRGNMGGGNYGNQNGGGNWNNGGN
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

Table 118. Results table (continued)

Column	Data
TRANSLATED_DNA	KIFVGGIKEDTEEHHLRDYFEQYGKIEVIEIMTDRGSGK KRGFAxVTFDDHDSVDKIVIQKYHTVNGHNCEVRKAL SKQEMASASSSQRGRSGS------ GNFGGGRGGGFGGNDNFGRGGN
DNA	aagatatttgttggtggcattaaagaagacactgaagaacatcacctaagagat...

Motif user-defined functions - overview

Motif user-defined functions match patterns in nucleotide or amino acid sequences.

There are three motif user-defined functions:

- LSBarCode
- LSMultiMatch
- LSMultiMatch3

LSBarCode user-defined function

Use the LSBarCode user-defined function to input a sequence to generate another sequence that replaces every character except spaces and plus signs with the vertical bar symbol (|).

```
DB2LS.LSBarCode(input string sequence)
```

input string sequence
A valid character string representing an HSP alignment between two sequence fragments. The character string representation must have a data type of VARCHAR and an actual length that is no greater than 32672 bytes.

The schema name that you use with this user-defined function is DB2LS.

The result of the function is a sequence of variable characters that represent a barcode sequence.

LSBarCode user-defined function - example:

Example for the LSBarCode user-defined function.

Barcodes from a string sequence

This example creates a barcode from a string sequence:

```
VALUES db2lsLSBarCode(
   'MDY +G++L  GN  ++ +PASLTK+MT YVV +A+ + +I   D+VTVG+DAWA  NP ')
```

The result of this values statement is:

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

Compute a percentage from the perfect matches in an alignment

A researcher running a BLAST search wants to return only HSP alignments that contain fewer than 25% prolines that are among their perfect matches. This example uses the LSBarCode user-defined function to compute the percentage of prolines (symbol ’P’) that are among the perfect matches in an alignment that is returned by BLAST.

This example invokes the LSMultiMatch3 user-defined function to find the perfect matches. The LSMultiMatch3 user-defined function is used in conjunction with the LSBarCode function in this example because BLAST does not always return a sequence of bars ("|") in an alignment. The following example shows this:

```
Query:       MDYTTGQILTAGNEHQQRNPASLTKLMTGYVVDRAIDSHRITPDDIVTVGRDAWAKDNPV
Alignment:   MDY +G++L  GN  ++ +PASLTK+MT YVV +A+ + +I   D+VTVG+DAWA  NP
Target:      MDYASGKVLAEGNADEKLDPASLTKIMTSYVVGQALKADKIKLTDMVTVGKDAWATGNPA
```

To ensure that the output is aligned with the correct sequence of bars, use the LSBarCode user-defined function. The LSBarCode user-defined function replaces all of the characters except spaces and plus signs with a vertical bar.

For example:

```
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;
```

In this query, BlastOutput is actually a view over a Blast nickname. The query uses the LSMultiMatch3 user-defined function to return the perfect matches on alignment. The first usage of the LSMultiMatch3 user-defined function returns the perfect matches for symbol "P", the second usage of the LSMultiMatch3 user-defined function returns all of the perfect matches.

A row from the result table is shown in the following table.

Table 119. Sample results row

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

The previous query was adapted from 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.

LSMultiMatch user-defined function

```
DB2LS.LSMultiMatch(input nucleotide or peptide sequence, pattern)
```

input nucleotide or peptide sequence

A valid character string representation describing a nucleotide or peptide sequence. The character string representation must have a data type of VARCHAR and an actual length that is no greater than 32672 bytes.

pattern

The pattern matching grammar specified by the Perl language. The character string representation must have a data type of VARCHAR and an actual length that is no greater than 32672 bytes.

The schema name that you use with this user-defined function is DB2LS.

Use the LSMultiMatch user-defined function to return a table for each match that does not overlap in the input sequence. Each table consists of a start position and the matching sequence fragment.

The result of the function is a table with two columns. The first column is an integer representing the start position of a match of the pattern in the sequence. The second column is the matching sequence fragment.

LSMultiMatch user-defined function - example:

This example looks for the position and the matching fragments for all the non-overlapping matches taken from the input.

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

The query returns a table that is based on this select statement that shows the results of the matches:

Table 120. Result of LSMultiMatch returning multiple rows

POSITION	MATCH
7	not
15	non

LSMultiMatch returns the position and the matched string for all matches. The following example searches Entrez Nucleotide for sequence entries that contain a certain motif. The query prints the sequence identifiers and the matched sequences. The sub-patterns ".{0,9}" at the beginning and at the end have to match up to nine characters before and after the sequence. The query also prints these characters.

```
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;
```

The result of this query is as follows:

Table 121. Search Entrez data

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

LSMultiMatch3 user-defined function

```
DB2LS.LSMultiMatch3(input string1, pattern1, input string2,
  pattern2, input string3, pattern3)
```

input strings
A valid character string representation describing a nucleotide or peptide sequence, or an HSP_Midline string from a blast alignment. The character string representation must have a data type of VARCHAR and an actual length that is no greater than 32672 bytes.

pattern
The pattern matching grammar specified by the Perl language. The character string representation must have a data type of VARCHAR and an actual length that is no greater than 32672 bytes.

The schema name that you use with this user-defined function is DB2LS.

Use the LSMultiMatch3 user-defined function to input three patterns and three strings and return any positions where all three strings match their respective patterns. You can use this user-defined function to perform a pattern match on an alignment.

The result of the function is a table with four columns. The first column is an integer representing the start position of a match of the pattern in all the sequences. The function anchors all the strings together at the first position. The second, third, and fourth columns are the matching sequence fragments.

LSMultiMatch3 user-defined function – example:
Use the LSMultiMatch3 function with the LSBarCode function

The following example uses the function to compute the percentage of a particular amino-acid symbol among the perfect matches returned by Blast. Notice that this example also invokes the LSBarCode user-defined function. The query needs this because Blast does not always return a sequence of bars ("|") in an alignment. The following example illustrates this:

```
Query:       MDYTTGQILTAGNEHQQRNPASLTKLMTGYVVDRAIDSHRITPDDIVTVGRDAWAKDNPV
Alignment:   MDY +G++L  GN  ++ +PASLTK+MT YVV +A+ + +I   D+VTVG+DAWA  NP
Target:      MDYASGKVLAEGNADEKLDPASLTKIMTSYVVGQALKADKIKLTDMVTVGKDAWATGNPA
```

To ensure that the output is aligned with the correct sequence of bars, use the LSBarCode function to convert the sequence. The function replaces all non-space and non-"+" characters with a vertical bar.

```
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;
```

In this query, BlastOutput is a view over a Blast select. The query uses the LSMultiMatch3 function to return the perfect matches on alignment. The first usage returns the perfect matches for symbol "P", the second usage of the LSMultiMatch3 function returns all the perfect matches. A row from the result table is shown in the following table.

Table 122. Sample results row

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

The previous query was adapted from `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.`

Find unique patterns in separate string fragments

The following example looks for three separate patterns in three separate string fragments:

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

The query returns the positions and the matching strings for all of the matches, as shown in the following table:

Table 123. Result of a multi-match using three inputs

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

The next example finds three separate patterns within three separate string fragments:

```
SELECT position, match_1, match_2, match_3
  FROM table
  (LSMultiMatch3('cbccbbcccbbbccccbbbbccccc','c{1,3}b{1,3}c{1,3}',
   'abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz',
   '.','0123456789012345678901234567890123456789','\d')) as f
```

The results are in the following table:

Table 124. Result of a multi-match using three inputs

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

Reverse user-defined functions - overview

Reverse user-defined functions reverse a nucleotide or amino acid sequence.

There are three reverse user-defined functions:

- LSRevComp
- LSRevNuc
- LSRevPep

LSRevComp user-defined function

```
DB2LS.LSRevComp(input nucleotide sequence)
```

input nucleotide sequence

A valid character string representation describing a nucleotide sequence. The sequence can contain IUPAC ambiguity codes. A character string representation must have a data type of VARCHAR and an actual length that is no greater than 32672 bytes.

The schema name that you use with this user-defined function is DB2LS.

The result of the function is a character string with a data type of VARCHAR and an actual length that is not greater than 32672 bytes representing the reverse complement of the nucleotide sequence.

LSRevComp user-defined function - example:
Use the LSRevComp function in SQL statements

You can use the LSRevComp function in an SQL statement wherever you would use any built-in function that accepts a nucleotide sequence. For example:

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

This example uses the function to return the reverse complement of the input sequence that comes from a host variable.

If you use an invalid string, or invalid data type, you get the following error message:

```
SQL0443N Routine "DB2LS.LSREVCOMP" (specific name "LSREVCOMP") has returned
an error SQLSTATE with diagnostic text "Sequence not valid". SQLSTATE=38608
```

An exception is raised if the input alphabet is not correct.

Query BLAST data sources

The following example shows how the LSRevComp user-defined function works in a query:

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

The results of the query are shown below:

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

AGTATTGGTCAATCTTTTGAT          ATCAAAAGATTGACCAATACT

TGGTCAATCTTTTGATA              TATCAAAAGATTGACCA
```

```
TTGGCCAATCTTTTGATATCC         GGATATCAAAAGATTGGCCAA

TCAATCTTTTGATATCC             GGATATCAAAAGATTGA

GGATATCAAAAGATTGA             TCAATCTTTTGATATCC


  5 record(s) selected.
```

Use the LSRevComp function with other life sciences UDFs

You can use the reverse function along with other life sciences user-defined functions to translate the reverse complement of a nucleotide sequence, as in the following example:

```
VALUES db2lsLSNuc2Pep(
       db2ls.LSRevComp('TTTTTCTTATTGTCTTCCTCATCGTATTTCTTATGTTGCTGATGT'))
```

The query returns the following:

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

LSRevNuc user-defined function

```
DB2LS.LSRevNuc(input nucleotide sequence)
```

input nucleotide sequence
A valid character string representation describing a nucleotide sequence. A character string representation must have a data type of VARCHAR and an actual length that is no greater than 32672 bytes. The nucleotide sequence must be part or all of the DNA alphabet.

The schema name that you use with this user-defined function is DB2LS.

The result of the function is a character string with a data type of VARCHAR and an actual length that is not greater than 32672 bytes representing the reverse order of the nucleotide sequence.

LSRevNuc user-defined function - example:
Use the LSRevNuc function in SQL statements

You can use the LSRevNuc function in an SQL statement wherever you would use any built-in function that accepts a nucleotide sequence. For example:

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

This example uses the function to reverse input data that comes from a host variable.

If you use an invalid string, or invalid data type, you get the following error message:

```
SQL0443N Routine "DB2LS.LSREVNUC" (specific name "LSREVNUC") has returned
an error SQLSTATE with diagnostic text "Sequence not valid". SQLSTATE=38608
```

Query BLAST data sources

The following example shows the use of the LSRevNuc user-defined function in a query.

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

The following results table shows the reversed nucleotide sequences the query returns:

```
HSP_H_SEQ                                          REV_HSP_H_SEQ

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

GCGCTAGCCCCCTACGTATTAC                             CATTATGCATCCCCCGATCGCG

GTAATACGTAGGGGGCTAGCG                              GCGATCGGGGGATGCATAATG

GTAATACGTAGGGGGCTAGCG                              GCGATCGGGGGATGCATAATG

GTAATACGTAGGGGGCTAGCG                              GCGATCGGGGGATGCATAATG


  5 record(s) selected.
```

LSRevPep user-defined function

```
DB2LS.LSRevPep(input peptide sequence)
```

input peptide sequence
A valid character string representation describing a peptide sequence. A character string representation must have a data type of VARCHAR and an actual length that is no greater than 32672 bytes. The input sequence must be part of the protein alphabet.

The schema name that you use with this user-defined function is DB2LS.

The result of the function is a character string with a data type of VARCHAR and an actual length that is not greater than 32672 bytes representing the reverse order of the peptide sequence.

LSRevPep user-defined function - example:
Use the LSRevPep function in SQL statements

You can use the LSRevPep function in an SQL statement wherever you would use any built-in function that accepts a peptide sequence. For example:

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

This example uses the function to reverse input data that comes from a host variable.

If you use an invalid string, or invalid data type, you get the following error message:

```
SQL0443N Routine "DB2LS.LSREVPEP" (specific name "LSREVPEP") has returned
an error SQLSTATE with diagnostic text "Sequence not valid". SQLSTATE=38608
```

Query BLAST data sources

The following example shows how the LSRevPep user-defined function is used in a query:

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

The following table shows the reversed peptide sequences the query returns.

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

MLCEIECRALSTAHTRLIHDFEPRDALTYL...   HDETFINKGELYTLADRPEFDHILRTHATS...

RVVSTEHTRLVTDAYPEFSISFTATKN         NKTATFSISFEPYADTVLRTHETSVVR

STAHIRVLRDMVPGDEITCFYGSEFF          FFESGYFCTIEDGPVMDRLVRIHATS

AHTRRCPDHEPRGVITYL                  LYTIVGRPEHDPCRRTHA


  4 record(s) selected.
```

Translate user-defined functions - overview

The translation user-defined functions convert a nucleotide sequence into a peptide sequence.

There are two translate user-defined functions:

- LSNuc2Pep
- LSTransAllFrames

LSNuc2Pep user-defined function

```
DB2LS.LSNuc2Pep(input nucleotide sequence,filepath to external translation table)
```

input nucleotide sequence

A valid character string representation describing a nucleotide sequence. A character string representation must have a data type of VARCHAR and an actual length that is no greater than 32672 bytes.

filepath to external translation table

If you use a customized translation table, include the file path information to find the translation table. The string value of the path must be no greater than 255 characters.

The schema name that you use with this user-defined function is DB2LS.

The result of the function is a character string with a data type of VARCHAR and an actual length that is not greater than 10890 bytes representing the peptide sequence.

The input is a nucleotide sequence using the IUB character set. The functions assume that the first codon begins at the first character of the nucleotide sequence. If the first codon does not begin at the first character of the nucleotide sequence, use a SUBSTR function on the input sequence.

The result of the function is a peptide sequence, using the standard amino acid symbols.

The LSNuc2Pep function:

- Excises spaces in input sequences.
- Ignores extraneous nucleotides outside of a reading frame.
- Returns null output if you input a null nucleotide sequence.

LSNuc2Pep user-defined function – example:

Use the LSNuc2Pep function in a values statement

Assume that you want to translate your nucleotide sequence data into a peptide sequence. This example assumes that the first codon begins at the first character of the nucleotide sequence.

You can invoke the function with a values statement. The single input is a nucleotide sequence, as in the following example:

```
VALUES db2lsLSNuc2Pep('TTTTTCTTATTGTCTTCCTCATCGTATTTCTTATGTTGCTGATGT')
```

The result of the above statement is a peptide sequence using the standard amino acid symbols:

```
FFLLSSSSYFLCC*C
```

If you want the translation in the +2 reading frame, then use the following example:

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

The integer in the statement indicates the starting position of the search for the codon.

Use the LSNuc2Pep function as a predicate in a query

Here is an example of using this function as a predicate in a query.

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

The result is shown in the following table.

Table 125. Results using the LSNuc2Pep function as a predicate

ID	PROTEINNAME	PEPTIDESEQ
1	proteinA	FSYCLPHRISYVAD

Use the LSNuc2Pep function with an external translation table

The following example translates a nucleotide sequence into a peptide sequence using an external translation table. The first parameter is the nucleotide sequence, and the second parameter is the path to the external translation table.

```
VALUES db2lsLSNuc2Pep('TTTTCTTATTGTCTTCCTCATCGTATTTCTTATGTTGCTGATGT',
                      'C:\translation.txt')
```

The result of the above statement using this particular translation table is the following string:

```
FSYCLPHRISYVAD
```

Use the LSNuc2Pep function with other life sciences UDFs

The following example combines two of the user-defined functions to demonstrate the additional uses of the functions:

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

Notice that the previous example returns the same result as the following query:

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

LSTransAllFrames user-defined function

```
DB2LS.LSTransAllFrames(input nucleotide sequence,
   filepath to external translation table)
```

input nucleotide sequence
A valid character string representation describing a nucleotide sequence. The input sequence can contain IUPAC ambiguity codes. A character string representation must have a data type of VARCHAR and an actual length that is no greater than 32672 bytes.

filepath to external translation table
If you use a customized translation table, include the file path information to find the translation table. The string value of the path must be no greater than 255 characters.

The schema name that you use with this user-defined function is DB2LS.

Use the LSTransAllFrames user-defined function to produce a set of peptide sequences from a given nucleotide sequence. These peptide sequences represent possible translations of the input nucleotide sequence, in each of 6 frames. This function is useful when the input contains errors or the reading frame is not known.

The result of the function is a table with two columns. The first column is labelled READFRAME and represents the frame that is used for the translation. This column has an integer value that represents the start position of translation. A negative integer indicates a translation of the opposite strand. The second column, called PEPTIDE, is a character string with a data type of VARCHAR and an actual length that is not greater than 10890 bytes representing the peptide sequence.

The LSTransAllFrames function:

- Excises spaces in input sequences.
- Ignores extraneous nucleotides outside of a reading frame.
- Returns null output if you input a null nucleotide sequence.

LSTransAllFrames user-defined function - example:
Use a built-in translation table

Assume that you want to translate a nucleotide sequence in all six reading frames using the built-in translation table. The following example shows how to do this:

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

The query returns the peptides in a table, as in the following example:

Table 126. Result of translating a nucleotide sequence

READFRAME	PEPTIDE
1	FFLLSSSSYFLCC*C
2	FSYCLPHRISYVAD
3	FLIVFLIVFLMLLM

Table 126. Result of translating a nucleotide sequence (continued)

READFRAME	PEPTIDE
–1	TSAT*EIR*GRQ*EK
–2	HQQHKKYDEEDNKK
–3	ISNIRNTMRKTIRK

Use a customized translation table

The next example uses a customized translation table to translate a nucleotide sequence in all six reading frames.

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

The resulting table is the same as the previous example because the input sequence is the same and translation table is the same as the one built into the function.

Use the LSTransAllFrames function instead of combining multiple life sciences functions

The following example combines two of the user-defined functions to demonstrate the additional uses of the functions:

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

Notice that the previous example returns the same result as the following query:

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

Select a specific reading frame from the output

The following example selects a specific reading frame from the output produced by the LSTransAllFrames function.

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

The result of this query is:

Table 127. Readframe function usage

READFRAME	PEPTIDE
–2	HQQHKKYDEEDNKK

Codon frequency table format

A codon frequency table shows the frequency to which the amino acids are back translated into a particular codon.

The LSPep2ProbNuc user-defined function uses the codon frequency table to determine a nucleotide sequence from a given peptide sequence.

The following list describes the format of the codon frequency table file:

- Two adjacent periods mark the beginning of the table. Any text that comes before is commentary. The two adjacent periods are required even if there is no commentary before them.
- The table contains the following columns:
  1. Am-Acid: a three letter code for the amino acid symbol.
  2. Codon: the codon for that amino acid symbol.
  3. Number: the number of occurrences of that codon in the genes from which the table is compiled.
  4. x/1000: the expected number of occurrences of the amino acid, codon pair per 1000 translations in genes.
  5. Fraction: the fraction of occurrences of the codon in its synonymous codon family.

IBM WebSphere Federation Server provides sample translation tables in the sqllib/samples/lifesci/ls_udfs subdirectory.

Codon frequency table - example

Example of a codon frequency table

The following figure shows the format of a sample codon frequency table.

```
Am-Acid Codon    Number       x/1000   Fraction   ..

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
... (truncated)
```

Figure 38. Sample codon frequency table

Translation table format

Some of the life sciences user-defined functions use a translation table.

This topic describes the format of a translation table that are used by the LSPep2AmbNuc, LSTransAllFrames, and LSNuc2Pep life sciences user-defined functions.

The following list describes the format of the translation table file:

- Two adjacent periods mark the beginning of the table. Any text that comes before is commentary.

- Each line of the table consists of a single-letter amino acid symbol, the three-letter amino acid name, the unambiguous codons, an exclamation mark, and the ambiguous codons. White space separates each word in the line.
- Each codon and amino acid symbol must appear only once in the file.
- Stop codons translate to the symbol '*'.
- Codons made up of lowercase letters are start codons.
- All other codons are uppercase.
- Codons that do not have a translation to a corresponding amino acid symbol are translated to the symbol 'X'.

IBM WebSphere Federation Server provides sample translation tables in the sqllib/samples/lifesci/ls_udfs subdirectory.

Translation table - example

Example of the sample translation table

The following figure shows the format of a sample translation table.

```
                 Standard Translation Table

Symbol   3-letter    Codons                    ! 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
```

Figure 39. Sample translation table

Chapter 24. Federated system and data source configuration parameters

Views in the global catalog table containing federated information

Most of the catalog views in a federated database are the same as the catalog views in any other DB2 Version 9.1 for Linux, UNIX, and Windows database.

There are several unique views that contain information pertinent to a federated system, such as the SYSCAT.WRAPPERS view.

The DB2 Version 8 SYSCAT views are read-only. If you issue an UPDATE or INSERT operation on a view in the SYSCAT schema, it will fail. Using the SYSSTAT views is the recommended way to update the system catalog. Change applications that reference the SYSCAT view to reference the updatable SYSSTAT view instead.

The following table lists the SYSCAT views which contain federated information. These are read-only views.

Table 128. Catalog views typically used with a federated system

Catalog views	Description
SYSCAT.CHECKS	Contains check constraint information that you defined.
SYSCAT.COLCHECKS	Contains columns referenced by a check constraint.
SYSCAT.COLUMNS	Contains column information about the data source objects (tables and views) that you created nicknames for.
SYSCAT.COLOPTIONS	Contains information about column option values that you set for a nickname.
SYSCAT.CONSTDEP	Contains the dependency of an informational constraint that you defined.
SYSCAT.DATATYPES	Contains data type information about local built-in and user-defined DB2 data types.
SYSCAT.DBAUTH	Contains the database authorities held by individual users and groups.
SYSCAT.FUNCMAPOPTIONS	Contains information about option values that you have set for a function mapping.
SYSCAT.FUNCMAPPINGS	Contains the function mappings between the federated database and the data source objects.
SYSCAT.INDEXCOLUSE	Contains columns that participate in an index.
SYSCAT.INDEXES	Contains index specifications for data source objects.
SYSCAT.INDEXOPTIONS	Contains information about index options.

Table 128. Catalog views typically used with a federated system (continued)

Catalog views	Description
SYSCAT.KEYCOLUSE	Contains columns that participate in a key defined by a unique key, primary key, or foreign key constraint.
SYSCAT.NICKNAMES	Contains information about nicknames that you created.
SYSCAT.REFERENCES	Contains information about referential constraints that you defined.
SYSCAT.ROUTINES	Contains local DB2 user-defined functions, or function templates. Function templates are used to map to a data source function.
SYSCAT.REVTYPEMAPPINGS	This view is not used. All data type mappings are recorded in the SYSCAT.TYPEMAPPINGS view.
SYSCAT.ROUTINEOPTIONS	Contains information about federated routine option values.
SYSCAT.ROUTINEPARMOPTIONS	Contains information about federated routine parameter option values.
SYSCAT.ROUTINEPARMS	Contains a parameter or the result of a routine defined in SYSCAT.ROUTINES.
SYSCAT.ROUTINESFEDERATED	Contains information about federated routines that you defined.
SYSCAT.SERVERS	Contains server definitions that you create for data source servers.
SYSCAT.TABCONST	Each row represents a table and nickname constraints of type CHECK, UNIQUE, PRIMARY KEY, or FOREIGN KEY.
SYSCAT.TABLES	Contains information about each local DB2 table, federated view, and nickname that you create.
SYSCAT.TYPEMAPPINGS	Contains forward data type mappings and reverse data type mappings. The mapping is to local DB2 data types from data source data types. These mappings are used when you create a nickname on a data source object.
SYSCAT.USEROPTIONS	Contains user authorization information that you set when you create user mappings between the federated database and the data source servers.
SYSCAT.VIEWS	Contains information about local federated views that you create.
SYSCAT.WRAPOPTIONS	Contains information about option values that you have set for a wrapper.
SYSCAT.WRAPPERS	Contains the name of the wrapper and library file for each data source that you create a wrapper for.

The following table lists the SYSSTAT views which contain federated information. These are read-write views that contain statistics you can update.

Table 129. Federated updatable global catalog views

Catalog views	Description
SYSSTAT.COLUMNS	Contains statistical information about each column in the data source objects (tables and views) that you have created nicknames for. Statistics are not recorded for inherited columns of typed tables.
SYSSTAT.INDEXES	Contains statistical information about each index specification for data source objects.
SYSSTAT.ROUTINES	Contains statistical information about each user-defined function. Does not include built-in functions. Statistics are not recorded for inherited columns of typed tables.
SYSSTAT.TABLES	Contains information about each base table. View, synonym, and alias information is not included in this view. For typed tables, only the root table of a table hierarchy is included in the view. Statistics are not recorded for inherited columns of typed tables.

Nickname column options for federated systems

You can specify column information in the CREATE NICKNAME or ALTER NICKNAME statements using parameters called nickname column options.

The following table lists the nickname column options for data source. Table two contains a complete listing of nickname column options.

Table 130. Nickname column options for relational data sources

Data source	DOCUMENT	ESCAPE_ INPUT	FOREIGN_ KEY	NUMERIC_ STRING	PRIMARY_ KEY	TEMPLATE	XPATH
DB2 UDB for iSeries				X			
DB2 UDB for z/OS and OS/390				X			
DB2 for VM and VSE				X			
DB2 Version 9.1 for Linux, UNIX, and Windows				X			
Informix				X			
Microsoft SQL Server				X			
ODBC				X			
OLE DB							
Oracle				X			
Sybase				X			
Teradata				X			

Table 131. Nickname column options for nonrelational data sources

Options	BLAST	Script	Table-structured files	WebSphere Business Integration	Web services	XML
DELIMITER	X					
DOCUMENT			X			X

Table 131. Nickname column options for nonrelational data sources (continued)

Options	BLAST	Script	Table-structured files	WebSphere Business Integration	Web services	XML
ESCAPE_INPUT				X	X	
FINAL_XDROPOFF	X					
FOREIGN_KEY				X	X	X
INDEX	X					
INPUT_MODE		X				
MASK_LOWER_CASE	X					
POSITION		X				
PRIMARY_KEY				X	X	X
QUERY_GENETIC_CODE	X					
SWITCH		X				
SWITCH_ONLY		X				
TEMPLATE				X	X	
VALID_VALUES		X				
XDROPOFF_GAPPED	X					
XDROPOFF_UNGAPPED	X					
XPATH				X	X	X

Table 132. Column options and their settings

Option	Description and valid settings	Default setting
DEFAULT	Specifies a new default value for the following input fixed columns: • E_value • QueryStrands • GapAlign • NMisMatchPenalty • NMatchReward • Matrix • FilterSequence • NumberOfAlignments • GapCost • ExtendedGapCost • WordSize • ThresholdEx This new value overrides the preset default values. The new default value must be of the same type as the indicated value for a given column.	

Table 132. Column options and their settings (continued)

Option	Description and valid settings	Default setting
DELIMITER	The delimiter characters to be used to determine the end point of the definition line information for the column on which this option appears. If more than one character appears in this option's value, then the first occurrence of any one of the characters signals the end of this field's information. The default is end of line. This option is required, unless you want the last specified column to contain the remainder of the definition line.	The default delimiter is end of line.

Table 132. Column options and their settings (continued)

Option	Description and valid settings	Default setting
DOCUMENT	For table-structured files: Specifies the kind of table-structured file. This wrapper supports only the value FILE for this option. Only one column can be specified with the DOCUMENT option per nickname. The column that is associated with the DOCUMENT option must be a data type of VARCHAR or CHAR. Using the DOCUMENT nickname column option instead of the FILE_PATH nickname option implies that the file that corresponds to this nickname will be supplied when the query runs. If the DOCUMENT option has the FILE value, the value that is supplied when the query runs is the full path of the file whose schema matches the nickname definition for this nickname. For XML: Specifies that this column is a DOCUMENT column. The value of the DOCUMENT column indicates the type of XML source data that is supplied to the nickname when the query runs. This option is accepted only for columns of the root nickname (the nickname that identifies the elements at the top level of the XML document). Only one column can be specified with the DOCUMENT option per nickname. The column that is associated with the DOCUMENT option must be a VARCHAR data type. If you use a DOCUMENT column option instead of the FILE_PATH or DIRECTORY_PATH nickname option the document that corresponds to this nickname is supplied when the query runs. The valid values for the DOCUMENT option are: **FILE** Specifies that the value of the nickname column is bound to the path name of a file. The data from this file is supplied when the query runs. **DIRECTORY** Specifies that the value of the nickname column is bound to the path name of a directory that contains multiple XML data files. The XML data from multiple files is supplied when the query runs. The data is in XML files in the specified directory path. The XML wrapper uses only the files with an .xml extension that are located in the directory that you specify. The XML wrapper ignores all other files in this directory. **URI** Specifies that the value of the nickname column is bound to the path name of a remote XML file to which a URI refers. The URI address indicates the remote location of this XML file on the Web. The URI can contain a colon-separated IPv6 address if it is enclosed in square brackets (for example, http://[1080:0:0:0:8:800:200C:417A]). **COLUMN** Specifies that the XML document is stored in a relational column.	

Table 132. Column options and their settings (continued)

Option	Description and valid settings	Default setting
ELEMENT_NAME	Specifies the BioRS element name. The case sensitivity of this name depends on the case sensitivity of the BioRS server and on the value of the CASE_SENSITIVE server option. You must specify the BioRS element name only if it is different from the column name.	
ESCAPE_INPUT	Specifies whether XML special characters are replaced in XML input values or not. Use this option to include XML fragments as input, such as XML fragments with repeating elements. The TEMPLATE column option must be defined on columns that use the ESCAPE_INPUT column option. The column data type must be VARCHAR or CHAR. Valid values are: **Y** If the XML input contains special characters these are replaced with their counterpart characters that XML uses to represent the input characters. **N** Input characters are preserved exactly as they appear.	Y
FINAL_XDROPOFF	The X dropoff value for the final gapped alignment, measured in bits. The value 0 invokes the default behavior.	50 bits for blastn and megablast queries. 0 bits for tblastx queries. 25 bits (INTEGER data types) for all other query types.
FOREIGN_KEY	Indicates that this nickname is a child nickname and specifies the name of the corresponding parent nickname. A nickname can have at most one FOREIGN_KEY column option. The value for this option is case sensitive. The table that is designated with this option holds a key that is generated by the wrapper. The XPATH option must not be specified for this column. The column can be used only to join parent nicknames and child nicknames. A CREATE NICKNAME statement with a FOREIGN_KEY option will fail if the parent nickname has a different schema name. Unless the nickname that is referred to in a FOREIGN_KEY clause was explicitly defined as lowercase or mixed case by enclosing it in quotation marks in the corresponding CREATE NICKNAME statement, then when you refer to this nickname in the FOREIGN_KEY clause, you must specify the nickname in uppercase. When this option is set on a column, no other option can be set on the column.	
INDEX	The ordinal number of the column on which this option appears in the group of definition line columns. This option is required.	
INPUT_MODE	Specifies the input mode for a column. Valid values are CONFIG or FILE_INPUT. The wrapper passes the specified value to the script daemon.	

Table 132. Column options and their settings (continued)

Option	Description and valid settings	Default setting
IS_INDEXED	Indicates whether the corresponding column is indexed (whether the column can be referenced in a predicate). The valid values are Y and N. The value Y can be specified only for columns whose corresponding element is indexed by the BioRS server.	When a nickname is created, this option is automatically added with the value Y to any columns that correspond to a BioRS indexed element.
MASK_LOWER_CASE	Use lowercase filtering with a FASTA sequence.	
NUMERIC_STRING	Specifies whether a column contains strings of numeric characters. **Y** This column contains strings of numeric characters ’0’, ’1’, ’2’, ... ’9’. It does not contain blanks. If this column contains only numeric strings followed by trailing blanks, do not specify Y. When you set NUMERIC_STRING to Y for a column, you are informing the optimizer that this column contains no blanks that could interfere with sorting of the column’s data. Use this option when the collating sequence of a data source is different from the collating sequence that the federated server uses. Columns that use this option are not excluded from remote evaluation because of a different collating sequence. **N** This column is either not a numeric string column or is a numeric string column that contains blanks.	N
POSITION	An integer value for positional parameters. This option applies only to input columns. If the positional value is set to an integer, then this input must be in this position in the command line. If this option is set, the switch is inserted into the appropriate location when the query is run. If POSITION is set to -1, the option is added as the last command line option. POSITION integer values cannot be duplicated in a nickname. This option is not required.	
PRIMARY_KEY	Indicates that this nickname is a parent nickname. The column data type must be VARCHAR(16). A nickname can have at most one PRIMARY_KEY column option. YES is the only valid value. The column that is designated with this option holds a key that is generated by the wrapper. The XPATH option must not be specified for this column. The column can be used only to join parent nicknames and child nicknames. When this option is set on a column, no other option can be set on the column.	
QUERY_GENETIC_CODE	Query genetic code uses default = 1.	
REFERENCED_OBJECT	This option is valid only for columns whose BioRS data type is Reference. This option specifies the name of the BioRS databank that is referenced by the current column. The case sensitivity of this name depends on the case sensitivity of the BioRS server and on the value of the CASE_SENSITIVE server option.	

Table 132. Column options and their settings (continued)

Option	Description and valid settings	Default setting
SOAPACTIONCOLUMN	A column to dynamically specify the URI SOAPACTION attribute from the Web Service Description Language (WSDL) format. This option is specified on only the root nickname. When this option is set on a column, no other option can be set on the column. The URL can contain a colon-separated IPv6 address if it is enclosed in square brackets (for example, http://[1080:0:0:0:8:800:200C:417A]).	
SWITCH	A character string to specify a parameter for the script on the command line. This option applies only to input columns.	
SWITCH_ONLY	Enables the use of switches without a command line argument. If the SWITCH_ONLY option is specified with a value of Y, then valid input values are Y or N. For an input value of Y, only the switch is added to the command line. For an input value of N, no value is added to the command line.	
TEMPLATE	The column template fragment to use to construct the XML input document. The fragment must conform to the specified template syntax.	
URLCOLUMN	A column to dynamically specify the URL for the Web service endpoint when you run a query. This option is specified on only the root nickname. When this option is set on a column, no other option can be set on the column. The URL can contain a colon-separated IPv6 address if it is enclosed in square brackets (for example, http://[1080:0:0:0:8:800:200C:417A]).	
VALID_VALUES	A semicolon-separated set of valid values for a column.	

Table 132. Column options and their settings (continued)

Option	Description and valid settings	Default setting
VARCHAR_NO_TRAILING_BLANKS	This option applies to data sources that have variable character data types that do not pad the length with trailing blanks during comparison. Some data sources, such as Oracle, do not have blank-padded character comparison semantics that return the same results as the DB2 for Linux, UNIX, and Windows comparison semantics. Set this option when you want it to apply only to a specific VARCHAR or VARCHAR2 column in a data source object. **Y** Trailing blanks are absent from these VARCHAR columns, or the data source has blank-padded character comparison semantics that are similar to the semantics on the federated server. The federated server sends character comparison operations to the data source for processing. **N** Trailing blanks are present in these VARCHAR columns, and the data source has blank-padded character comparison semantics that are different than the federated server. The federated server processes character comparison operations if it is not possible to compensate for equivalent semantics. For example, rewriting the predicate.	N for affected data sources
XDROPOFF_GAPPED	The X dropoff value for gapped alignment, measured in bits. The value 0 invokes the default behavior.	30 bits for blastn queries. 20 bits for megablast queries. 15 bits INTEGER data types) for all other query types.
XDROPOFF_UNGAPPED	The X dropoff value for ungapped extension measured in bits. The value 0.0 invokes the default behavior. For blastn queries, the default is 20 bits. For megablast queries, the default is 10 bits. For all other query types, the default is 7 bits (REAL data types).	20 bits for blastn queries. 10 bits for megablast queries. 7 bits (REAL data types) for all other query types.
XPATH	Specifies the XPath expression in the XML document that contains the data that corresponds to this column. The wrapper evaluates the XPath expression after the CREATE NICKNAME statement applies this XPath expression from this XPATH nickname option.	

Function mapping options for federated systems

The primary purpose of function mapping options, is to provide information about the potential cost of executing a data source function at the data source.

WebSphere Federation Server supplies default mappings between existing built-in data source functions and built-in DB2 functions. For most data sources, the default function mappings are in the wrappers. To use a data source function that the federated server does not recognize, you must create a function mapping between a data source function and a counterpart function at the federated database.

Pushdown analysis determines if a function at the data source is able to execute a function in a query. The query optimizer decides if pushing down the function processing to the data source is the least cost alternative.

The statistical information provided in the function mapping definition helps the query optimizer compare the estimated cost of executing the data source function with the estimated cost of executing the DB2 function.

Table 133. Function mapping options and their settings

Option	Valid settings	Default setting
DISABLE	Disable a default function mapping. Valid values are ′Y′ and ′N′.	′N′
INITIAL_INSTS	Estimated number of instructions processed the first and last time that the data source function is invoked.	′0′
INITIAL_IOS	Estimated number of I/Os performed the first and last time that the data source function is invoked.	′0′
IOS_PER_ARGBYTE	Estimated number of I/Os expended for each byte of the argument set that′s passed to the data source function.	′0′
IOS_PER_INVOC	Estimated number of I/Os per invocation of a data source function.	′0′
INSTS_PER_ARGBYTE	Estimated number of instructions processed for each byte of the argument set that′s passed to the data source function.	′0′
INSTS_PER_INVOC	Estimated number of instructions processed per invocation of the data source function.	′450′
PERCENT_ARGBYTES	Estimated average percent of input argument bytes that the data source function will actually read.	′100′
REMOTE_NAME	Name of the data source function.	local name

Nickname options for federated systems

Some nickname options are required and cannot be dropped. Other nickname options cannot be added if specific nickname options are already set.

Table 134 on page 446 list the nickname options for each data source. Table 135 on page 446 describes each nickname option and lists the valid and default values.

Table 134. Available nickname options

Data source	BUSOBJ_NAME	COLUMN_DELIMITER	DATASOURCE	FILE_PATH	NAMESPACES	REMOTE_OBJECT	REQUIRE_PREDICATE	RESULTSET_SIZE	STREAMING	TEMPLATE	TIMEOUT	VALIDATE	XPATH
BioRS						X							
BLAST			X								X		
Entrez						X							
Excel				X									
HMMER			X								X		
Script			X		X				X		X	X	X
Table- structured files		X		X									
Web services					X				X	X			X
WebSphere Business Integration	X				X		X	X		X			X
XML				X	X				X				X

Table 135 describes each nickname option and lists the valid and default settings.

Table 135. Nickname options and their settings

Option	Description and valid settings	Default setting
BUSOBJ_NAME	The name of the XML schema definition file (.xsd) that represents the business object. For example sap_bapi_customer_get_detail2 . This option must be specified in a parent nickname.	
COLUMN_ DELIMITER	The delimiter that is used to separate columns of a table-structured file, enclosed in single quotation marks. The column delimiter can be more than one character in length. If no column delimiter is defined, the default delimiter is a comma. A single quotation mark cannot be used as a delimiter. The column delimiter must be consistent throughout the file. A null value is represented by two delimiters next to each other or a delimiter followed by a line terminator, if the NULL field is the last one on the line. The column delimiter cannot exist as valid data for a column.	The default delimiter is a comma.

Table 135. Nickname options and their settings (continued)

Option	Description and valid settings	Default setting
DATASOURCE	For BLAST: The name of the data source on which the BLAST search will run. The same string that is used here must be present in the configuration file of the BLAST daemon. This option is required. For HMMER (type PFAM): The name of the HMM Profile database that is to be searched by HMMPFAM. The same string that is used here must be present in the configuration file of the HMMER daeamon. This option is required. For HMMER (type SEARCH): The name of the sequence file that is to be searched by HMMSEARCH. The same string that is used here must be present in the configuration file of the HMMER daeamon. This option is required.	
DIRECTORY_PATH	Specifies the path name of a directory that contains one or more XML files. Use this option to create a single nickname over multiple XML source files. The XML wrapper uses only the files with an .xml extension that are located in the directory that you specify. The XML wrapper ignores all other files in this directory. If you specify this nickname option, do not specify a DOCUMENT column. This option is accepted only for the root nickname (the nickname that identifies the elements at the top level of the XML document).	
FILE_PATH	For Microsoft Excel: Specifies the fully qualified directory path and file name of the Excel spreadsheet that you want to access. This option is required. For table-structured files: The fully qualified path to the table-structured file to be accessed, enclosed in single quotation marks. The data file must be a standard file or a symbolic link, rather then a pipe or another non-standard file type. Either the FILE_PATH or the DOCUMENT nickname column option must be specified. If the FILE_PATH nickname option is specified, then no DOCUMENT nickname column option can be specified. For XML: Specifies the file path of the XML document. If you specify this nickname option, do not specify a DOCUMENT column. This option is accepted only for the root nickname (the nickname that identifies the elements at the top level of the XML document).	
HMMTYPE	Optional: The alphabet that is used in both models and gene sequences. The value can be either NUCLEIC or PROTEIN and is not case sensitive.	PROTEIN
INSTANCE_ PARSE_TIME	Specifies the time (in milliseconds) to parse the data in one row of the XML source document. You can modify the INSTANCE_PARSE_TIME, XPATH_EVAL_TIME, and NEXT_TIME options to optimize queries of large or complex XML source structures. This option is accepted only for columns of the root nickname (the nickname that identifies the elements at the top level of the XML document). The number that you specify can be an integer or a decimal value.	7
KEY_COLUMN	The name of the column in the file that forms the key on which the file is sorted, enclosed in single quotation marks. Use this option for sorted files only. A column that is designated with the DOCUMENT nickname column option must not be specified as the key column. Only single-column keys are supported. Multi-column keys are not allowed. The value must be the name of a column that is defined in the CREATE NICKNAME statement. The column must be sorted in ascending order. The key column must be designated not nullable by adding the NOT NULL option to its definition in the nickname statement. This option is case-sensitive. However, DB2 changes column names to uppercase unless the column is defined with double quotation marks.	If the value is not specified for a sorted nickname, the value is the name of the first column in the nicknamed file.

Table 135. Nickname options and their settings (continued)

Option	Description and valid settings	Default setting

NAMESPACES	The namespaces that are associated with the namespace prefixes that is used in the XPATH and TEMPLATE options for each column. The syntax is:
	`NAMESPACES 'prefix1=`
	`"actual_namespace1",`
	`prefix2="actual_namespace2" '`
	Separate each namespace with a comma. For example:
	`NAMESPACES '`
	`c="http://www.myweb.com/cust",`
	`i="http://www.myweb.com/cust/id",`
	`n="http://www.myweb.com/cust/name"'`	

NEXT_TIME	Specifies the time (in milliseconds) that is required to locate subsequent source elements from the XPath expression. You can modify the NEXT_TIME, XPATH_EVAL_TIME, and INSTANCE_PARSE_TIME options to optimize queries of large or complex XML source structures. This option is accepted for root nicknames and non-root nicknames.	1
PARENT	Specified only for a child nickname whose parent was renamed through the REMOTE_OBJECT option. The PARENT option associates a child with a parent when multiple nickname families are defined within a DB2 schema. This name is case-sensitive.	
PROCESSORS	Specifies the number of processors to be used when a BLAST query is evaluated. This option corresponds to the blastall -a option.	1
RANGE	Specifies a range of cells to be used in the data source.	
REMOTE_OBJECT	For BioRS: Specifies the name of the BioRS databank that is associated with the nickname. This name determines the schema and the BioRS databank for the nickname. This name also specifies the relationship of the nickname to other nicknames. The case sensitivity of this name depends on the case sensitivity of the BioRS server and on the value of the CASE_SENSITIVE server option. You cannot use the ALTER NICKNAME statement to change or delete this name. If the name of the BioRS databank that is used in this option changes, you must delete and then create the entire nickname again. For Entrez and OMIM: Specifies the name of the object type that is associated with the nickname. This name determines the schema and NCBI database for the nickname and its relationship to other nicknames. This name is case insensitive.	
REQUIRE_ PREDICATE	Specify Y to require an equality predicate on at least one input column in all queries on the nickname hierarchy, which can limit the size of the result set. If you know that the size of the result set that returns on a query with no predicate will not exceed the JVM memory limit, you can set the value of the REQUIRE_PREDICATE nickname option to N.	
RESULTSET_SIZE	Specifies how many business objects the adapter should return to the wrapper. Specify any other value to have the adapter return the specified number of business objects. You must enable result sets in the wrapper (RESULTSET_ENABLED set to Yes) for the RESULTSET_SIZE option to work. If you specify a nonzero value for RESULTSET_SIZE, an incomplete result might be returned. Any rows that exceed the specified number are discarded, and the wrapper issues a warning message that indicates that an incomplete result set was returned to the application.	0, which specifies that all business objects that match the query should be returned.
SOAPACTION	The URI SOAPACTION attribute from the Web Service Description Language (WSDL) format. This option is required for the root nickname. This option is not allowed with nonroot nicknames. The URL can contain a colon-separated IPv6 address if it is enclosed in square brackets (for example, http://[1080:0:0:0:8:800:200C:417A])	

Table 135. Nickname options and their settings (continued)

Option	Description and valid settings	Default setting
SORTED	Specifies whether the data source file is sorted or unsorted. This option accepts either Y, y, n, or N. Sorted data sources must be sorted in ascending order according to the collation sequence for the current locale, as defined by the settings in the LC_COLLATE National Language Support category. If you specify that the data source is sorted, set the VALIDATE_DATA_FILE option to Y.	N
STREAMING	Specifies whether the XML source document should be separated into logical fragments for processing. The fragments correspond to the node that matches the XPath expression of the nickname. The wrapper then parses and processes the XML source data fragment by fragment. This type of parsing minimizes memory usage. This option is specified on only the root nickname. You can specify streaming for any XML source document (FILE, DIRECTORY, URI, or COLUMN). This option is accepted only for columns of the root nickname (the nickname that identifies the elements at the top level of the XML document). Valid values are: **Y** Streaming mode is enabled. **N** Streaming mode is disabled. Do not set the STREAMING parameter to YES if you set the VALIDATE parameter to YES. If you set both parameters to YES, you will receive an error message.	N
TEMPLATE	For WebSphere Business Integration: The nickname template fragment to use to construct an XML input document. The fragment must conform to the specified template syntax. For Web Services: The nickname template fragment to use to construct a SOAP request. The fragment must conform to the specified template syntax.	
TIMEOUT	For BLAST and HMMER: The maximum time, in minutes, that the wrapper waits for results from the daemon.	For BLAST and HMMER: 60.
URL	The URL for the Web service endpoint. This option is required for the root nickname. This option is not allowed with nonroot nicknames. Supported protocols are HTTP and HTTPS. The URL can contain a colon-separated IPv6 address if it is enclosed in square brackets (for example, http://[1080:0:0:0:8:800:200C:417A]).	
VALIDATE	Specifies whether the XML source document is validated before the XML data is extracted. If this option is set to YES, the nickname option verifies that the structure of the source document conforms to an XML schema or to a document type definition (DTD). This option is accepted only for columns of the root nickname (the nickname that identifies the elements at the top level of the XML document). The XML source document is not validated if the XML wrapper cannot locate the XML schema file or DTD file (.xsd or .dtd). DB2 does not issue an error message if the validation does not occur. Therefore, ensure that the XML schema file or DTD file exists in the location that is specified in the XML source document. Do not set the VALIDATE parameter to YES if you set the STREAMING parameter to YES. If you set both parameters to YES, you will receive an error message.	NO

Table 135. Nickname options and their settings (continued)

Option	Description and valid settings	Default setting
VALIDATE_ DATA_FILE	For sorted files, this option specifies whether the wrapper verifies that the key column is sorted in ascending order and checks for null keys. The only valid values for this option are Y or N. The check is done once at registration time. This option is not allowed if the DOCUMENT nickname column option is used for the file path.	N
XPATH	Specifies the XPATH expression that identifies the elements that represent the individual tuples. The XPATH nickname option for a child nickname is evaluated in the context of the path that is specified by the XPATH nickname option of its parent. This XPATH expression is used as a context for evaluating column values that are identified by the XPATH nickname column options.	
XPATH_EVAL_ TIME	Specifies the time (in milliseconds) to evaluate the XPath expression of the nickname and to locate the first element. You can modify the XPATH_EVAL_TIME, INSTANCE_PARSE_TIME, and NEXT_TIME options to optimize queries of large or complex XML source structures. This option is accepted for root nicknames and nonroot nicknames. The number that you specify can be an integer or a decimal value.	1

Server options for federated systems

Server options are used to describe a data source server.

Server options specify data integrity, location, security, and performance information. Some server options are available for all data sources, and other server options are data source specific.

The common federated server options for relational data sources are:

- Compatibility options. COLLATING_SEQUENCE, IGNORE_UDT
- Data integrity options. IUD_APP_SVPT_ENFORCE
- Data and time options. DATEFORMAT, TIMEFORMAT, TIMESTAMPFORMAT
- Location options. CONNECTSTRING, DBNAME, IFILE
- Security options. FOLD_ID, FOLD_PW, INFORMIX_LOCK_MODE
- Performance options. COMM_RATE, CPU_RATIO, DB2_MAXIMAL_PUSHDOWN, IO_RATIO, LOGIN_TIMEOUT, PACKET_SIZE, PLAN_HINTS, PUSHDOWN, TIMEOUT, VARCHAR_NO_TRAILING_BLANKS

The following table lists the server definition server options applicable for each relational data source.

Table 136. Server options for relational data sources

Data Source	CODEPAGE	COLLATING_SEQUENCE	COMM_RATE	CONNECTSTRING	CONV_EMPTY_STRING	CPU_RATIO	DATEFORMAT	DB2_MAXIMAL_PUSHDOWN	DB2_PRESERVE_CUR_ON_CONNECTION	DB2_UM_PLUGIN	DB2_TWO_PHASE_COMMIT	DBNAME	FOLD_ID	FOLD_PW	IFILE	INFORMIX_CLIENT_LOCALE	INFORMIX_DB MODE	INFORMIX_LOCK MODE	IO_RATIO	IUD_APP_SVPT_ENFORCE	LOGIN_TIMEOUT	NODE	PACKET_SIZE	PASSWORD	PLAN_HINTS	PUSHDOWN	TIMEOUT	TIMEFORMAT	TIMESTAMPFORMAT	VARCHAR_NO_TRAILING_BLANKS
DB2 UDB for iSeries		X	X			X		X		X	X	X	X	X					X	X				X		X				X
DB2 UDB for z/OS and OS/390		X	X			X		X		X	X	X	X	X					X	X				X		X				X
DB2 for VM and VSE		X	X			X		X		X		X	X	X					X	X				X		X				X
DB2 for Linux, UNIX, and Windows		X	X			X		X		X	X	X	X	X					X	X				X		X				X
Informix		X	X			X		X		X	X	X	X	X	X	X		X	X	X		X		X		X				
Microsoft SQL Server	X	X	X			X		X	X	X	X	X	X	X					X	X		X		X		X				
ODBC	X	X	X			X	X	X		X		X	X	X					X	X		X		X		X		X	X	X
OLE DB		X		X						X																				
Oracle		X	X			X		X		X	X		X	X					X			X		X	X	X				X
Sybase		X	X		X	X		X		X	X	X	X	X	X				X		X	X	X	X	X	X	X			
Teradata		X	X			X		X		X									X	X		X				X				

The following table lists the server definition server options applicable for each nonrelational data source, except WebSphere Business Integration. The server definition server options for WebSphere Business Integration are listed in Table 138 on page 453.

Table 137. Server options for nonrelational data sources.

Data Source	DAEMON_PORT	DB2_UM_PLUGIN	MAX_ROWS	NODE	PROXY_AUTHID	PROXY_PASSWORD	PROXY_SERVER_NAME	PROXY_SERVER_PORT	PROXY_TYPE	SSL_CLIET_CERTIFICATE_LABEL	SSL_KEYSTORE_FILE	SSL_KEYSTORE_PASSWORD	SSL_VERIFY_SERVER_CERTIFICATE	SOCKET_TIMEOUT	TIMEOUT	USE_CLOB_SEQUENCE
BioRS		X		X					X						X	
BLAST	X			X	X	X	X	X	X	X	X	X	X			X
Entrez			X		X	X	X	X	X					X		
Excel																
HMMER	X			X	X	X	X	X	X	X	X	X	X			X
SCRIPT	X			X	X	X	X	X	X	X	X	X	X			
Table-structured files																
Web services		X								X	X	X	X		X	
XML					X	X	X	X	X	X	X	X	X	X		

The following table lists the server definition server options applicable for WebSphere Business Integration data sources.

Table 138. Server options for WebSphere Business Integration data sources.

Data Source	APP_TYPE	FAULT_QUEUE	MQ_CONN_NAME	MQ_MANAGER	MQ_RESPONSE_TIMEOUT	MQ_SVRCONN_CHANNELNAME	REQUEST_QUEUE	RESPONSE_QUEUE
WebSphere Business Integration	X	X	X	X	X	X	X	X

The following table describes each server option and lists the valid and default settings.

Table 139. Server options and their settings

Option	Description and valid settings	Default setting
APP_TYPE	The type of remote application. Valid values are ’PSOFT’, ’SAP’, and ’SIEBEL’. This option is required.	None.

Table 139. Server options and their settings (continued)

Option	Description and valid settings	Default setting
CASE_SENSITIVE	Specifies whether the BioRS server treats names in a case sensitive manner. Valid values are Y or N. **’Y’** The BioRS server treats names in a case sensitive manner. **’N’** The BioRS server does not treat names in a case sensitive manner In the BioRS product, a configuration parameter controls the case sensitivity of the data that is stored on the BioRS server. The CASE_SENSITIVE option is the federated server counterpart to that BioRS system configuration parameter. You must synchronize the BioRS server case sensitivity configuration settings in your BioRS system and in the federated server. If you do not keep the case sensitivity configuration settings synchronized between BioRS and the federated server, errors will occur when you attempt to access BioRS data through federated server. You cannot change or delete the CASE_SENSITIVE option after you create a new BioRS server in federated server. If you need to change the CASE_SENSITIVE option, you must drop and then create the entire server again. If you drop the BioRS server, you must also create all of the corresponding BioRS nicknames again. Federated server automatically drops all nicknames that correspond to a dropped server.	Y
CODEPAGE	Specifies the DB2 code page identifier corresponding to the coded character set of the data source client configuration. You must specify the client’s code page if the client’s code page and the federated database code page do not match. For data sources that support Unicode, the CODEPAGE option can be set to the DB2 code page identifier corresponding to the supported Unicode encoding of the data source client.	On UNIX or Windows systems with a non-Unicode federated database: The federated database code page. On UNIX systems with a Unicode federated database: 1208 On Windows systems with a Unicode federated database: 1202

Table 139. Server options and their settings (continued)

Option	Description and valid settings	Default setting
COLLATING_ SEQUENCE	Specifies whether the data source uses the same default collating sequence as the federated database, based on the NLS code set and the country/region information. **’Y’** The data source has the same collating sequence as the DB2 federated database. **’N’** The data source has a different collating sequence than the DB2 federated database collating sequence. **’I’** The data source has a different collating sequence than the DB2 federated database collating sequence, and the data source collating sequence is insensitive to case (for example, ’STEWART’ and ’StewART’ are considered equal).	’N’
COMM_RATE	Specifies the communication rate between the federated server and the data source server. Expressed in megabytes per second. Valid values are greater than 0 and less than $1x10^{23}$. Values can be expressed in any valid REAL notation.	’2’
CONNECTSTRING	Specifies initialization properties needed to connect to an OLE DB provider.	None.
CONNECTSTRING	Specifies initialization properties needed to connect to an OLE DB provider.	None.
CONV_EMPTY_STRING	Use for Sybase wrapper that works with replication tasks. When you set the CONV_EMPTY_STRING option into Y, the Sybase wrapper converts an empty string into a space. Set this option to Y when a source data server has a non-nullable character column that stores an empty string and the target data server is Sybase.	N
CPU_RATIO	Indicates how much faster or slower a data source CPU runs than the federated server CPU. Valid values are greater than 0 and less than $1x10^{23}$. Values can be expressed in any valid REAL notation. A setting of 1 indicates that the DB2 federated CPU speed and the data source CPU speed have the same CPU speed, a 1:1 ratio. A setting of 0.5 indicates that the DB2 federated CPU speed is 50% slower than the data source CPU speed. A setting of 2 indicates that the DB2 federated CPU speed is twice as fast as the data source CPU speed.	’1.0’
DATEFORMAT	The date format used by the data source. Enter the format using ’DD’, ’MM’, and ’YY’ or ’YYYY’ to represent the numeric form of the date. You should also specify the delimiter such as a space or comma. For example, to represent the date format for ’2003-01-01’, use ’YYYY-MM-DD’. This field is nullable.	None.

Table 139. Server options and their settings (continued)

Option	Description and valid settings	Default setting
DAEMON_PORT	Specifies the port number on which the daemon will listen for BLAST or HMMER job requests. The port number must be the same number specified in the DAEMON_PORT option of the daemon configuration file.	BLAST: 4007 HMMER: 4098
DB2_MAXIMAL_ PUSHDOWN	Specifies the primary criteria that the query optimizer uses when choosing an access plan. The query optimizer can choose access plans based on cost or based on the user requirement that as much query processing as possible be performed by the remote data sources. **’Y’** The query optimizer chooses an access plan that pushes down more query operations to the data source than other plans. When several access plans provide the same amount of pushdown, the query optimizer then chooses the plan with the lowest cost. If a materialized query table (MQT) on the federated server can process part or all of the query, then an access plan that includes the materialized query table might be used. The federated database does not push down queries that result in a Cartesian product. **’N’** The query optimizer chooses an access plan based on cost.	’N’
DB2_PRESERVE_CUR_ON_ CONNECTION	Specifies the behavior of cursors for committed or rolled back transactions. If value is set to Y, cursors can remain open on Microsoft SQL Server even if COMMIT or ROLLBACK is sent. If this option is not set or the value is set to N, cursors are closed if COMMIT or ROLLBACK is sent. This option is optional.	None
DB2_TWO_PHASE_COMMIT	Specify to allow or disallow federated two-phase commit for each data source using the CREATE SERVER or ALTER SERVER statements. Set to Y to configure and activate two-phase commit at the database level. Set to N to configure but not activate federated two-phase commit at the database level. Applications can activate or deactivate this option with the SET SERVER OPTION statement.	
DB2_UM_PLUGIN	If you use a plugin for retrieving user mappings from an external repository, specify the class name including package (for example, ’com.ibm.ii.um.ldap.UserMappingRepository’). If you are using the Lightweight Directory Access Protocol (LDAP) sample plugin, you can use just the class name (for example, UserMappingRepository).	None.

Table 139. Server options and their settings (continued)

Option	Description and valid settings	Default setting
DBNAME	Name of the data source database that you want the federated server to access. For DB2 database, this value corresponds to a specific database for the initial remote DB2 database connection. This specific database is the database alias for the remote DB2 database that is cataloged at the federated server using the CATALOG DATABASE command or the DB2 Configuration Assistant. Does not apply to Oracle data sources because Oracle instances contain only one database. Does not apply to Teradata.	None.
FAULT_QUEUE	The name of the fault queue that delivers error messages from the adapter to the wrapper. The name must conform to the specifications for queue names for WebSphere MQ. This is a required option.	None.
FOLD_ID (See notes 1 and 4 at the end of this table.)	Applies to user IDs that the federated server sends to the data source server for authentication. Valid values are: **'U'** The federated server folds the user ID to uppercase before sending it to the data source. This is a logical choice for DB2 family and Oracle data sources (See note 2 at end of this table.) **'N'** The federated server does nothing to the user ID before sending it to the data source. (See note 2 at end of this table.) **'L'** The federated server folds the user ID to lowercase before sending it to the data source. If this option is not specified, the federated server tries to send the user ID to the data source in uppercase (unchanged) and in lowercase.	None.
FOLD_PW (See notes 1, 3 and 4 at the end of this table.)	Applies to passwords that the federated server sends to data sources for authentication. Valid values are: **'U'** The federated server folds the password to uppercase before sending it to the data source. This is a logical choice for DB2 family and Oracle data sources. **'N'** The federated server does nothing to the password before sending it to the data source. **'L'** The federated server folds the password to lowercase before sending it to the data source. If this option is not specified, the federated server tries to send the user ID to the data source in uppercase (unchanged) and in lowercase.	None.

Table 139. Server options and their settings (continued)

Option	Description and valid settings	Default setting
HMMPFAM_OPTIONS	Specifies hmmpfam options such as --null2, --pvm, and --xnu that have no corresponding column name in a reference table that maps options to column names. For example: `HMMPFAM_OPTIONS '--xnu --pvm'` In this example, the daemon runs the HMMPFAM program with options from the WHERE clause of the query, plus the additional options --xnu --pvm.	
HMMSEARCH_ OPTIONS	Allows the user to provide additional command line options to the hmmsearch command. Only valid with type SEARCH. See the HMMER User's Guide for more information.	None.
IFILE	Use to specify the path and name of the Sybase Open Client interfaces file if you do not want to use the default interface file. On Windows NT federated servers, the default is %SYBASE%\ini\sql.ini. On UNIX federated servers, the default is $SYBASE%/interfaces.	None.
INFORMIX_CLIENT_LOCALE	Specifies the CLIENT_LOCALE to use for the connection between the federated server and the data source server. If the INFORMIX_CLIENT_LOCALE option is not specified, the Informix CLIENT_LOCALE environment variable is set to the value specified in the db2dj.ini file (if any). If db2dj.ini does not specify CLIENT_LOCALE, the Informix CLIENT_LOCALE environment variable is set to the Informix locale that most closely matches the code page and territory of the federated database. Any valid Informix locale is a valid value. This option is optional.	None.
INFORMIX_DB_LOCALE	Specifies the DB_LOCALE to use for the connection between the federated server and the data source server. If the INFORMIX_DB_LOCALE option is not specified, the Informix DB_LOCALE environment variable is set to the value specified in the db2dj.ini file (if any). If db2dj.ini does not specify DB_LOCALE, the Informix DB_LOCALE environment variable is not set. Any valid Informix locale is a valid value. This option is optional.	None.

Table 139. Server options and their settings (continued)

Option	Description and valid settings	Default setting

INFORMIX_LOCK_ MODE	Specifies the lock mode to be set for an Informix data source. The Informix wrapper issues the ’SET LOCK MODE’ command immediately after establishing the connection to an Informix data source. Valid values are:
	’W’ Sets the Informix lock mode to WAIT. If the wrapper tries to access a locked table or row, Informix waits until the lock is released.
	’N’ Sets the Informix lock mode to NOWAIT. If the wrapper tries to access a locked table or row, Informix returns an error.
	’n’ Sets the Informix lock mode to WAIT *n* seconds. If the wrapper tries to access a locked table or row and the lock is not released within the specified number of seconds, Informix returns an error.
	If a deadlock or timeout error occurs when a federated server attempts to connect to an Informix data source, changing the lock mode setting on the federated server can often resolve the error. Use the ALTER SERVER statement to change the lock mode setting on the federated server.
	For example:
	`ALTER SERVER TYPE informix VERSION 9 WRAPPER`
	`    informix`
	`   OPTIONS (ADD informix_lock_mode '60')`
		’W’

IO_RATIO	Denotes how much faster or slower a data source I/O system runs than the federated server I/O system. Valid values are greater than 0 and less than $1x10^{23}$. Values can be expressed in any valid REAL notation. A setting of 1 indicates that the DB2 federated I/O speed and the data source I/O speed have the same I/O speed, a 1:1 ratio. A setting of .5 indicates that the DB2 federated I/O speed is 50% slower than the data source I/O speed. A setting of 2 indicates that the DB2 federated I/O speed is twice as fast as the data source I/O speed.	’1.0’

Table 139. Server options and their settings (continued)

Option	Description and valid settings	Default setting
IUD_APP_SVPT_ ENFORCE	Specifies whether the DB2 federated system should enforce detecting or building of application savepoint statements. When set using the SET SERVER OPTION statement, this server option will have no effect with static SQL statements. **'Y'** The federated server rolls back insert, update, or delete transactions if an error occurs in an insert, update, or delete operation and the data source does not enforce application savepoint statements. SQL error code SQL1476N is returned. **'N'** The federated server will not roll back transactions when an error is encountered. Your application must handle the error recovery.	'Y'
LOGIN_TIMEOUT	Specifies the number of seconds for the DB2 federated server to wait for a response from Sybase Open Client to the login request. If you specify 0, the federated server will wait indefinitely for a response.	'0'
MAX_ROWS	For Entrez: Specifies the number of rows that the federated server returns for a query. For OMIM: Limits to the number of records for the root nickname that a query can return. For example, if the MAX_ROWS server option is set to 25, a maximum of 25 records for the root nickname and all of the records for the child-related nicknames are returned. You can specify only positive numbers and zero. When you set the option to be zero, you enable queries to retrieve an unlimited number of rows from the NCBI Web site. However, setting the MAX_ROWS server option to zero or to a very high number can impact your query performance. The MAX_ROWS server option is not required.	Microsoft Windows operating systems: 2000 rows. UNIX-based operating systems: 5000 rows.
MQ_CONN_NAME	The hostname or network address of the computer where the Websphere MQ server is running. An example of a connection name is: 9.30.76.151(1420) where 1420 is the port number. If the port number is excluded a default value of 1414 will be used. This option is optional. If it is omitted, the MQSERVER environment variable (if sepcified in db2dj.ini file) is used to select the channel definition. If MQSERVER is not set, the client channel table is used.	The wrapper uses the MQSERVER environment variable, if specified in the db2dj.ini file, to select the channel definition. If the MQSERVER environment variable is not set, the wrapper uses the client channel table.
MQ_MANAGER	The name of the WebSphere MQ manager. Any valid WebSphere MQ manager name. This option is required.	None.

Table 139. Server options and their settings (continued)

Option	Description and valid settings	Default setting
MQ_RESPONSE_ TIMEOUT	The amount of time that the wrapper should wait for a response message from the response queue. The value is in milliseconds. You can specify a special value of -1 to indicate that there is no timeout period. This option is optional.	10000
MQ_SVRCONN_ CHANNELNAME	The name of the server-connection channel on the Webspehere MQ Manager that the wrapper should try to connect to. This parameter can be specified only if the MQ_CONN_NAME server option is specified. The default server-connection channel, SYSTEM.DEF.SVRCONN, is used if this option is omitted.	SYSTEM.DEF.SVRCONN
NODE	Relational data sources: Name by which a data source is defined as an instance to its RDBMS. BLAST: Specifies the host name or IP address of the system on which the BLAST daemon process is running. The IP address can be an IPv4 address (for example, 192.168.1.1) or it can be a colon-separated IPv6 address (for example, 1080:0:0:0:8:800:200C:417A). This option is required. HMMER: Specifies the host name or IP address of the server on which the HMMER daemon process runs. The IP address can be an IPv4 address or it can be a colon-separated IPv6 address. This option is required. BioRS: Specifies the host name of the system on which the BioRS query tool is available. The IP address can be an IPv4 address or it can be a colon-separated IPv6 address. This option is optional.	BioRS: localhost
PACKET_SIZE	Specifies the byte size of the packet that the Client-Library uses when sending special packets. If the Sybase wrapper needs to send or receive large amounts of text or image data, a larger packet size might improve efficiency.	
PASSWORD	Specifies whether passwords are sent to a data source. **'Y'** Passwords are sent to the data source and validated. **'N'** Passwords are not sent to the data source and not validated.	'Y'

Table 139. Server options and their settings (continued)

Option	Description and valid settings	Default setting
PLAN_HINTS	Specifies whether plan hints are to be enabled. Plan hints are statement fragments that provide extra information for data source optimizers. This information can, for certain query types, improve query performance. The plan hints can help the data source optimizer decide whether to use an index, which index to use, or which table join sequence to use. **'Y'** Plan hints are to be enabled at the data source if the data source supports plan hints. **'N'** Plan hints are not to be enabled at the data source. This option is only available for Oracle and Sybase data sources.	'N'
PORT	Specifies the number of the port the wrapper uses to connect to the BioRS server. This option is optional.	'5014'
PROCESSORS	Specifies the number of processors that the HMMER program uses. This option is equivalent to the --cpu option of the `hmmpfam` command.	None.
PROXY_AUTHID	Specifies the user name to use when the proxy server requires authentication. Contact your network administrator for the user name.	None.
PROXY_PASSWORD	Specifies the password to use when the proxy server requires authentication. Contact your network administrator for the password.	None.
PROXY_SERVER_ NAME	Specifies the proxy server name or the IP address. This field is required if the value of PROXY_TYPE is 'HTTP' or 'SOCKS'. Contact your network administrator for the server name or IP address of the proxy server. The IP address can be an IPv4 address (for example, 192.168.1.1) or it can be a colon-separated IPv6 address (for example, 1080:0:0:0:8:800:200C:417A).	None.
PROXY_SERVER_ PORT	Specifies the proxy server port number. This field is required if the value of PROXY_TYPE is 'HTTP' or 'SOCKS'. Contact your network administrator for the port number of the proxy server.	None.
PROXY_TYPE	Specifies the type of proxy type that is used to access the Internet when behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'. The default value is 'NONE'. Contact your network administrator for the type of proxy that is used. BLAST, HMMER, and SCRIPT do not support HTTP proxies.	'NONE'

Table 139. Server options and their settings (continued)

Option	Description and valid settings	Default setting
PUSHDOWN	**'Y'** DB2 will consider letting the data source evaluate operations. **'N'** DB2 will send the data source SQL statements that include only SELECT with column names. Predicates (such as WHERE=), column and scalar functions (such as MAX and MIN), sorts (such as ORDER BY or GROUP BY), and joins will not be included in any SQL sent to the data source. Default for the ODBC wrapper.	'Y'
RESPONSE_QUEUE	The name of the response queue that delivers query results from the adapter to the wrapper. The name must conform to the specifications for queue names for WebSphere MQ. This option is required.	None.
REQUEST_QUEUE	The name of the request queue that delivers query requests from the wrapper to the adapter. The name must conform to the specifications for queue names for WebSphere MQ. This option is required.	None.
SOCKET_TIMEOUT	Specifies the maximum time in minutes that the DB2 federated server will wait for results from the proxy server. A valid value is any number that is greater than or equal to zero. The default is zero '0'. A value of zero denotes an unlimited amount of time to wait.	0
SSL_CLIENT_CERTIFICATE_ LABEL	Specifies the client certificate that is sent during SSL authentication. If the value is not specified, the current DB2 authorization ID will be sent.	None.
SSL_KEYSTORE_FILE	Specifies the name of the certificate storage file to use for SSL/TLS communications. The value that you specify must be a full path that is accessible by the DB2 agent or FMP process. Optional: You can specify the value 'GSK_MS_CERTIFICATE_STORE' to use the native Microsoft certificate storage.	None.
SSL_KEYSTORE_PASWORD	Specifies the password that is used to access the SSL certificate storage. This password is encrypted when it is stored in the DB2 catalog.	None.
SSL_VERIFY_SERVER_ CERTIFICATE	Specifies if the server certificate should be verified during SSL authenticate. The values are not case sensitive. To authenticate, use one of the following values: 'Y', 'YES', 'T', or 'TRUE'. To disable, use 'F', 'FALSE', 'N', or 'NO'.	'NO'
TIMEFORMAT	The time format used by the data source. Enter the format using 'hh12', 'hh24', 'mm', 'ss', 'AM', or 'A.M'. For example, to represent the time format of '16:00:00', use 'hh24:mm:ss'. To represent the time format of '8:00:00 AM', use 'hh12:mm:ss AM'. This field is nullable.	None.

Table 139. Server options and their settings (continued)

Option	Description and valid settings	Default setting
TIMESTAMPFORMAT	The timestamp format used by the data source. The format follows that for date and time, plus ’n’ for tenth of a second, ’nn’ for hundredth of a second, ’nnn’ for milliseconds, and so on, up to ’nnnnnn’ for microseconds. For example, to represent the timestamp format of ’2003-01-01-24:00:00.000000’, use ’YYYY-MM-DD-hh24:mm:ss.nnnnnn’. This field is nullable.	None.
TIMEOUT	The timeout value that you specify depends on which wrapper that you are using. If you specify 0, DB2 will wait indefinitely for a response. Sybase: Specifies the number of seconds that the DB2 federated server will wait for a response from the Sybase server for any SQL statement. The value of *seconds* is a positive whole number. BioRS: Specifies the time, in minutes, that the BioRS wrapper should wait for a response from the BioRS server. The default value is 10. This option is optional. Web services: Specifies the time, in minutes, that DB2 should wait for a network transfer and the computation of a result.	Sybase: 0 BioRS: 10
USE_CLOB_ SEQUENCE	This option specifies the data type the federated server uses for the BlastSeq or HmmQSeq column. The values can be ’Y’ or ’N’. You can use the CREATE NICKNAME or ALTER NICKNAME statement. to override the default data type for the BlastSeq or HmmQSeq column. • If you specify a value of N, the data type is VARCHAR(32000). • If you specify a value of Y, the data type is CLOB(5M). The default value is N, not Y.	’Y’

Table 139. Server options and their settings (continued)

Option	Description and valid settings	Default setting
VARCHAR_NO_ TRAILING_BLANKS	This option applies to data sources which have variable character data types that do not pad the length with trailing blanks during comparison. Some data sources, such as Oracle, do not have blank-padded character comparison semantics that return the same results as the DB2 for Linux, UNIX, and Windows comparison semantics. Set this option when you want it to apply to all the VARCHAR and VARCHAR2 columns in the data source objects that will be accessed from the designated server. This includes views. **Y** Trailing blanks are absent from these VARCHAR columns, or the data source has blank-padded character comparison semantics that are similar to the semantics on the federated server. The federated server pushes down character comparison operations to the data source for processing. **N** Trailing blanks are present in these VARCHAR columns and the data source has blank-padded character comparison semantics that are different than the federated server. The federated server processes character comparison operations if it is not possible to compensate for equivalent semantics. For example, rewriting the predicate.	N for affected data sources.

Notes on this table:

1. This field is applied regardless of the value specified for authentication.
2. Because DB2 stores user IDs in uppercase, the values ’N’ and ’U’ are logically equivalent to each other.
3. The setting for FOLD_PW has no effect when the setting for password is ’N’. Because no password is sent, case cannot be a factor.
4. Avoid null settings for either of these options. A null setting can seem attractive because DB2 will make multiple attempts to resolve user IDs and passwords; however, performance might suffer (it is possible that DB2 will send a user ID and password up to nine times before successfully passing data source authentication).

Valid server types in SQL statements

Server types indicate the kind of data source that the server definition represents.

Server types vary by vendor, purpose, and operating system. Supported values depend on the wrapper being used.

For most data sources, you must specify a valid server type in the CREATE SERVER statement.

BioRS wrapper

A server type specification is optional for BioRS data sources.

Server Type	Data Source
Not required in the CREATE SERVER statement.	BioRS

BLAST wrapper

A server type specification is required for each type of BLAST search that you want to run for BLAST data sources supported by the BLAST daemon.

Server Type	Data Source
BLASTN	BLAST searches in which a nucleotide sequence is compared with the contents of a nucleotide sequence database to find sequences with regions homologous to regions of the original sequence.
BLASTP	BLAST searches in which an amino acid sequence is compared with the contents of an amino acid sequence database to find sequences with regions homologous to regions of the original sequence.
BLASTX	BLAST searches in which a nucleotide sequence is compared with the contents of an amino acid sequence database to find sequences with regions homologous to regions of the original sequence.
TBLASTN	BLAST searches in which an amino acid sequence is compared with the contents of a nucleotide sequence database to find sequences with regions homologous to regions of the original sequence.
TBLASTX	BLAST searches in which a nucleotide sequence is compared with the contents of a nucleotide sequence database to find sequences with regions homologous to regions of the original sequence.

CTLIB wrapper

The CTLIB wrapper supports Sysbase data sources. A server type specification is required for Sybase data sources supported by the CTLIB client software.

Server Type	Data Source
SYBASE	Sybase

DRDA wrapper

The DRDA wrapper is used for DB2 family data sources. A server type specification is required for the DB2 family data sources.

Table 140. DB2 family data sources

Server Type	Data Source
DB2/UDB	IBM DB2 Version 9.1 for Linux, UNIX, and Windows
DB2/ISERIES	IBM DB2 UDB for iSeries and AS/400
DB2/ZOS	IBM DB2 UDB for z/OS
DB2/VM	IBM DB2 for VM

Entrez wrapper

A server type specification is required for Entrez data sources.

Server Type	Data Source
NUCLEOTIDE	Entrez
OMIM	Entrez
PUBMED	Entrez

Excel wrapper

A server type specification is not required for Excel data sources.

Server Type	Data Source
Not required in the CREATE SERVER statement.	Microsoft Excel

HMMER wrapper

A server type specification is required for each server that you want to run a HMMER search on for HMMER data sources supported by the HMMER daemon.

Server Type	Data Source
PFAM	HMMER
SEARCH	HMMER

Informix wrapper

A server type specification is required for Informix data sources supported by Informix Client SDK software.

Server Type	Data Source
INFORMIX	Informix

MSSQLODBC3 wrapper

A server type specification is required for Microsoft SQL Server data sources supported by the DataDirect Connect ODBC 4.2 (or later) driver or the Microsoft SQL Server ODBC 3.0 (or later) driver.

Server Type	Data Source
MSSQLSERVER	Microsoft SQL Server

NET8 wrapper

A server type specification is required for Oracle data sources supported by Oracle NET8 client software.

Server Type	Data Source
ORACLE	Oracle Version 8.0. or later

ODBC wrapper

A server type specification is required for ODBC data sources supported by the ODBC 3.x driver.

Server Type	Data Source
ODBC	ODBC

OLE DB wrapper

A server type definition is not required for OLE DB providers compliant with Microsoft OLE DB 2.0 or later.

Server Type	Data Source
Not required in the CREATE SERVER statement.	Any OLE DB provider

Table-structured files wrapper

A server type definition is not required for table-structured file data sources.

Server Type	Data Source
Not required in the CREATE SERVER statement.	Table-structured files

Teradata wrapper

A server type definition is required for Teradata data sources supported by the Teradata client software.

Server Type	Data Source
TERADATA	Teradata

Web services wrapper

A server type definition is not required for Web services data sources.

Server Type	Data Source
Not required in the CREATE SERVER statement.	Any Web services data source.

WebSphere Business Integration wrapper

A server type definition is required for business application data sources supported by the WebSphere Business Integration wrapper.

Server Type	Data Source
WBI	WebSphere Business Integration 2.2 or 2.3

XML wrapper

A server type definition is not required for XML data sources.

Server Type	Data Source
Not required in the CREATE SERVER statement.	XML

User mapping options for federated systems

These options are used with the CREATE USER MAPPING and ALTER USER MAPPING statements.

Table 141. User mapping options and their settings

Option	Valid settings	Default setting
ACCOUNTING	DRDA: Used to specify a DRDA accounting string. Valid settings include any string of length 255 or less. This option is required only if accounting information needs to be passed. See the DB2 Connect™ Users Guide for more information.	None
GUEST	Specifies if the wrapper is to use the guest access mode to the BioRS server. **Y** The wrapper uses the guest access mode to the BioRS server. **N** The wrapper does not use the guest access mode to the BioRS server. When set to a value of Y, this option is mutually exclusive with the REMOTE_AUTHID option and the REMOTE_PASSWORD option. Valid for the BioRS data source.	N
REMOTE_AUTHID	Indicates the authorization ID used at the data source. Valid settings include any string of length 255 or less. Valid for the BioRS and Web services data sources.	The authorization ID you use to connect to DB2.
PROXY_AUTHID	Specifies the password to use when the proxy server requires authentication. Contact your network administrator for the password. Valid for BioRS, Blast, Entrez, HMMER, Script, and Web services data sources.	
PROXY_PASSWORD	Specifies the user name to use when the proxy server requires authentication. Contact your network administrator for the user name. Valid for BioRS, Blast, Entrez, HMMER, Script, and Web services data sources.	

Table 141. User mapping options and their settings (continued)

Option	Valid settings	Default setting
REMOTE_PASSWORD	Indicates the authorization password used at the data source. Valid settings include any string of length 32 or less. If your server requires a password and you do not set this option, you must ensure that the following conditions are met or the connection will fail: • The database manager configuration parameter AUTHENTICATON is set to SERVER. • The server option PASSWORD is omitted or set to Y (the default). • When you connected to the DB2 database, you specified an authorization ID and password. The password that you specified must be the same as the password of your remote server. Valid for the BioRS and Web services data sources.	The password you use to connect to the DB2 if both conditions listed in the valid settings column are met.
SSL_CLIENT_ CERTIFICATE_LABEL	Specifies the client certificate that is sent during SSL authentication. If the value is not specified, the current DB2 authorization ID will be sent. Valid for the Web services data source.	None.

Wrapper options for federated systems

Wrapper options are used to configure the wrapper or to define how the federated server uses the wrapper. Wrapper options can be set when you create or alter the wrapper.

All relational and nonrelational data sources use the DB2_FENCED wrapper option. The ODBC and Teradata wrappers support the DB2_SOURCE_CLIENT_MODE wrapper option. The Entrez data source uses the EMAIL wrapper option. The ODBC data source uses the MODULE wrapper option. BioRS, BLAST, Entrez, HMMER, web services, and XML data sources can use the wrapper options for proxies. The SSL options are supported by the BLAST, HMMER, SCRIPT, web services, and XML wrappers.

Table 142. Wrapper options and their settings

Option	Valid settings	Default setting
DB2_FENCED	Specifies whether the wrapper runs in fenced or trusted mode. **Y** The wrapper runs in fenced mode. **N** The wrapper runs in trusted mode.	Relational wrappers: N. Nonrelational wrappers from IBM: N. Nonrelational wrappers from third parties: Y.

Table 142. Wrapper options and their settings (continued)

Option	Valid settings	Default setting
DB2_SOURCE_ CLIENT_MODE	Specifies that the data source client is 32-bit and that the database instance on the federated server is 64-bit. When you specify this option, you must also set the DB2_FENCED wrapper option to Y. This option applies only to ODBC and Teradata data sources, and is currently supported only on AIX or Solaris operating systems. The only valid value is 32BIT. The value is not case sensitive. **32BIT** The data source client that is installed on the federated server is 32-bit.	None.
DB2_UM_PLUGIN	If you use a plugin for retrieving user mappings from an external repository, specify the class name including package (for example, 'com.ibm.ii.um.ldap.UserMappingRepository'). If you are using the Lightweight Directory Access Protocol (LDAP) sample plugin, you can use the class name (for example, UserMappingRepository).	None.
EMAIL	Specifies an e-mail address when you register the Entrez wrapper. This e-mail address is included with all queries and allows NCBI to contact you if there are problems, such as too many queries overloading the NCBI servers. This option is required.	
MODULE	Specifies the full path of the library that contains the ODBC Driver Manager implementation or the SQL/CLI implementation. Required for the ODBC wrapper on UNIX federated servers.	On Windows, the default value is odbc32.dll
PROXY_SERVER_ NAME	Specifies the proxy server name or the IP address. This field is required if the value of PROXY_TYPE is 'HTTP' or 'SOCKS'. The IP address can be an IPv4 address (for example, 192.168.1.1) or it can be a colon-separated IPv6 address (for example, 1080:0:0:0:8:800:200C:417A). Contact your network administrator for the server name or IP address of the proxy server.	None.
PROXY_SERVER_ PORT	Specifies the proxy server port number. This field is required if the value of PROXY_TYPE is 'HTTP' or 'SOCKS'. Contact your network administrator for the port number of the proxy server. Specifying the PROXY_SERVER_PORT option in a CREATE SERVER statement overrides the PROXY_SERVER_PORT option in a CREATE WRAPPER statement overrides the the	None.

Table 142. Wrapper options and their settings (continued)

Option	Valid settings	Default setting
PROXY_TYPE	Specifies the type of proxy type that is used to access the Internet when behind a firewall. The valid values are ’NONE’, ’HTTP’, or ’SOCKS’. The default value is ’NONE’. Contact your network administrator for the type of proxy that is used. The BLAST, HMMER, and SCRIPT wrappers do not support HTTP proxies.	’NONE’
SSL_KEYSTORE_ FILE	Specifies the name of the certificate storage file to use for SSL/TLS communications. The value that you specify must be a full path that is accessible by the DB2 agent or FMP process. Optional: You can specify the value ’GSK_MS_CERTIFICATE_STORE’ to use the native Microsoft certificate storage.	None.
SSL_KEYSTORE_ PASWORD	Specifies the password that is used to access the SSL certificate storage. This password is encrypted when it is stored in the DB2 catalog.	None.
SSL_VERIFY_ SERVER_ CERTIFICATE	Specifies if the server certificate should be verified during SSL authenticate. The values are not case sensitive. To authenticate, use one of the following values: ’Y’, ’YES’, ’T’, or ’TRUE’. To disable, use ’F’, ’FALSE’, ’N’, or ’NO’.	’NO’

Chapter 25. Federated system and data source mappings

Default forward data type mappings

The two kinds of mappings between data source data types and federated database data types are forward type mappings and reverse type mappings. In a forward type mapping, the mapping is from a remote type to a comparable local type.

You can override a default type mapping, or create a new type mapping with the CREATE TYPE MAPPING statement.

These mappings are valid with all the supported versions, unless otherwise noted.

For all default forward data types mapping from a data source to the federated database, the federated schema is SYSIBM.

The following tables show the default forward mappings between federated database data types and data source data types.

DB2 Database for Linux, UNIX, and Windows data sources

The following table lists the forward default data type mappings for DB2 Database for Linux, UNIX, and Windows data sources.

Table 143. DB2 Database for Linux, UNIX, and Windows forward default data type mappings (Not all columns shown)

REMOTE_ TYPE NAME	REMOTE_ LOWER _LEN	REMOTE_ UPPER_ LEN	REMOTE_ LOWER_ SCALE	REMOTE_ UPPER_ SCALE	REMOTE_ BIT_ DATA	REMOTE_ DATA_ OPERATORS	FEDERATED_ TYPE NAME	FEDERATED_ LENGTH	FEDERATED_ SCALE	FEDERATED_ BIT_ DATA
BIGINT	-	-	-	-	-	-	BIGINT	-	0	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	-	-	-	-	-	-	CHAR	-	0	N
CHAR	-	-	-	-	Y	-	CHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	-	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
LONGVAR	-	-	-	-	N	-	CLOB	-	-	-
LONGVAR	-	-	-	-	Y	-	BLOB	-	-	-
LONGVARG	-	-	-	-	-	-	DBCLOB	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	0	N

Table 143. DB2 Database for Linux, UNIX, and Windows forward default data type mappings (Not all columns shown) (continued)

REMOTE_ TYPE NAME	REMOTE_ LOWER _LEN	REMOTE_ UPPER_ LEN	REMOTE_ LOWER_ SCALE	REMOTE_ UPPER_ SCALE	REMOTE_ BIT_ DATA	REMOTE_ DATA_ OPERATORS	FEDERATED_ TYPE NAME	FEDERATED_ LENGTH	FEDERATED_ SCALE	FEDERATED_ BIT_ DATA
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	0	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	0	N

DB2 for iSeries data sources

The following table lists the forward default data type mappings for DB2 for iSeries data sources.

Table 144. DB2 for iSeries forward default data type mappings (Not all columns shown)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
NUMERIC	-	-	-	-	-	-	DECIMAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

DB2 for VM and VSE data sources

The following table lists the forward default data type mappings for DB2 for VM and VSE data sources.

Table 145. DB2 Server for VM and VSE forward default data type mappings (Not all columns shown)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBAHW	-	-	-	-	-	-	SMALLINT	-	0	-
DBAINT	-	-	-	-	-	-	INTEGER	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPH	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

DB2 for z/OS data sources

The following table lists the forward default data type mappings for DB2 for z/OS data sources.

Table 146. DB2 for z/OS forward default data type mappings (Not all columns shown)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
ROWID	-	-	-	-	Y	-	VARCHAR	40	-	Y

Table 146. DB2 for z/OS forward default data type mappings (Not all columns shown) (continued)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

Informix data sources

The following table lists the forward default data type mappings for Informix data sources.

Table 147. Informix forward default data type mappings (Not all columns shown)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
BLOB	-	-	-	-	-	-	BLOB	2147483647	-	-
BOOLEAN	-	-	-	-	-	-	CHARACTER	1	-	-
BYTE	-	-	-	-	-	-	BLOB	2147483647	-	-
CHAR	1	254	-	-	-	-	CHARACTER	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CLOB	-	-	-	-	-	-	CLOB	2147483647	-	-
DATE	-	-	-	-	-	-	DATE	4	-	-
DATETIME	0	4	0	4	-	-	DATE	4	-	-
DATETIME	6	10	6	10	-	-	TIME	3	-	-
DATETIME	0	4	6	15	-	-	TIMESTAMP	10	-	-
DATETIME	6	10	11	15	-	-	TIMESTAMP	10	-	-
DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
DECIMAL	32	130	-	-	-	-	DOUBLE	8	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
INTERVAL	-	-	-	-	-	-	VARCHAR	25	-	-
INT8	-	-	-	-	-	-	BIGINT	19	0	-
LVARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
MONEY	1	31	0	31	-	-	DECIMAL	-	-	-
MONEY	32	32	-	-	-	-	DOUBLE	8	-	-
NCHAR	1	254	-	-	-	-	CHARACTER	-	-	-
NCHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
NVARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
REAL	-	-	-	-	-	-	REAL	4	-	-
SERIAL	-	-	-	-	-	-	INTEGER	4	-	-
SERIAL8	-	-	-	-	-	-	BIGINT	-	-	-
SMALLFLOAT	-	-	-	-	-	-	REAL	4	-	-

Table 147. Informix forward default data type mappings (Not all columns shown) (continued)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
TEXT	-	-	-	-	-	-	CLOB	2147483647	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-

Notes:

- For the Informix DATETIME data type, the DB2 UNIX and Windows federated server uses the Informix high-level qualifier as the REMOTE_LENGTH and the Informix low-level qualifier as the REMOTE_SCALE.

  The Informix qualifiers are the ″TU_″ constants defined in the Informix Client SDK `datatime.h` file. The constants are:

0 = YEAR	8 = MINUTE	13 = FRACTION(3)
2 = MONTH	10 = SECOND	14 = FRACTION(4)
4 = DAY	11 = FRACTION(1)	15 = FRACTION(5)
6 = HOUR	12 = FRACTION(2)	

Microsoft SQL Server data sources

The following table lists the forward default data type mappings for Microsoft SQL Server data sources.

Table 148. Microsoft SQL Server forward default data type mappings

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
bigint [2]	-	-	-	-	-	-	BIGINT	-	-	-
binary	1	254	-	-	-	-	CHARACTER	-	-	Y
binary	255	8000	-	-	-	-	VARCHAR	-	-	Y
bit	-	-	-	-	-	-	SMALLINT	2	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	8000	-	-	-	-	VARCHAR	-	-	N
datetime	-	-	-	-	-	-	TIMESTAMP	10	-	-
datetimen	-	-	-	-	-	-	TIMESTAMP	10	-	-
decimal	1	31	0	31	-	-	DECIMAL	-	-	-
decimal	32	38	0	38	-	-	DOUBLE	-	-	-
decimaln	1	31	0	31	-	-	DECIMAL	-	-	-
decimaln	32	38	0	38	-	-	DOUBLE	-	-	-
DUMMY2000 [1]	1	38	-84	127	-	-	DOUBLE	-	-	-
float	-	8	-	-	-	-	DOUBLE	8	-	-
floatn	-	8	-	-	-	-	DOUBLE	8	-	-
float	-	4	-	-	-	-	REAL	4	-	-
floatn	-	4	-	-	-	-	REAL	4	-	-
image	-	-	-	-	-	-	BLOB	2147483647	-	Y
int	-	-	-	-	-	-	INTEGER	4	-	-
intn	-	-	-	-	-	-	INTEGER	4	-	-
money	-	-	-	-	-	-	DECIMAL	19	4	-
moneyn	-	-	-	-	-	-	DECIMAL	19	4	-
nchar	1	127	-	-	-	-	CHAR	-	-	N
nchar	128	4000	-	-	-	-	VARCHAR	-	-	N

Table 148. Microsoft SQL Server forward default data type mappings (continued)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
numeric	1	31	0	31	-	-	DECIMAL	-	-	-
numeric	32	38	0	38	-	-	DOUBLE	8	-	-
numericn	32	38	0	38	-	-	DOUBLE	-	-	-
numericn	1	31	0	31	-	-	DECIMAL	-	-	-
ntext	-	-	-	-	-	-	CLOB	2147483647	-	Y
nvarchar	1	4000	-	-	-	-	VARCHAR	-	-	N
real	-	-	-	-	-	-	REAL	4	-	-
smallint	-	-	-	-	-	-	SMALLINT	2	-	-
smalldatetime	-	-	-	-	-	-	TIMESTAMP	10	-	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
smallmoneyn	-	-	-	-	-	-	DECIMAL	10	4	-
SQL_BIGINT	-	-	-	-	-	-	DECIMAL	-	-	-
SQL_BIGINT [2]	-	-	-	-	-	-	BIGINT	-	-	-
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	8000	-	-	-	-	VARCHAR	-	-	N
SQL_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_DECIMAL	32	32	0	31	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_GUID	1	4000	-	-	Y	-	VARCHAR	16	-	Y
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_ LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N
SQL_ LONGVARBINARY	-	-	-	-	-	-	BLOB	-	-	Y
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_REAL	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	-	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_VARBINARY	1	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	8000	-	-	-	-	VARCHAR	-	-	N
text	-	-	-	-	-	-	CLOB	-	-	N
timestamp	-	-	-	-	-	-	VARCHAR	8		Y
tinyint	-	-	-	-	-	-	SMALLINT	2	-	-
uniqueidentifier	1	4000	-	-	Y	-	VARCHAR	16	-	Y
varbinary	1	8000	-	-	-	-	VARCHAR	-	-	Y
varchar	1	8000	-	-	-	-	VARCHAR	-	-	N

Table 148. Microsoft SQL Server forward default data type mappings (continued)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data

Note:

1. This type mapping is valid only with Windows 2000 operating systems.
2. This type mapping is valid only with Microsoft SQL Server Version 2000.

ODBC data sources

The following table lists the forward default data type mappings for ODBC data sources.

Table 149. ODBC forward default data type mappings (Not all columns shown)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
SQL_BIGINT	-	-	-	-	-	-	BIGINT	8	-	-
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	32672	-	-	-	-	VARCHAR	-	-	N
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_ LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N
SQL_ LONGVARBINARY	-	-	-	-	-	-	BLOB	-	-	Y
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_NUMERIC	32	32	0	31	-	-	DOUBLE	8	-	-
SQL_REAL	-	-	-	-	-	-	REAL	4	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TYPE_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_TYPE_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TYPE_ TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	-	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_VARBINARY	1	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	N
SQL_WCHAR	1	127	-	-	-	-	CHAR	-	-	N
SQL_WCHAR	128	16336	-	-	-	-	VARCHAR	-	-	N
SQL_WVARCHAR	1	16336	-	-	-	-	VARCHAR	-	-	N
SQL_ WLONGVARCHAR	-	1073741823		-	-	-	CLOB	2147483647	-	N

Oracle NET8 data sources

The following table lists the forward default data type mappings for Oracle NET8 data sources.

Table 150. Oracle NET8 forward default data type mappings

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
BLOB	0	0	0	0	-	\0	BLOB	2147483647	0	Y
CHAR	1	254	0	0	-	\0	CHAR	0	0	N
CHAR	255	2000	0	0	-	\0	VARCHAR	0	0	N
CLOB	0	0	0	0	-	\0	CLOB	2147483647	0	N
DATE	0	0	0	0	-	\0	TIMESTAMP	0	0	N
FLOAT	1	126	0	0	-	\0	DOUBLE	0	0	N
LONG	0	0	0	0	-	\0	CLOB	2147483647	0	N
LONG RAW	0	0	0	0	-	\0	BLOB	2147483647	0	Y
NUMBER	10	18	0	0	-	\0	BIGINT	0	0	N
NUMBER	1	38	-84	127	-	\0	DOUBLE	0	0	N
NUMBER	1	31	0	31	-	>=	DECIMAL	0	0	N
NUMBER	1	4	0	0	-	\0	SMALLINT	0	0	N
NUMBER	5	9	0	0	-	\0	INTEGER	0	0	N
NUMBER	-	10	0	0	-	\0	DECIMAL	0	0	N
RAW	1	2000	0	0	-	\0	VARCHAR	0	0	Y
ROWID	0	0	0	NULL	-	\0	CHAR	18	0	N
TIMESTAMP [1]	-	-	-	-	-	-	TIMESTAMP	10	-	-
VARCHAR2	1	4000	0	0	-	\0	VARCHAR	0	0	N

Note:

1. This type mapping is valid only for Oracle 9i (or later) client and server configurations.

Sybase data sources

The following table lists the forward default data type mappings for Sybase data sources.

Table 151. Sybase CTLIB forward default data type mappings

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
binary	1	254	-	-	-	-	CHAR	-	-	Y
binary	255	16384	-	-	-	-	VARCHAR	-	-	Y
bit	-	-	-	-	-	-	SMALLINT	-	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	16384	-	-	-	-	VARCHAR	-	-	N
char null (see varchar)										
datetime	-	-	-	-	-	-	TIMESTAMP	-	-	-
datetimn	-	-	-	-	-	-	TIMESTAMP	-	-	-
decimal	1	31	0	31	-	-	DECIMAL	-	-	-
decimal	32	38	0	38	-	-	DOUBLE	-	-	-
decimaln	1	31	0	31	-	-	DECIMAL	-	-	-

Table 151. Sybase CTLIB forward default data type mappings (continued)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
decimaln	32	38	0	38	-	-	DOUBLE	-	-	-
float	-	4	-	-	-	-	REAL	-	-	-
float	-	8	-	-	-	-	DOUBLE	-	-	-
floatn	-	4	-	-	-	-	REAL	-	-	-
floatn	-	8	-	-	-	-	DOUBLE	-	-	-
image	-	-	-	-	-	-	BLOB	-	-	-
int	-	-	-	-	-	-	INTEGER	-	-	-
intn	-	-	-	-	-	-	INTEGER	-	-	-
money	-	-	-	-	-	-	DECIMAL	19	4	-
moneyn	-	-	-	-	-	-	DECIMAL	19	4	-
nchar	1	254	-	-	-	-	CHAR	-	-	N
nchar	255	16384	-	-	-	-	VARCHAR	-	-	N
nchar null (see nvarchar)										
numeric	1	31	0	31	-	-	DECIMAL	-	-	-
numeric	32	38	0	38	-	-	DOUBLE	-	-	-
numericn	1	31	0	31	-	-	DECIMAL	-	-	-
numericn	32	38	0	38	-	-	DOUBLE	-	-	-
nvarchar	1	16384	-	-	-	-	VARCHAR	-	-	N
real	-	-	-	-	-	-	REAL	-	-	-
smalldatetime	-	-	-	-	-	-	TIMESTAMP	-	-	-
smallint	-	-	-	-	-	-	SMALLINT	-	-	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
sysname	1	254	-	-	-	-	CHAR	-	-	N
text	-	-	-	-	-	-	CLOB	-	-	-
timestamp	-	-	-	-	-	-	VARCHAR	8	-	Y
tinyint	-	-	-	-	-	-	SMALLINT	-	-	-
unichar[1]	1	254	-	-	-	-	CHAR	-	-	N
unichar[1]	255	16384	-	-	-	-	VARCHAR	-	-	N
unichar null (see univarchar)										
univarchar[1]	1	16384	-	-	-	-	VARCHAR	-	-	N
varbinary	1	16384	-	-	-	-	VARCHAR	-	-	Y
varchar	1	16384	-	-	-	-	VARCHAR	-	-	N

Note:

1. Valid for non-Unicode federated databases.

Teradata data sources

The following table lists the forward default data type mappings for Teradata data sources.

Table 152. Teradata forward default data type mappings (Not all columns shown)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
BYTE	1	254	-	-	-	-	CHAR	-	-	Y
BYTE	255	32672	-	-	-	-	VARCHAR	-	-	Y
BYTE	32673	64000	-	-	-	-	BLOB	-	-	-
BYTEINT	-	-	-	-	-	-	SMALLINT	-	-	-
CHAR	1	254	-	-	-	-	CHARACTER	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CHAR	32673	64000	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-
DECIMAL	1	18	0	18	-	-	DECIMAL	-	-	-
DOUBLE PRECISION	-	-	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	-	-
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	-	-
GRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
INTERVAL	-	-	-	-	-	-	CHAR	-	-	-
NUMERIC	1	18	0	18	-	-	DECIMAL	-	-	-
REAL	-	-	-	-	-	-	DOUBLE	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	-	-
VARBYTE	1	32762	-	-	-	-	VARCHAR	-	-	Y
VARBYTE	32763	64000	-	-	-	-	BLOB	-	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
VARCHAR	32673	64000	-	-	-	-	CLOB	-	-	-
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	-	-
VARGRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-

Default reverse data type mappings

For most data sources, the default type mappings are in the wrappers.

The two kinds of mappings between data source data types and federated database data types are forward type mappings and reverse type mappings. In a forward type mapping, the mapping is from a remote type to a comparable local type. The other type of mapping is a reverse type mapping, which is used with transparent DDL to create or modify remote tables.

The default type mappings for DB2 family data sources are in the DRDA wrapper. The default type mappings for Informix are in the INFORMIX wrapper, and so forth.

When you define a remote table or view to the federated database, the definition includes a reverse type mapping. The mapping is from a local federated database data type for each column, and the corresponding remote data type. For example, there is a default reverse type mapping in which the local type REAL points to the Informix type SMALLFLOAT.

Federated databases do not support mappings for LONG VARCHAR, LONG VARGRAPHIC, DATALINK, and user-defined types.

When you use the CREATE TABLE statement to create a remote table, you specify the local data types you want to include in the remote table. These default reverse type mappings will assign corresponding remote types to these columns. For example, suppose that you use the CREATE TABLE statement to define an Informix table with a column C2. You specify BIGINT as the data type for C2 in the statement. The default reverse type mapping of BIGINT depends on which version of Informix you are creating the table on. The mapping for C2 in the Informix table will be to DECIMAL in Informix Version 8 and to INT8 in Informix Version 9.

You can override a default reverse type mapping, or create a new reverse type mapping with the CREATE TYPE MAPPING statement.

The following tables show the default reverse mappings between federated database local data types and remote data source data types.

These mappings are valid with all the supported versions, unless otherwise noted.

DB2 Database for Linux, UNIX, and Windows data sources

The following table lists the reverse default data type mappings for DB2 Database for Linux, UNIX, and Windows data sources.

Table 153. DB2 Database for Linux, UNIX, and Windows reverse default data type mappings (Not all columns shown)

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Federated Bit Data
BIGINT	-	8	-	-	-	-	BIGINT	-	-	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	-	N

Table 153. DB2 Database for Linux, UNIX, and Windows reverse default data type mappings (Not all columns shown) (continued)

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Federated Bit Data
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

DB2 for iSeries data sources

The following table lists the reverse default data type mappings for DB2 for iSeries data sources.

Table 154. DB2 for iSeries reverse default data type mappings (Not all columns shown)

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operations	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	N
CHARACTER	-	-	-	-	Y	-	CHARACTER	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	NUMERIC	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	FLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	VARG	-	-	N

DB2 for VM and VSE data sources

The following table lists the reverse default data type mappings for DB2 for VM and VSE data sources.

Table 155. DB2 for VM and VSE reverse default data type mappings (Not all columns shown)

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	-
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-

Table 155. DB2 for VM and VSE reverse default data type mappings (Not all columns shown) (continued)

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPH	-	-	N

DB2 for z/OS data sources

The following table lists the reverse default data type mappings for DB2 for z/OS data sources.

Table 156. DB2 for z/OS reverse default data type mappings (Not all columns shown)

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	–
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	N

Informix data sources

The following table lists the reverse default data type mappings for Informix data sources.

Table 157. Informix reverse default data type mappings

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
BIGINT [1]	-	-	-	-	-	-	DECIMAL	19	-	-
BIGINT [2]	-	-	-	-	-	-	INT8	-	-	-
BLOB	1	2147483647	-	-	-	-	BYTE	-	-	-
CHARACTER	-	-	-	-	N	-	CHAR	-	-	-
CHARACTER	-	-	-	-	Y	-	BYTE	-	-	-
CLOB	1	2147483647	-	-	-	-	TEXT	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	SMALLFLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	DATETIME	6	10	-
TIMESTAMP	-	10	-	-	-	-	DATETIME	0	15	-
VARCHAR	1	254	-	-	N	-	VARCHAR	-	-	-
VARCHAR [1]	255	32672	-	-	N	-	TEXT	-	-	-
VARCHAR	-	-	-	-	Y	-	BYTE	-	-	-
VARCHAR [2]	255	2048	-	-	N	-	LVARCHAR	-	-	-
VARCHAR [2]	2049	32672	-	-	N	-	TEXT	-	-	-

Note:

1. This type mapping is valid only with Informix server Version 8 (or lower).
2. This type mapping is valid only with Informix server Version 9 (or higher).

For the Informix DATETIME data type, the federated server uses the Informix high-level qualifier as the REMOTE_LENGTH and the Informix low-level qualifier as the REMOTE_SCALE.

The Informix qualifiers are the "TU_" constants defined in the Informix Client SDK `datatime.h` file. The constants are:

0 = YEAR	8 = MINUTE	13 = FRACTION(3)
2 = MONTH	10 = SECOND	14 = FRACTION(4)
4 = DAY	11 = FRACTION(1)	15 = FRACTION(5)
6 = HOUR	12 = FRACTION(2)	

Microsoft SQL Server data sources

The following table lists the reverse default data type mappings for Microsoft SQL Server data sources.

Table 158. Microsoft SQL Server reverse default data type mappings (Not all columns shown)

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
BIGINT [1]	-	-	-	-	-	-	bigint	-	-	-
BLOB	-	-	-	-	-	-	image	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
DATE	-	4	-	-	-	-	datetime	-	-	-

Table 158. Microsoft SQL Server reverse default data type mappings (Not all columns shown) (continued)

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
DECIMAL	-	-	-	-	-	-	decimal	-	-	-
DOUBLE	-	8	-	-	-	-	float	-	-	-
INTEGER	-	-	-	-	-	-	int	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
REAL	-	4	-	-	-	-	real	-	-	-
TIME	-	3	-	-	-	-	datetime	-	-	-
TIMESTAMP	-	10	-	-	-	-	datetime	-	-	-
VARCHAR	1	8000	-	-	N	-	varchar	-	-	-
VARCHAR	8001	32672	-	-	N	-	text	-	-	-
VARCHAR	1	8000	-	-	Y	-	varbinary	-	-	-
VARCHAR	8001	32672	-	-	Y	-	image	-	-	-

Note:

1. This type mapping is valid only with Microsoft SQL Server Version 2000.

Oracle NET8 data sources

The following table lists the reverse default data type mappings for Oracle NET8 data sources.

Table 159. Oracle NET8 reverse default data type mappings

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
BIGINT	0	8	0	0	N	\0	NUMBER	19	0	N
BLOB	0	2147483647	0	0	Y	\0	BLOB	0	0	Y
CHARACTER	1	254	0	0	N	\0	CHAR	0	0	N
CHARACTER	1	254	0	0	Y	\0	RAW	0	0	Y
CLOB	0	2147483647	0	0	N	\0	CLOB	0	0	N
DATE	0	4	0	0	N	\0	DATE	0	0	N
DECIMAL	0	0	0	0	N	\0	NUMBER	0	0	N
DOUBLE	0	8	0	0	N	\0	FLOAT	126	0	N
FLOAT	0	8	0	0	N	\0	FLOAT	126	0	N
INTEGER	0	4	0	0	N	\0	NUMBER	10	0	N
REAL	0	4	0	0	N	\0	FLOAT	63	0	N
SMALLINT	0	2	0	0	N	\0	NUMBER	5	0	N
TIME	0	3	0	0	N	\0	DATE	0	0	N
TIMESTAMP [1]	0	10	0	0	N	\0	DATE	0	0	N
TIMESTAMP [2]	0	10	0	0	N	\0	TIMESTAMP	6	0	N
VARCHAR	1	4000	0	0	N	\0	VARCHAR2	0	0	N
VARCHAR	1	2000	0	0	Y	\0	RAW	0	0	Y

Note:

1. This type mapping is valid only with Oracle Version 8.
2. This type mapping is valid only with Oracle Version 9 and Version 10.

Sybase data sources

The following table lists the reverse default data type mappings for Sybase data sources.

Table 160. Sybase CTLIB default reverse data type mappings

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
BIGINT	-	-	-	-	-	-	decimal	19	0	-
BLOB	-	-	-	-	-	-	image	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
DATE	-	-	-	-	-	-	datetime	-	-	-
DECIMAL	-	-	-	-	-	-	decimal	-	-	-
DOUBLE	-	-	-	-	-	-	float	-	-	-
INTEGER	-	-	-	-	-	-	integer	-	-	-
REAL	-	-	-	-	-	-	real	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
TIME	-	-	-	-	-	-	datetime	-	-	-
TIMESTAMP	-	-	-	-	-	-	datetime	-	-	-
VARCHAR[1]	1	255	-	-	N	-	varchar	-	-	-
VARCHAR[1]	256	32672	-	-	N	-	text	-	-	-
VARCHAR [2]	1	16384	-	-	N	-	varchar	-	-	-
VARCHAR [2]	16385	32672	-	-	N	-	text	-	-	-
VARCHAR[1]	1	255	-	-	Y	-	varbinary	-	-	-
VARCHAR[1]	256	32672	-	-	Y	-	image	-	-	-
VARCHAR [2]	1	16384	-	-	Y	-	varbinary	-	-	-
VARCHAR [2]	16385	32672	-	-	Y	-	image	-	-	-

Note:

1. This type mapping is valid only for CTLIB with Sybase server version 12.0 (or earlier).
2. This type mapping is valid only for CTLIB with Sybase server version 12.5 (or later).

Teradata data sources

The following table lists the reverse default data type mappings for Teradata data sources.

Table 161. Teradata reverse default data type mappings (Not all columns shown)

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
BLOB [1]	1	64000	-	-	-	-	VARBYTE	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	-
CHARACTER	-	-	-	-	Y	-	BYTE	-	-	-
CLOB [2]	1	64000	-	-		-	VARCHAR	-	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-
DBCLOB [3]	1	32000	-	-	-	-	VARGRAPHIC	-	-	-
DECIMAL	1	18	0	18	-	-	DECIMAL	-	-	-
DECIMAL	19	31	0	31	-	-	FLOAT	-	-	-

Table 161. Teradata reverse default data type mappings (Not all columns shown) (continued)

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
DOUBLE	-	-	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	FLOAT	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARBYTE	-	-	-
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

Note:

1. The Teradata VARBYTE data type can contain only the specified length (1 to 64000) of a BLOB data type.
2. The Teradata VARCHAR data type can contain only the specified length (1 to 64000) of a CLOB data type.
3. The Teradata VARGRAPHIC data type can contain only the specified length (1 to 32000) of a DBCLOB data type.

Unicode default data type mappings

Unicode default forward data type mappings - Microsoft SQL Server wrapper

The following table lists the default forward data type mapping for the Microsoft SQL Server wrapper when the federated database is a Unicode database.

Table 162. Unicode default forward data type mappings for the Microsoft SQL Server wrapper

UTF-8	Microsoft SQL Server	
Data type	**Data type**	**Length**
CHAR	CHAR	1 to 254 bytes
VARCHAR	CHAR	255 to 8000 bytes
	VARCHAR	1 to 8000 bytes
CLOB	TEXT	-
GRAPHIC	NCHAR	1 to 127 characters
VARGRAPHIC	NCHAR	128 to 16336 characters
	NVARCHAR	1 to 16336 characters
DBCLOB	NTEXT	-

Unicode default reverse data type mappings - Microsoft SQL Server wrapper

The following table lists the default reverse data type mapping for the Microsoft SQL Server wrapper when the federated database is a Unicode database.

Table 163. Unicode default reverse data type mappings for the Microsoft SQL Server wrapper

UTF-8		Microsoft SQL Server
Data type	**Length**	**Data type**
CHAR	1 to 254 bytes	CHAR
VARCHAR	1 to 32672 bytes	VARCHAR
CLOB	1 to 2 147 483 647 bytes	TEXT
GRAPHIC	1 to 127 characters	NCHAR
VARGRAPHIC	1 to 16336 characters	NVARCHAR
DBCLOB	1 to 1 073 741 823 characters	NTEXT

Unicode default forward data type mappings - NET8 wrapper

The following table lists the default forward data type mapping for the NET8 wrapper when the federated database is a Unicode database.

Table 164. Unicode default forward data type mappings for the NET8 wrapper

UTF-8	Oracle	
Data type	**Data type**	**Length**
CHAR	CHAR	1 to 254 bytes
VARCHAR	CHAR	255 to 2000 bytes
	VARCHAR2	1 to 4000 bytes
DBCLOB	NCLOB	
GRAPHIC	NCHAR	1 to 127 characters
VARGRAPHIC	NCHAR	128 to 1000 characters
	NVARCHAR2	1 to 2000 characters

Unicode default reverse data type mappings - NET8 wrapper

The following table lists the default reverse data type mapping for the NET8 wrapper when the federated database is a Unicode database.

Table 165. Unicode default reverse data type mappings for the NET8 wrapper

UTF-8		Oracle
Data type	**Length**	**Data type**
CHAR	1 to 254 bytes	CHAR
VARCHAR	1 to 4000 bytes	VARCHAR2
CLOB	1 to 2 147 483 647 bytes	CLOB
GRAPHIC	1 to 127 characters	NCHAR
VARGRAPHIC	1 to 2000 characters	NVARCHAR2
DBCLOB	1 to 1 073 741 823 characters	NCLOB

Unicode default forward data type mappings - ODBC wrapper

The following table lists the default forward data type mapping for the ODBC wrapper when the federated database is a Unicode database.

Table 166. Unicode default forward data type mappings for the ODBC wrapper

UTF-8	**ODBC**	
Data type	**Data type**	**Length**
CHAR	SQL_CHAR	1 to 254 bytes
VARCHAR	SQL_CHAR	255 to 32672 bytes
	SQL_VARCHAR	1 to 32672 bytes
CLOB	SQL_LONGVARCHAR	-
GRAPHIC	SQL_WCHAR	1 to 127 characters
VARGRAPHIC	SQL_WVARCHAR	128 to 16336 characters
	SQL_WVARCHAR	1 to 16336 characters
DBCLOB	SQL_WLONGVARCHAR	-

Unicode default reverse data type mappings - ODBC wrapper

The following table lists the default reverse data type mapping for the ODBC wrapper when the federated database is a Unicode database.

Table 167. Unicode default reverse data type mappings for the ODBC wrapper

UTF-8		**ODBC**
Data type	**Length**	**Data type**
CHAR	1 to 254 bytes	SQL_CHAR
VARCHAR	1 to 32672 bytes	SQL_VARCHAR
CLOB	1 to 2 147 483 647 bytes	SQL_LONGVARCHAR
GRAPHIC	1 to 127 characters	SQL_WCHAR
VARGRAPHIC	1 to 16336 characters	SQL_WVARCHAR
DBCLOB	1 to 1 073 741 823 characters	SQL_WLONGVARCHAR

Unicode default forward data type mappings - Sybase wrapper

The following table lists the default forward data type mapping for the CTLIB wrapper when the federated database is a Unicode database.

Table 168. Unicode default forward data type mappings for the Sybase CTLIB wrapper

UTF-8	**Sybase**	
Data type	**Data type**	**Length**
CHAR	char	1 to 254 bytes
	nchar	1 to 127 characters
VARCHAR	char	255 to 32672 bytes
	varchar	1 to 32672 bytes
	nchar	128 to 16336 characters
	nvarchar	1 to 16336 characters
CLOB	text	
GRAPHIC	unichar	1 to 127 characters
VARGRAPHIC	unichar	128 to 16336 characters
	univarchar	1 to 16336 characters

Unicode default reverse data type mappings - Sybase wrapper

The following table lists the default reverse data type mapping for the CTLIB wrapper when the federated database is a Unicode database.

Table 169. Unicode default reverse data type mappings for the Sybase CTLIB wrapper

UTF-8		**Sybase**
Data type	**Length**	**Data type**
CHAR	1 to 254 bytes	char
VARCHAR	1 to 32672 bytes	varchar
CLOB	1 to 2 147 483 647 bytes	text
GRAPHIC	1 to 127 characters	unichar
VARGRAPHIC	1 to 16336 characters	univarchar

Data types supported for nonrelational data sources

For most of the nonrelational data sources, you must specify the column information, including data type, when you create the nicknames to access the data source.

Some of the nonrelational wrappers create all of the columns required to access a data source. These are called *fixed columns*. Other wrappers let you specify some or all of the data types for the columns in the CREATE NICKNAME statement.

The following sections list the wrappers that you can specify the data types for, and the data types that the wrapper supports.

Data types supported by the BioRS wrapper

The following table lists the DB2 data types that the BioRS wrapper supports.

Table 170. BioRS data types that map to DB2 data types

BioRS data types	**DB2 data type**
	CHARACTER
	CLOB
	VARCHAR

Data types supported by the BLAST wrapper

Some of the data types are automatically set for the fixed columns that the BLAST wrapper creates.

For the definition line fields, you can assign when you create a nickname. If the data in the definition line column is not compatible with the local column data type, you will get an error. For example, if you define a definition line column of type INTEGER and there are values in the column that are not numeric, an error is returned.

The following table lists the DB2 data types that the BLAST wrapper supports.

Table 171. BLAST data types that map to DB2 data types

BLAST data types	DB2 data type
definition line	CLOB (maximum length is 5 megabytes)
definition line	DOUBLE
definition line	FLOAT
definition line	INTEGER
definition line	VARCHAR

Data types supported by the Entrez wrapper

The following table lists the DB2 data types that the Entrez wrapper supports.

Table 172. Entrez data types that map to DB2 data types

Entrez data types	DB2 data type
	CHARACTER
	CLOB (maximum length is 5 megabytes)
	DATE
	DECIMAL
	DOUBLE
	INTEGER
	REAL
	SMALLINT
	TIMESTAMP
	VARCHAR

Data types supported by the Excel wrapper

The following table lists the DB2 data types that the Excel wrapper supports.

Table 173. Excel data types that map to DB2 data types

Excel data types	DB2 data type
	DATE
	FLOAT
	INTEGER
	VARCHAR

Data types supported by the HMMER wrapper

The following table lists the DB2 data types that the HMMER wrapper supports.

Table 174. HMMER data types that map to DB2 data types

HMMER data types	DB2 data type
	CLOB (maximum length is 5 megabytes)
	DOUBLE

Table 174. HMMER data types that map to DB2 data types (continued)

HMMER data types	DB2 data type
	FLOAT
	INTEGER
	VARCHAR

Data types supported by the Script wrapper

The following table lists the DB2 data types that the Script wrapper supports.

Table 175. Script data types that map to DB2 data types

Script data types	DB2 data type
	CLOB (maximum length is 5 megabytes)
	DATE
	DOUBLE
	INTEGER
	VARCHAR

Data types supported by the table-structured file wrapper

The following table lists the DB2 data types that the table-structured file wrapper supports.

Table 176. Table-structured file data types that map to DB2 data types

Table-structured file data types	DB2 data type
	CHARACTER
	CLOB (maximum length is 5 megabytes)
	DECIMAL
	DOUBLE
	FLOAT
	INTEGER
	REAL
	SMALLINT
	VARCHAR

Data types supported by the Web services wrapper

The following table lists the DB2 data types that the Web services wrapper supports. The Web services wrapper uses XML data types.

Table 177. XML data types that map to DB2 data types for the Web services wrapper

XML data types	DB2 data type
	BIGINT
	CHARACTER
	CHARACTER FOR BIT DATA
	CLOB (maximum length is 5 megabytes)

Table 177. XML data types that map to DB2 data types for the Web services wrapper (continued)

XML data types	DB2 data type
	DATE
	DECIMAL
	DOUBLE
	FLOAT
	INTEGER
	REAL
	TIME
	TIMESTAMP
	SMALLINT
	VARCHAR
	VARCHAR FOR BIT DATA

Data types supported by the WebSphere Business Integration wrapper

The following table lists the DB2 data types that the WebSphere Business Integration wrapper supports. The WebSphere Business Integration wrapper uses XML data types.

Table 178. XML data types that map to DB2 data types for the WebSphere Business Integration wrapper

XML data types	DB2 data type
	BIGINT
	CHARACTER
	CHARACTER FOR BIT DATA
	CLOB (limited to 5 megabytes)
	DATE
	DECIMAL
	DOUBLE
	FLOAT
	INTEGER
	REAL
	TIME
	TIMESTAMP
	SMALLINT
	VARCHAR
	VARCHAR FOR BIT DATA

Data types supported by the XML wrapper

The following table lists the DB2 data types that the XML wrapper supports.

Table 179. XML data types that map to DB2 data types for the XML wrapper

XML data types	DB2 data type
	CHARACTER
	CHARACTER FOR BIT DATA
	CLOB (maximum length is 5 megabytes)
	DATE
	DECIMAL
	DOUBLE
	FLOAT
	INTEGER
	REAL
	SMALLINT
	VARCHAR
	VARCHAR FOR BIT DATA

Accessing information about IBM

IBM has several methods for you to learn about products and services.

You can find the latest information on the Web at www.ibm.com/software/data/integration/db2ii/support.html:

- Product documentation in PDF and online information centers
- Product downloads and fix packs
- Release notes and other support documentation
- Web resources, such as white papers and IBM Redbooks™
- Newsgroups and user groups
- Book orders

To access product documentation, go to this site:

publib.boulder.ibm.com/infocenter/db2help/topic/

You can order IBM publications online or through your local IBM representative.

- To order publications online, go to the IBM Publications Center at www.ibm.com/shop/publications/order.
- To order publications by telephone in the United States, call 1-800-879-2755.

To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at www.ibm.com/planetwide.

Contacting IBM

You can contact IBM by telephone for customer support, software services, and general information.

Customer support

To contact IBM customer service in the United States or Canada, call 1-800-IBM-SERV (1-800-426-7378).

Software services

To learn about available service options, call one of the following numbers:

- In the United States: 1-888-426-4343
- In Canada: 1-800-465-9600

General information

To find general information in the United States, call 1-800-IBM-CALL (1-800-426-2255).

Go to www.ibm.com for a list of numbers outside of the United States.

Accessible documentation

Documentation is provided in XHTML format, which is viewable in most Web browsers.

XHTML allows you to view documentation according to the display preferences that you set in your browser. It also allows you to use screen readers and other assistive technologies.

Syntax diagrams are provided in dotted decimal format. This format is available only if you are accessing the online documentation using a screen reader.

Providing comments on the documentation

Please send any comments that you have about this information or other documentation.

Your feedback helps IBM to provide quality information. You can use any of the following methods to provide comments:

- Send your comments using the online readers' comment form at www.ibm.com/software/awdtools/rcf/.
- Send your comments by e-mail to comments@us.ibm.com. Include the name of the product, the version number of the product, and the name and part number of the information (if applicable). If you are commenting on specific text, please include the location of the text (for example, a title, a table number, or a page number).

Notices and trademarks

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing 2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION ″AS IS″ WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM trademarks and certain non-IBM trademarks are marked at their first occurrence in this document.

See http://www.ibm.com/legal/copytrade.shtml for information about IBM trademarks.

The following terms are trademarks or registered trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel®, Intel Inside® (logos), MMX and Pentium® are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product or service names might be trademarks or service marks of others.

Index

Printed in USA