IBM DB2 Information Integrator

Data Source Configuration Guide

Version 8

IBM DB2 Information Integrator

Data Source Configuration Guide

Version 8

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 435.

Contents

About this book

This book contains:

- Instructions for adding data sources to a federated system by registering wrappers. Wrappers are modules that enable you or an application to communicate with a data source using SQL statements.

Technical changes to the text are indicated by a vertical line to the left of the change.

Who should read this book

This book is for administrators who are setting up a federated database environment, and for application programmers who are developing applications for such an environment.

Conventions

This book uses these highlighting conventions:

Boldface type
: Indicates commands and graphical user interface (GUI) controls (for example, names of fields, names of folders, menu choices).

`Monospace type`
: Indicates examples of coding or of text that you type.

Italic type
: Indicates variables that you should replace with a value. Italic type also indicates book titles and emphasizes words.

UPPERCASE TYPE
: Indicates SQL keywords and names of objects (for example, tables, views, and servers).

How to read the syntax diagrams

Throughout this book, syntax is described using the structure defined as follows:

Read the syntax diagrams from left to right and top to bottom, following the path of the line.

The ►►─── symbol indicates the beginning of a statement.

The ───► symbol indicates that the statement syntax is continued on the next line.

The ►─── symbol indicates that a statement is continued from the previous line.

The ───►◄ symbol indicates the end of a statement.

Required items appear on the horizontal line (the main path).

```
►►──STATEMENT──required item────────────────────────────────────────────►◄
```

Optional items appear below the main path.

```
►►──STATEMENT──┬───────────────┬────────────────────────────────────────►◄
               └─optional item─┘
```

If an optional item appears above the main path, that item has no effect on the execution of the statement and is used only for readability.

```
               ┌─optional item─┐
►►──STATEMENT──┴───────────────┴────────────────────────────────────────►◄
```

If you can choose from two or more items, they appear in a stack.

If you *must* choose one of the items, one item of the stack appears on the main path.

```
►►──STATEMENT──┬─required choice1─┬─────────────────────────────────────►◄
               └─required choice2─┘
```

If choosing none of the items is an option, the entire stack appears below the main path.

```
►►──STATEMENT──┬──────────────────┬─────────────────────────────────────►◄
               ├─optional choice1─┤
               └─optional choice2─┘
```

If one of the items is the default, it will appear above the main path and the remaining choices will be shown below.

```
             ┌─default choice──┐
►►──STATEMENT─┼─────────────────┼─────────────────────────────────────►◄
             ├─optional choice─┤
             └─optional choice─┘
```

An arrow returning to the left, above the main line, indicates an item that can be repeated. In this case, repeated items must be separated by one or more blanks.

```
              ┌─────────────────┐
              ▼                 │
►►──STATEMENT───repeatable item─┴─────────────────────────────────────►◄
```

If the repeat arrow contains a comma, you must separate repeated items with a comma.

```
              ┌─,───────────────┐
              ▼                 │
►►──STATEMENT───repeatable item─┴─────────────────────────────────────►◄
```

A repeat arrow above a stack indicates that you can make more than one choice from the stacked items or repeat a single choice.

Keywords appear in uppercase (for example, FROM). They must be spelled exactly as shown. Variables appear in lowercase (for example, column-name). They represent user-supplied names or values in the syntax.

If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.

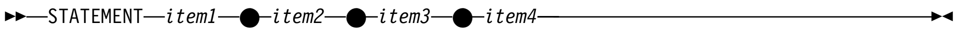
Sometimes a single variable represents a set of several parameters. For example, in the following diagram, the variable parameter-block can be replaced by any of the interpretations of the diagram that is headed **parameter-block**:

```
►►──STATEMENT──┤ parameter-block ├────────────────────────────────────►◄
```

parameter-block:

```
├──┬─parameter1─────────────────┬─────────────────────────────────────┤
   └─parameter2──┬─parameter3─┬─┘
                 └─parameter4─┘
```

Adjacent segments occurring between “large bullets” (●) may be specified in any sequence.



The above diagram shows that item2 and item3 may be specified in either order. Both of the following are valid:

```
STATEMENT item1 item2 item3 item4
STATEMENT item1 item3 item2 item4
```

Chapter 1. Overview of configuring access to data sources

The following sections provide a concise guide to configuring a federated server and database to access your data sources:

- They contain information about the basic steps needed to quickly perform the configuration steps.
- They outline several optional steps, if you need them, to fine-tune the data source configuration.

There are individual configuration chapters for each data source.

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 Center for the steps that require a command line. Toggle between these graphical user interfaces to quickly configure access to a data source. The steps to configure access are simliar, regardless of the data source. The basic steps and recommended interface are:

Table 1. The recommended interface and configuration steps

Configuration step	Recommended interface	Notes
1. Prepare the federated server for the data source	Client Configuration Assistant	For DB2 family data sources: Catalog the node and the remote database For Informix, Oracle, Sybase, Microsoft® SQL Server data sources: Setup and test the client configuration file
2. Create the wrappers	DB2 Control Center	
3. Create the server definitions	DB2 Control Center	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. In DB2 a *node* is equivalent to an instance, which is running copy of DB2.

Table 1. The recommended interface and configuration steps (continued)

Configuration step	Recommended interface	Notes
4. Create the user mappings	DB2 Control Center	If 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	Use the Show All Tables panel in the DB2 Control Center to verify the connections.
6. Create the nicknames	DB2 Control Center	

However, before you can configure access to a data source, you must make sure that the federated server has been set up properly. It is especially important that you:

- Link DB2 to the client software. This creates the data source wrapper libraries on the federated server.
- Set up the data source environment variables.

Related concepts:

- "Optional configuration steps" on page 7

Related tasks:

- "Checking the federated server setup" in the *DB2 Information Integrator Installation Guide*

Related reference:

- "Federated object naming rules" on page 19

Supported data sources

There are many data sources that you can access using a federated system. The following table lists the supported data sources:

Table 2. Supported data source versions and access methods.

Data source	Supported versions	Access method
DB2 Universal Database™ for Linux, UNIX, and Windows®	7.1, 7.2, 8.1	DRDA®
DB2 Universal Database for z/OS™ and OS/390®	6.1, 7.1 with the following APARs applied: • PQ62695 • PQ55393 • PQ56616 • PQ54605 • PQ46183 • PQ62139	DRDA
DB2 Universal Database for iSeries™	4.5 (or later) with the following APARs applied: • SA95719 • SE06003 • SE06872 • SI05990 • SI05991	DRDA
DB2 Server for VM and VSE	7.1 (or later) with fixes for APARs for schema functions applied.	DRDA
Informix™	7, 8, 9	Informix Client SDK
ODBC	3.x	ODBC driver for the data source, such as Redbrick ODBC Driver to access Redbrick.
OLE DB		OLE DB 2.0 (or later)
Oracle	7.3.4, 8.x, 9.x	SQLNET or NET8 client software
Microsoft SQL Server	6.5, 7.0, 2000	On Windows, the Microsoft SQL Server Client ODBC 3.0 (or later) driver. On UNIX, the DataDirect Technologies (formerly MERANT) Connect ODBC 3.7 (or later) driver.
Sybase	11.x,12.x	Sybase Open Client

Table 2. Supported data source versions and access methods. (continued)

Data source	Supported versions	Access method
Teradata	V2R3, V2R4	Teradata Call-Level Interface Version 2 (CLIv2) Release 04.06 (or later)
BLAST	2.x	BLAST daemon (supplied with the wrapper)
Documentum	Documentum server: EDMS 98 (also referred to as version 3) and 4i.	Documentum Client API/Library
Entrez	1.0	None
HMMER	2.2g	HMMER daemon (supplied with the wrapper)
IBM Lotus Extended Search	4.0	Extended Search Client Library (supplied with the wrapper)
Microsoft Excel	97, 2000	Excel 97 or 2000 installed on the federated server
Table-structured files		None
XML	1.0 specification	None

Related concepts:

- "Data sources" in the *Federated Systems Guide*

Create nicknames for each data source object

The task of creating a nickname is typically the most involved of the configuration tasks. This section provides an example of what you have to do to identify candidates for nicknames and to register a nickname for a federated data source object.

Data source objects can be relational or nonrelational. Examples of relational data source objects are: database tables, views, and synonyms (Informix only). Examples of nonrelational data source objects are: BLAST search algorithms, objects and registered tables in a Documentum Docbase, Microsoft® Excel files (.xls), table-structured files (.txt), and XML tagged files.

Tables and views that reside in the federated database are *local objects*. You do not create nicknames for these objects. You use the actual object name in your queries.

Remote objects are:

- Tables and views in another DB2® database instance on the federated server. You need to create nicknames for these objects.
- Data source objects that reside in another data source, such as: Oracle, Sybase, Documentum, and ODBC. You need to create nicknames for these objects.

When you submit a distributed request to the federated server, the request references a data source object by its nickname. Nicknames are mapped to specific object names at the data source. The mappings eliminate the need to qualify the nicknames by data source names. The location of the data source objects are transparent to the client application or end user. Nicknames are not alternative names for data source objects. They are pointers by which the federated server references these objects.

For example, if you define the nickname *DEPT* to represent an Informix™ database table called *NFX1.PERSON.DEPT*, the statement SELECT * FROM *DEPT* is allowed from the federated server. However, the statement, SELECT * FROM *NFX1.PERSON.DEPT* is not allowed.

When you create a nickname for a relational data source object, catalog data from the remote server is retrieved and stored in the federated global catalog.

For non-relational data sources, the way that data source information is stored in the global catalog varies from data source to data source. The information might be retrieved from the remote server, or you might have to include this information in the CREATE NICKNAME statement.

SQL Compiler uses this metadata to facilitate access to the data source object. For example, suppose that a nickname is defined for a table with an index. The metadata supplied to the global catalog is information related to the index, such as the name of each column in the index key.

To create a nickname, use the DB2 Control Center. You can also issue the CREATE NICKNAME statement in the DB2 Command Center or in the command line processor (CLP). You can define more than one nickname for the same data source object.

The following example shows a CREATE NICKNAME statement:

```
CREATE NICKNAME SYBSALES FOR SYBSERVER."salesdata"."europe"
```

where:

SYBSALES
: Is a unique nickname for the Sybase table or view.

Note: The nickname is a two-part name—the schema and the nickname. If you omit the schema when creating the nickname, the schema of the nickname will be the authid of the user creating the nickname. Nicknames can be 128 characters in length.

SYBSERVER."salesdata"."europe"
Is a three-part identifier for the remote data source object.

- *SYBSERVER* is the name you assigned to the data source server in the CREATE SERVER statement.
- *salesdata* is the name of the remote schema to which the object belongs. This value is case sensitive.
- *europe* is the name of the remote object that you want to access. This value is case sensitive.

When you create the nickname, the federated server uses the nickname to test the connection to the data source. It attempts to query the data source catalog. If the connection does not work, you will receive an error message.

Including column options when you create a nickname

Suppose that you want to create the nickname INDSALES for a table called INDONESIA_SALES. The table contains the column POSTAL_CODE with the data type of CHAR. The column contains only numeric characters. The data source has a collating sequence that differs from the federated database collating sequence. Typcially, the federated server would not sort this column at the data source. However, the POSTAL_CODE column contains only numeric characters ('0','1',...,'9'). You can indicate this by assigning a value of 'Y' to the NUMERIC_STRING column option. This gives the DB2 query optimizer the option of performing the sort at the data source. If the sort is performed remotely, you can avoid the overhead of porting the data to the federated server. To provide this information to the federated server, you add the NUMERIC_STRING column option to the CREATE NICKNAME statement. For example:

```
CREATE NICKNAME INDSALES FOR SERVER44."sales"."INDONESIA_SALES"
OPTIONS (POSTAL_CODE NUMERIC_STRING 'Y')
```

For some nonrelational data sources, the wrappers do not contain the default type mappings. If the wrapper does not contain the default type mappings, the corresponding DB2 for UNIX® and Windows® data types must be specified for each column of the data source object when the nickname is created. Each column must be mapped to a particular field or column in the data source object. For example:

```
CREATE NICKNAME DRUGDATA1
(DCODE INTEGER,DRUG CHAR(20),MANUFACTURER CHAR(20))
FOR SERVER biochem_lab
OPTIONS (FILE_PATH '/usr/pat/DRUGDATA1.TXT',
COLUMN_DELIMITER ',', KEY_COLUMN 'Dcode', VALIDATE_DATA_FILE 'Y')
```

Creating a nickname on a nickname

Occasionally, you may need to create a nickname on a nickname. Suppose you have a federated server using AIX® and a federated server using Windows. You want to access an Excel spreadsheet from both federated servers. However, the Excel wrapper is only supported on federated servers that use Windows. To access the Excel spreadsheet from the AIX federated server, use these steps:

1. On the Windows federated server, setup and configure the server to access Excel data sources.
2. Create a nickname for the Excel spreadsheet.
3. On the AIX federated server, setup and configure the server to access DB2 family data sources.
4. Create a nickname for the Excel nickname on the Windows federated server.

Related tasks:

- "Nicknames : Federated Systems help"
- "Filtering tables and views for creating nicknames : Federated Systems help"
- "Filtering tables for creating nicknames: Federated Systems help"
- "Creating nicknames: Federated Systems help"

Related reference:

- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*
- "Federated object naming rules" on page 19

Optional configuration steps

You can customize or tune columns of a nickname that are used in federated queries in the following manner:

- Specify indexes for objects that did not have an index when you originally configured access to the data source. For example, you would create an index specification when a table acquires a new index. Likewise, you would create an index specification if the data source object (such as a view) typically does not have indexes.
- Define 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.
- Define alternative function mappings, instead of using the default function mappings. This is especially useful when you want to force DB2® to use a user-defined function at the data source.

Chapter 2. Overview of a federated system

The following sections provide an overview of a federated system.

Wrappers and wrapper modules

Wrappers are mechanisms by which the federated server interacts with data sources. The federated server uses routines stored in a library called a *wrapper module* to implement a wrapper. These routines allow the federated server to perform operations such as connecting to a data source and retrieving data from it iteratively. Typically, the DB2® federated instance owner uses the CREATE WRAPPER statement to register a wrapper in the federated database.

You create one wrapper for each type of data source that you want to access. For example, suppose that you want to access three DB2 for z/OS™ database tables, one DB2 for iSeries™ table, two Informix™ tables, and one Informix view. You need to create only two wrappers: one for the DB2 data source objects and one for the Informix data source objects. Once these wrappers are registered in the federated database, you can use these wrappers to access other objects from those data sources. For example, you can use the DRDA® wrapper with all DB2 family data source objects—DB2 for Linux, UNIX,® and Windows, DB2 for z/OS and OS/390, DB2 for iSeries, and DB2 Server for VM and VSE.

You use the server definitions and nicknames to identify the specifics (name, location, and so forth) of each data source object.

A wrapper performs many tasks. Some of these tasks are:
- It connects to the data source. The wrapper uses the standard connection API of the data source.
- It submits queries to the data source.
  - For data sources that support SQL, the query is submitted in SQL.
  - For data sources that do not support SQL, the query is translated into the native query language of the source or into a series of source API calls.
- It receives results sets from the data source. The wrapper uses the data source standard APIs for receiving results set.
- It responds to federated server queries about the default data type mappings for a data source. The wrapper contains the default type

mappings that are used when nicknames are created for a data source object. For relational wrappers, data type mappings that you create override the default data type mappings. User-defined data type mappings are stored in the global catalog.

- It responds to federated server queries about the default function mappings for a data source. The wrapper contains information that the federated server needs to determine if DB2 functions are mapped to functions of the data source, and how the functions are mapped. This information is used by the SQL Compiler to determine if the data source is able to perform the query operations. For relational wrappers, function mappings that you create override the default function type mappings. User-defined function mappings are stored in the global catalog.

Wrapper options are used to configure the wrapper or to define how DB2 uses the wrapper.

Related concepts:

- "Server definitions and server options" on page 10

Related reference:

- "Default wrapper names" in the *Federated Systems Guide*

Server definitions and server options

After wrappers are created for the data sources, the federated instance owner defines the data sources to the federated database. The instance owner supplies a name to identify the data source, and other information that pertains to the data source. If the data source is an RDBMS, this information includes:

- The type and version of the RDBMS
- The database name for the data source on the RDBMS
- Metadata that is specific to the RDBMS

For example, a DB2® family data source can have multiple databases. The definition must specify which database the federated server can connect to. In contrast, an Oracle® data source has one database, and the federated server can connect to the database without knowing its name. The database name is not included in the federated server definition of an Oracle data source.

The name and other information that the instance owner supplies to the federated server are collectively called a *server definition*. Data sources answer requests for data and are servers in their own right.

The CREATE SERVER and ALTER SERVER statements are used to create and modify a server definition.

Some of the information within a server definition is stored as *server options*. When you create server definitions, it is important to understand the options that you can specify about the server. Some server options configure the wrapper and some affect the way DB2 uses the wrapper.

Server options can be set to persist over successive connections to the data source, or set for the duration of a single connection.

Related concepts:

- "User mappings" on page 11

Related reference:

- Appendix B, "Server options for federated systems", on page 367

Collating sequences and data source configuration

As part of the DB2 Information Integrator installation, the federated database was created. At that time, a collating sequence that matches the data source collating sequence was designated. When you register the server definition with the federated database, you need to set the COLLATING_SEQUENCE server option to 'Y'. This setting tells the federated database that the collating sequences of the federated database and the data source server match.

User mappings

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 most data sources, the federated server does this by using a valid user ID and password to that data source. When a user ID and password is required to connect to a data source, 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*.

Related concepts:

- "Nicknames and data source objects" on page 12

Nicknames and data source objects

After you create the server definitions and user mappings, the federated instance owner creates the nicknames. A *nickname* is an identifier that is used to reference the object located at the data sources that you want to access. The objects that nicknames identify are referred to as *data source objects*.

Nicknames are not alternative names for data source objects in the same way that aliases are alternative names. They are pointers by which the federated server references these objects. Nicknames are typically defined with the CREATE NICKNAME statement.

When an end user or a client application submits a distributed request to the federated server, the request does not need to specify the data sources. Instead, the request references the data source objects by their nicknames. The nicknames are mapped to specific objects at the data source. These mappings eliminate the need to qualify the nicknames by data source names. The location of the data source objects is transparent to the end user or the client application.

Suppose that you define the nickname *DEPT* to represent an Informix™ database table called *NFX1.PERSON*. The statement SELECT * FROM *DEPT* is allowed from the federated server. However, the statement SELECT * FROM *NFX1.PERSON* is not allowed from the federated server (except in a pass-through session).

When you create a nickname for a data source object, metadata about the object is added to the global catalog. The query optimizer uses this metadata, and the information in the wrapper, to facilitate access to the data source object. For example, if the nickname is for a table that has an index, the global catalog contains information about the index. The wrapper contains the mappings between the DB2® data types and the data source data types.

Currently, you cannot execute DB2 some utility operations (REORG, REORGCHK, IMPORT, RUNSTATS, and so on) on nicknames.

Related concepts:

- “Column options” on page 14

Related reference:

- “Valid data source objects” on page 13

Valid data source objects

Nicknames identify objects at the data source that you want to access. The following table lists the types of objects that you can create a nickname for in a federated system.

Table 3. Valid data source objects

Data source	Valid objects
DB2 for Linux, UNIX, and Windows	Nicknames, materialized query tables, tables, views
DB2 for z/OS and OS/390	Tables, views
DB2 for iSeries	Tables, views
DB2 for VM and VSE	Tables, views
Informix	Tables, views, synonyms
Microsoft SQL Server	Tables, views
ODBC	Tables, views
Oracle	Tables, views, synonyms
Sybase	Tables, views
Teradata	Tables, views
BLAST	FASTA files indexed for BLAST search algorithms
Documentum	Objects and registered tables in a Documentum Docbase
Entrez	Entrez databases
Extended Search	Files from data sources such as Lotus Notes databases, Microsoft Access, Microsoft Index Server, Web search engines, and LDAP directories.
HMMER	HMM database files (libraries of Hierarchical Markov Models, such as PFAM), that can be searched by HMMER's hmmpfam program.
Microsoft Excel	.xls files (only the first sheet in the workbook is accessed)
Table-structured files	.txt files (text files that meet a very specific format)
XML-tagged files	Sets of items in an XML document

Related concepts:

- “Nicknames and data source objects” on page 12
- “Column options” on page 14

Column options

You can supply the global catalog with additional metadata information about the nicknamed object. This metadata describes values in certain columns of the data source object. You assign this metadata to parameters that are called *column options*. The column options tell the wrapper to handle the data in a column differently than it normally would handle it. The SQL Complier and query optimizer use the metadata to develop better plans for accessing the data.

Column options are used to provide other information to the wrapper as well. For example for XML data sources, a column option is used to tell the wrapper the XPath expression to use when the wrapper parses the column out of the XML document.

With federation, the DB2® server treats the data source object that a nickname references as if it is a local DB2 table. As a result, you can set column options for any data source object that you create a nickname for. Some column options are designed for specific types of data sources and can be applied only to those data sources.

Suppose that a data source has a collating sequence that differs from the federated database collating sequence. The federated server typically would not sort any columns containing character data at the data source. It would return the data to the federated database and perform the sort locally. However, suppose that the column is a character data type (CHAR or VARCHAR) and contains only numeric characters (’0’,’1’,...,’9’). You can indicate this by assigning a value of ’Y’ to the NUMERIC_STRING column option. This gives the DB2 query optimizer the option of performing the sort at the data source. If the sort is performed remotely, you can avoid the overhead of porting the data to the federated server and performing the sort locally.

Attention: The NUMERIC_STRING column option is valid for only relational data sources.

You can define column options in the CREATE NICKNAME and ALTER NICKNAME statements.

Related concepts:

- “Data type mappings” on page 15

Related tasks:

- "Working with nicknames" in the *Federated Systems Guide*

Related reference:

- Appendix D, "Column options for federated systems", on page 379

Data type mappings

The data types at the data source must map to corresponding DB2® data types so that the federated server can retrieve data from data sources. Some examples of default data type mappings are:

- The Oracle® type FLOAT maps to the DB2 type DOUBLE
- The Oracle type DATE maps to the DB2 type TIMESTAMP
- The DB2 for z/OS™ type DATE maps to the DB2 type DATE

For most data sources, the default type mappings are in the wrappers. The default type mappings for DB2 data sources are in the DRDA® wrapper. The default type mappings for Informix™ are in the INFORMIX wrapper, and so forth.

For some nonrelational data sources, you must specify data type information in the CREATE NICKNAME statement. The corresponding DB2 for Linux, UNIX,® and Windows® data types must be specified for each column of the data source object when the nickname is created. Each column must be mapped to a particular field or column in the data source object.

For relational data sources, you can override the default data type mappings, or create mappings when there is no default. For example, you can create a type mapping when a new built-in type is available at the data source, or when there is a user-defined type at the data source that you want to map to.

Attention: You should create new type mappings or modify the default type mappings before you create nicknames. Otherwise nicknames created before the type mapping changes will not reflect the new mappings.

Related concepts:

- "Data type mappings in a federated system" in the *Federated Systems Guide*

Function mappings

For the federated server to recognize a data source function, the function must be mapped against an existing counterpart function in DB2® for Linux, UNIX® and Windows. DB2 Information Integrator supplies default mappings between existing built-in data source functions and built-in DB2 counterpart functions. For most data sources, the default function mappings are in the wrappers. The default function mappings to DB2 for z/OS™ and OS/390® functions are in the DRDA® wrapper. The default function mappings to Sybase functions are in the CTLIB and DBLIB wrappers, and so forth.

For relational data sources, you can create a function mapping when you want to use a data source function that the federated server does not recognize. The mapping that you create is between the data source function and a DB2 counterpart function at the federated database. Function mappings are typically used when a new built-in function or a new user-defined function become available at the data source. Function mappings are also used when a DB2 counterpart function does not exist.

Related concepts:

- "Function mappings in a federated system" in the *Federated Systems Guide*
- "Index specifications" on page 16

Index specifications

When you create a nickname for a data source table, information about any indexes that the data source table has is added to the global catalog. The query optimizer uses this information to expedite the processing of distributed requests. The catalog information about a data source index is a set of metadata, and is called an *index specification*. A federated server does not create an index specification when you create a nickname for:

- A table that has no indexes
- A view, which typically does not have any index information stored in the remote catalog
- A data source object that does not have a remote catalog from which the federated server can obtain the index information

Suppose that a table acquires a new index, in addition to the ones it had when the nickname was created. Because index information is supplied to the global catalog at the time the nickname is created, the federated server is unaware of the new index. Similarly, when a nickname is created for a view, the federated server is unaware of the underlying table (and its indexes) from which the view was generated. In these circumstances, you can supply the necessary index information to the global catalog. You can create an index specification for tables that have no indexes. The index specification tells the query optimizer which column or columns in the table to search on to find data quickly.

Related concepts:

- "Index specifications in a federated system" in the *Federated Systems Guide*

Chapter 3. Planning for federated data source configuration

The following sections provide information you can use to help you plan your federated system.

Federated object naming rules

As with other DB2 objects, there are rules for naming federated database objects.

Federated database objects include:

- Function mappings
- Index specifications
- Nicknames
- Servers
- Type mappings
- User mappings
- Wrappers

Federated object names 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)

Federated object names cannot begin with a number or with the underscore character.

Federated object names can also include the following characters:

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

Federated object names cannot exceed 128 bytes.

Options (such as server options and nickname options) and option settings are limited to 255 bytes.

Names without quotation marks are converted to uppercase.

Related concepts:

- "Naming rules in an NLS environment" in the *Administration Guide: Planning*
- "Naming rules in a Unicode environment" in the *Administration Guide: Planning*

Related reference:

- "Preserving case-sensitive values in a federated system" on page 20

Preserving case-sensitive values in a federated system

In a federated system you occasionally need to specify values, such as user IDs and passwords, that are case-sensitive at the data source. 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 REMOTE_AUTHID and REMOTE_PASSWORD.
- 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 don't have to remember the required case for each data source. You can type the values in any case and they will be converted automatically.

Information about server options and their valid settings are discussed in separate topics.

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:

- Suppose the SQL statement contains double quotation marks, but no single quotation marks. Then you enclose the statement in single quotation marks. For example, if you want to issue this SQL statement:

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

  You enter the following text at the UNIX command prompt

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

- Suppose the SQL statement contains single quotation marks, but no double quotation marks. Then you enclose the 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') "
  ```

- Suppose the SQL statement contains both single and double quotation marks, then you enclose the statement in double quotation marks and precede all double quotes in the statement with a backslash. For example, if you want 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')"
  ```

Note: The above examples assume you are entering SQL statements from the UNIX command prompt and 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 not precede double quotation marks with a backslash.

From a Windows operating system command prompt:

On Windows, precede each quotation mark with a backward slash. For example, suppose you want to create the nickname `NICK1` for a Microsoft SQL Server table. The table resides in the `NORBASE` database. The schema is `my_schema` and the table is `weekly_salary`.

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 enter the value from the DB2 command line prompt (CLP) or you specify the value in an application program, you do not need the single quotation marks or the backslashes. Using the example above, at the DB2 command prompt you type:

```
CREATE NICKNAME nick1
    FOR norbase."my_schema"."weekly_salary"
```

Related reference:

- Appendix B, “Server options for federated systems”, on page 367

Update data source statistics

The federated database relies on catalog statistics for nicknamed objects to optimize query processing. These statistics are gathered when you create a nickname for a data source object 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 useful to the query optimizer is read from the data source catalogs and put into the global catalog on the federated server. Because some or all of the data source catalog information might be used by the query optimizer, it is advisable to update statistics (using the data source command equivalent to RUNSTATS) at the data source before you create a nickname.

The federated database retrieves remote statistical information for an object only once when you create a nickname for the object. If the remote source updates its catalog statistics for a remote object after your create the nickname, the changed statistical information is not propagated to the federated server’s global catalog. To make sure that the global catalog on the federated server reflects current statistics for the remote object, you must drop and recreate the nickname.

Action: Identify objects at the data sources that you want to include in the federated server. These will be objects that you will create nicknames for. Decide which of these data sources you can update the statistics for and list those data sources in the data source statistics table in the planning checklist.

Plan the data type mappings

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

There are two kinds of mappings between data source data types and federated database data types: forward type mappings and reverse type mappings. In a *forward type mapping*, the mapping is from a remote type to a comparable local type. A *reverse type mapping* is used with transparent DDL. In a reverse type mapping, the mapping is from a local type to a comparable remote type. Additional information about the two kinds of data type mappings is discussed in separate topics.

DB2 for UNIX and Windows uses data type mappings to determine what DB2-supported data types should be defined for columns in a data source object. Default data type mappings are built into the data source wrappers.

However, your applications might require data type mappings that are different than the default 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 will have to alter each nickname to reflect the new mapping or drop and re-create the nicknames.

Note: 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.

Action: 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.

Related concepts:

- "Data type mappings" on page 15
- "Tuning query processing" in the *Federated Systems Guide*

Related reference:

- Appendix G, "Default forward data type mappings", on page 387
- Appendix H, "Default reverse data type mappings", on page 407

Plan the function mappings

DB2 for UNIX and Windows supplies default function 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. For some nonrelational data sources, you cannot alter the default function mappings.

To use a data source function that the federated server does not recognize, you must create a function mapping. The mapping you create is between the data source function and a counterpart function at the federated database. Function mappings are typically used when a new built-in function or a new user-defined function becomes available at the data source.

Function mappings are also used when a DB2 counterpart function does not exist. In this situation, before you create the function mapping you will have to create a function template in the federated database.

Action: 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.

Related concepts:

- "Function mappings" on page 16

Plan the user mappings

When a federated server needs to pushdown a request to a data source, the server must first establish a connection to the data source. The server does this by using a valid user ID and password to that data source. 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*.

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.

Action: 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.

Choose the correct wrapper

Some data sources have more than one wrapper that you can use. The one you choose might depend on the version of the data source clients software that you are using. Or it might depend on the operating system that you have on your federated server.

For example there are two wrappers you can use with Oracle data sources: the SQLNET wrapper and the NET8 wrapper. Suppose that you are using Oracle Version 8, and the operating system on your federated server is Windows NT. Originally you create the SQLNET wrapper. Later you learn that the SQLNET wrapper does not support LOB data types, but that the NET8 wrapper does support LOBs. To take advantage of the LOB support in the NET8 wrapper, you will have to drop the SQLNET wrapper and create the NET8 wrapper.

Note: The NET8 wrapper requires a more recent version of the Oracle client than the SQLNET wrapper.

There are significant cascading consequences when you drop a wrapper. Other objects in the federated system are impacted:

- All server definitions, user-defined functions mappings, and user-defined data type mappings that are dependent on the dropped wrapper are also dropped.

- All user-defined function mappings, nicknames, user-defined data type mappings, and user mappings that are dependent on the dropped server definition are also dropped.
- Any index specifications dependent on the dropped nicknames are also dropped.
- Any federated views dependent on these nicknames are marked inoperative.
- All applications dependent on the dropped objects and inoperative views are invalidated.

DB2 Relational Connect provides multiple wrappers for Oracle, Microsoft SQL Server, and Sybase data sources. The distinctions between the wrappers is discussed in the configuration topics for each data source.

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

Related concepts:

- "Wrappers and wrapper modules" on page 9

Related tasks:

- "Adding Microsoft SQL Server data sources to a federated server" in the *Federated Systems Guide*
- "Adding Oracle data sources to a federated server" in the *Federated Systems Guide*
- "Adding Sybase data sources to a federated server" in the *Federated Systems Guide*

Checklist for planning your federated system configuration

You can make the federated system configuration easier by following this planning checklist. This checklist guides you in ways to optimize the federated system configuration.

Checklist: Federated object naming rules

Are you familiar with the naming rules for federated objects?

See the related links at the end of this section to locate information about the federated object naming rules.

Checklist: Preserving case-sensitive values

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

Data source	Server name (in the server definition)	Setting for the FOLD_ID server option	Setting for the FOLD_PW server option

Checklist: Data source statistics

In the following table, list the data sources that will be part of your federated system. Indicate which data sources you will update the statistics for.

Table 5. Planning checklist: Data sources statistics to update for the federated system

Data source	Maintains catalog information? (Y/N)	Will update statistics for this data source? (Y/N)	Data source utility name used to update statistics
DB2 for UNIX and Windows	Y	Y	RUNSTATS

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

Data source	Server name (in the server definition)	Data source data type	DB2 for UNIX and Windows data type

Table 6. Planning checklist: Data type mappings to create for the federated system (continued)

Data source	**Server name (in the server definition)**	**Data source data type**	**DB2 for UNIX and Windows data type**

Checklist: User mappings

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

		Data source ________	**Data source** ________	**Data source** ________
User name	DB2 for UNIX and Windows user ID	User ID	User ID	User ID

Checklist: Wrappers

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

Table 8. Planning checklist: Wrappers to create for the federated system

Data source	**Default wrapper name(s)**	**Wrapper to create**
DB2 Universal Database™ for UNIX and Windows® DB2 Universal Database for z/OS and OS/390® DB2 Universal Database for iSeries DB2 Server for VM and VSE	DRDA	
Informix	INFORMIX	

Table 8. Planning checklist: Wrappers to create for the federated system (continued)

Data source	**Default wrapper name(s)**	**Wrapper to create**
Oracle	SQLNet Net8	
Microsoft® SQL Server	DJXMSSQL3 MSSQLODBC3	
ODBC	none	
OLE DB	OLEDB	
Sybase	CTLIB DBLIB	
BLAST	none	
Documentum	none	
Microsoft Excel	none	
Table-structured files	none	
XML	none	

Chapter 4. Configuring access to DB2 family data sources

This chapter explains how to configure your federated server to access data that is stored in DB2 family databases. These databases include DB2 for UNIX and Windows, DB2 for z/OS and OS/390, DB2 for iSeries, and DB2 Server for VM and VSE.

This chapter lists the tasks that you need to perform, shows examples of the SQL statements that you need to issue, and provides tuning and troubleshooting information that you can use when you set up the configuration to DB2 family data sources.

Adding DB2 family data sources to federated servers

Configuring the federated server to access DB2 data sources involves supplying the server with information about the DB2 data sources and objects that you want to access.

You can configure access to DB2 data sources through the DB2 Control Center, through the DB2 Command Center, or through the DB2 command line processor.

The advantage of using the DB2 Control Center is that you do not need to type each statement and command. The DB2 Control Center provides the easiest method to configure access quickly to DB2 data sources. However, you cannot use the DB2 Control Center to initiate the following configuration tasks:

- Cataloging the node
- Cataloging the remote database
- Testing the connection to the data source server to validate the server definition and the user mappings
- Adding or droping column options

Prerequisites:

- Access to the DB2 Command Center or the DB2 command line processor
- A federated server and database that are set up to access DB2 family data sources

Restriction:

You cannot create a nickname for a DB2 data source alias if you are accessing data that is stored in DB2 for UNIX and Windows, Version 8.1.

Procedure:

To add a DB2 data source to a federated server:

1. Catalog the node.
2. Catalog the remote database.
3. Register the wrapper.
4. Register the server definition and set the server options.
5. Create the user mappings.
6. Test the connection to the DB2 server.
7. Register the nicknames for tables and views.

Related concepts:

- “Fast track to configuring your data sources” on page 1

Related tasks:

- “Checking the federated server setup” in the *DB2 Information Integrator Installation Guide*
- “Cataloging a node entry in the federated node directory” on page 32
- “Tuning and troubleshooting the configuration to DB2 family data sources” on page 43

Cataloging a node entry in the federated node directory

Cataloging a node entry in the federated node directory is part of the larger task of adding DB2 family data sources to federated servers.

To point to the location of the DB2 data source, catalog an entry in the node directory of the federated server. The federated server uses this 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:

1. Determine the communication protocol that you will be using.
2. Issue the appropriate command to catalog the node entry.
   - If your communication protocol is Transmission Control Protocol/Internet Protocol (TCP/IP), 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.

- If your communication protocol is SNA, issue the **CATALOG APPC NODE** command.

  For example:

  ```
  CATALOG APPC NODE DB2NODE REMOTE DB2CPIC SECURITY PROGRAM
  ```

  The *DB2NODE* value is the name that you assign to the node that you are cataloging. REMOTE *DB2CPIC* is the SNA partner logical unit (LU) name of the remote partner node. SECURITY *PROGRAM* specifies that both a user name and a password are to be included in the allocation request that is sent to the partner LU.

The next task in this sequence of tasks is cataloging the remote database in the federated system database directory.

Related tasks:

- "Cataloging the remote database in the federated system database directory" on page 33

Cataloging the remote database in the federated system database directory

Cataloging the remote database in the federated system database directory is part of the larger task of adding DB2 family data sources to federated servers.

You specify which DB2 data source database that the federated server connects to 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:

1. Use the Client Configuration Assistant (CCA).

   For federated servers on UNIX, you can alternatively use the **CATALOG DATABASE** command. 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. 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.

2. 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.

The next task in this sequence of tasks is registering the DB2 wrapper.

Related tasks:
- "Cataloging a node entry in the federated node directory" on page 32
- "Registering the DB2 wrapper" on page 34

Registering the DB2 wrapper

Registering the DB2 wrapper is part of the larger task of adding DB2 family data sources to federated servers.

To specify the wrapper that will be used to access DB2 data sources, issue the CREATE WRAPPER statement. Every DB2 Server Edition (Enterprise, Personal, and Workgroup) includes one wrapper called DRDA for the DB2 family.

Procedure:

To specify the wrapper that you want to use to access DB2 family data sources, issue the CREATE WRAPPER statement.

For example:

```
CREATE WRAPPER DRDA
```

Recommendation: Use the default wrapper name called DRDA. When you register the wrapper by using the default name, the federated server automatically takes the default library name that is associated with that wrapper name. If the wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name you choose. If you use a name that is different from the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement. Suppose that you have a federated server running on AIX, and you decide to use a wrapper name that is different from the default name. You must include the LIBRARY parameter in the CREATE WRAPPER statement.

For example:

```
CREATE WRAPPER mywrapper LIBRARY 'libdb2drda.a'
```

The *mywrapper* value is the name that you give to the wrapper instead of using the default wrapper name.

The following table lists the wrapper library names for DB2 by operating system:

Table 9. DB2 wrapper library names

Operating system on your federated server	**Wrapper library name**
AIX	libdb2drda.a
Solaris Operating Environment	libdb2drda.so
HP-UX	libdb2drda.sl
Linux	libdb2drda.so
Windows NT and Windows 2000	db2drda.dll

The next task in this sequence of tasks is registering the server definitions for a DB2 data source.

Related tasks:

- "Registering the server definitions for a DB2 data source" on page 35

Related reference:

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*

Registering the server definitions for a DB2 data source

Registering the server definitions for a DB2 data source is part of the larger task of adding DB2 family data sources to federated servers.

In the federated database, you must define each DB2 server that you want to access. When you register the server definition, the federated server connects to the DB2 server and binds packages to the database. Because the information for authorization and password 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, issue the CREATE SERVER statement.

For example:

```
CREATE SERVER server_name TYPE DB2/ZOS VERSION 6 WRAPPER DRDA
       AUTHORIZATION "name1" PASSWORD "passwd1"
        OPTIONS (DBNAME 'db_name')
```

The name that you assign to a server must be unique. Duplicate server names are not allowed.

The VERSION option that you specify is the version of the DB2 database server that you want to access. The supported versions are:

- DB2 for UNIX and Windows, Version 6, Version 7.1, Version 7.2, and Version 8.1
- DB2 for z/OS and OS/390, Version 5 or later
- DB2 for iSeries, Version 4 or later

The name of the WRAPPER parameter must be the name that you specified in the CREATE WRAPPER statement.

Although the database name is specified as an option in the CREATE SERVER statement, it 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.

The next task in this sequence of tasks is creating the user mapping for a DB2 data source.

Related tasks:

- "Creating the user mapping for a DB2 data source" on page 38

Related reference:

- “CREATE SERVER statement” in the *SQL Reference, Volume 2*
- “CREATE SERVER statement - Examples for DB2 wrapper” on page 37

CREATE SERVER statement - Examples for DB2 wrapper

This topic provides examples that show you how to use the CREATE SERVER statement to register servers for wrappers on DB2 family data sources. This topic includes a complete example, which shows how to create a server with all required parameters, and an example with additional server options.

Complete example:

The following example shows you how to create a server definition for a DB2 wrapper by using the CREATE SERVER statement:

```
CREATE SERVER DB2SERVER TYPE DB2/ZOS VERSION 6 WRAPPER DRDA
       AUTHORIZATION "spalten" PASSWORD "db2guru"
        OPTIONS (DBNAME 'CLIENTS390')
```

DB2SERVER
: A name that you assign to the DB2 database server. This name must be unique. Duplicate server names are not allowed.

TYPE *DB2/ZOS*
: Specifies the type of data source server to which you are configuring access.

VERSION *6*
: 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.

This database name 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 following example shows a server definition with the CPU_RATIO option.

```
CREATE SERVER DB2SERVER TYPE DB2/ZOS VERSION 6 WRAPPER DRDA
       AUTHORIZATION "spalten" PASSWORD "db2guru"
        OPTIONS (DBNAME 'CLIENTS390', CPU_RATIO '0.001')
```

If you set the CPU_RATIO option to ’0.001’, this indicates the CPU at the remote data source 1000 times more available capacity than the federated server.

Related tasks:

- “Registering the server definitions for a DB2 data source” on page 35

Related reference:

- “CREATE SERVER statement” in the *SQL Reference, Volume 2*
- Appendix F, “Valid server types in SQL statements”, on page 383

Creating the user mapping for a DB2 data source

Creating the user mapping for a DB2 data source is part of the larger task of adding DB2 family data sources to federated servers.

When you attempt to access a DB2 server, the federated server establishes a connection to the data source 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. Create a user mapping for each user ID that will access the federated system to send distributed requests.

Procedure:

To map the local user ID to the DB2 server user ID and password, issue a CREATE USER MAPPING statement.

For example:

```
CREATE USER MAPPING FOR USERID SERVER DB2SERVER
       OPTIONS (REMOTE_AUTHID 'remote_id', REMOTE_PASSWORD 'remote_password')
```

The REMOTE_AUTHID is the connect authorization ID, not the bind authorization ID.

The next task in this sequence of tasks is testing the connection to the DB2 data source server.

Related tasks:

- "Testing the connection to the DB2 data source server" on page 40

Related reference:

- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*
- "CREATE USER MAPPING statement - Examples for DB2 wrapper" on page 39

CREATE USER MAPPING statement - Examples for DB2 wrapper

This topic provides examples that show you how to use the CREATE USER MAPPING statement to map a local user ID to the DB2 server user ID and password. This topic includes a complete example with all 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 local user ID to the DB2 server user ID:

```
CREATE USER MAPPING FOR DB2USER SERVER DB2SERVER
       OPTIONS (REMOTE_AUTHID 'db2admin', REMOTE_PASSWORD 'day2night')
```

DB2USER
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 *'db2admin'*
Specifies the connect authorization user ID at the DB2 family data source server to which you are mapping *DB2USER*. 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 *'db2admin'*. 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 'db2admin', 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.

Related tasks:

- "Creating the user mapping for a DB2 data source" on page 38

Related reference:

- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*

Testing the connection to the DB2 data source server

Testing the connection to the DB2 data source server is part of the larger task of adding DB2 family data sources to federated servers.

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:

1. Open a pass-through session to issue an SQL SELECT statement on the DB2 system tables.

   For example:

   - On DB2 for z/OS and OS/390:

     ```
     SET PASSTHRU server_name
     SELECT count(*) FROM sysibm.systables
     SET PASSTHRU RESET
     ```

   - On DB2 for iSeries:

     ```
     SET PASSTHRU remote_server_name
     SELECT count(*) FROM qsys2.systables
     SET PASSTHRU RESET
     ```

   If the SQL SELECT statement returns a count, your server definition and your user mapping are set up properly.

2. If the SQL SELECT statement returns an error, you might need to:

   - Check the remote server to make sure that it is started.

- Check the listener on the remote server to make sure that it is configured for incoming connections.
- Check your user mapping to make sure that the settings for the REMOTE_AUTHID and REMOTE_PASSWORD options are valid for connections to the DB2 server.
- Check the DB2 catalog entries for the node and database.
- Check the settings of your DB2 federated variables to verify that you can access the remote DB2 server. These variables include the system environment variables, the db2dj.ini variables, and the DB2 Profile Registry (db2set) DB2COMM variable.
- Check your server definition. If necessary, drop the server definition, and create it again.
- Check your user mapping. If necessary, alter the user mapping, or create another user mapping.

The next task in this sequence of tasks is registering nicknames for DB2 tables and views.

Related concepts:

- "Server definitions and server options" on page 10

Related tasks:

- "Checking the data source environment variables" in the *DB2 Information Integrator Installation Guide*
- "Registering nicknames for DB2 tables and views" on page 41

Related reference:

- "ALTER USER MAPPING statement" in the *SQL Reference, Volume 2*

Registering nicknames for DB2 tables and views

Registering nicknames for DB2 tables and views is part of the larger task of adding DB2 family data sources to federated servers.

The federated database relies on catalog statistics for nicknamed objects to optimize query processing. These statistics are gathered when you register a nickname for a data source object 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 useful to the optimizer is read from the data source catalogs and put into the global catalog on the federated server. Because some or all of the data source catalog information might be used by the optimizer, it is advisable to update statistics (using the data source command equivalent to the **RUNSTATS** command) at the data source before you register a nickname.

Use the CREATE NICKNAME statement to register a nickname for a view or table that is located at your DB2 family data source. Use these nicknames, instead of the names of the data source objects, when you query the DB2 family data source.

Procedure:

To register a nickname, issue the CREATE NICKNAME statement.

For example:

```
CREATE NICKNAME DB2NICKNAME FOR DB2SERVER.remote_schema.remote_table
```

Nicknames can be up to 128 characters in length.

Repeat this step for each DB2 table or view for which you want to register a nickname.

When you register the nickname, the federated server will use the connection to query the data source catalog. This query tests your connection to the data source by using the nickname. If the connection does not work, you will receive an error message.

Related reference:

- "RUNSTATS Command" in the *Command Reference*
- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*
- "CREATE NICKNAME statement - Examples for DB2 wrapper" on page 42

CREATE NICKNAME statement - Examples for DB2 wrapper

This topic provides an example that shows you how to use the CREATE NICKNAME statement to register a nickname for a DB2 table or view that you want to access.

The following example shows a CREATE NICKNAME statement:

```
CREATE NICKNAME DB2SALES FOR DB2SERVER.SALESDATA.EUROPE
```

DB2SALES
: A unique nickname that is used to identify the DB2 table or view.

Note: The nickname is a two-part name that includes 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 creating the nickname.

DB2SERVER.SALESDATA.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.
- *SALESDATA* is the name of the remote schema 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.

Tuning and troubleshooting the configuration to DB2 family data sources

After you set up the configuration to DB2 data sources, you might want to modify the configuration to improve performance. For example, you might want to set the DB2_DJ_COMM DB2 profile registry variable to improve performance when the DB2 data source is accessed.

Improving performance by setting the DB2_DJ_COMM variable (UNIX)

If you find that it takes a long time to access the DB2 data source server, you can improve the performance by setting the DB2_DJ_COMM profile registry variable. When you set the DB2_DJ_COMM variable, the federated server loads the wrapper upon initialization rather than when you attempt to access the data source.

Procedure:

To set the DB2_DJ_COMM variable:

1. Set the DB2_DJ_COMM variable to the wrapper library that corresponds to the wrapper that you specified. Use the commands in the following table to set the DB2_DJ_COMM variable.

Table 10. Commands to set the DB2_DJ_COMM variable for DB2 data sources

Federated server operating system	Command
AIX	DB2_DJ_COMM= ’libdb2drda.a’
Solaris Operating Environment	DB2_DJ_COMM= ’libdb2drda.so’
HP-UX	DB2_DJ_COMM= ’libdb2drda.sl’
Linux	DB2_DJ_COMM= ’libdb2drda.so’

Table 10. Commands to set the DB2_DJ_COMM variable for DB2 data sources (continued)

Federated server operating system	Command
Windows NT and Windows 2000	DB2_DJ_COMM= ’db2drda.dll’

Use the **db2set** command to set the DB2_DJ_COMM variable. For example, if the federated server operating system is AIX, the command would be:

```
db2set DB2_DJ_COMM='libdb2drda.a'
```

2. Export the DB2_DJ_COMM variable. For example:

```
export DB2_DJ_COMM
```

3. Issue the following commands to recycle the DB2 instance:

```
db2stop
db2start
```

By recycling the DB2 instance, you ensure that the DB2 instance accepts the variable changes that you made.

Related tasks:

- “Adding DB2 family data sources to federated servers” on page 31

Related reference:

- “db2set - DB2 Profile Registry Command” in the *Command Reference*

Chapter 5. Configuring access to Informix data sources

This chapter explains how to configure your federated server to access data that is stored in Informix data sources.

This chapter lists the tasks that you need to perform, shows examples of the SQL statements that you need to issue, and provides tuning and troubleshooting information that you can use when you set up the configuration to Informix data sources.

Adding Informix data sources to federated servers

Configuring the federated server to access Informix data sources involves supplying the server with information about the Informix data sources and objects that you want to access.

You can configure access to Informix data sources through the DB2 Control Center, through the DB2 Command Center, or through the DB2 command line processor.

The advantage of using the DB2 Control Center is that you do not need to type each statement and command. The DB2 Control Center provides the easiest method to configure access quickly to Informix data sources. However, you cannot use the DB2 Control Center to initiate the following configuration tasks:

- Setting up and testing the Informix client configuration file
- Testing the connection to the Informix server to validate the server definition and user mappings
- Adding or dropping column options

Prerequisites:

- Access to the DB2 Command Center or the DB2 command line processor.
- A federated server and database that are set up to access Informix data sources.
- The Informix Client SDK software that is installed and configured on the federated server.
- The proper setup of the system environment variables, db2dj.ini variables (including code page conversion variables), and DB2 Profile Registry (db2set) variables. The variables are: INFORMIXDIR, INFORMIXSERVER, CLIENT_LOCALE (optional), DB_LOCALE (optional), DBNLS (optional),

and INFORMIXSQLHOSTS (optional). You must set the INFORMIXSQLHOSTS variable only if the sqlhosts file or registry is not in the default location.
- On AIX federated servers, the AIX Base Application Development Math Library. You can determine if the Library is installed by issuing the AIX command **lslpp -l bos.adt.libm**.

Procedure:

To add an Informix data source to a federated server:
1. Set up and test the Informix client configuration file.
2. Register the wrapper.
3. Register the server definition.
4. Create the user mappings.
5. Test the connection to the Informix server.
6. Register nicknames for Informix tables, views, and synonyms.

Related concepts:
- "Fast track to configuring your data sources" on page 1

Related tasks:
- "Checking the data source environment variables" in the *DB2 Information Integrator Installation Guide*
- "Setting up and testing the Informix client configuration file" on page 46
- "Tuning and troubleshooting the configuration to Informix data sources" on page 57

Setting up and testing the Informix client configuration file

Setting up and testing the Informix client configuration file is part of the larger task of adding Informix data sources to federated servers.

The client configuration file is used to connect to Informix, using the client libraries that are installed on the federated server. This file specifies the location of each Informix database server and type of connection (protocol) for the database server.
- On UNIX operating systems, the default name is `$INFORMIXDIR/etc/sqlhosts`. The `sqlhosts` file resides on each installation of the Informix client SDK.
- On Windows operating systems, 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. Create the `sqlhosts` file or set up the registry with the Informix Setnet32 utility.

   You can copy the `sqlhosts` file or registry from another system that has Informix Connect or Informix Client SDK installed. You can also configure the Informix Client SDK on the federated server to connect to an Informix server, which creates the `sqlhosts` file or registry. The federated server will use the `sqlhosts` that is in the Informix SDK directory or the Windows registry.
2. Verify the location of the `sqlhosts` file or registry.
   - On UNIX operating systems, the `sqlhosts` file is located in the `$INFORMIXDIR/etc/sqlhosts` directory.
   - On Windows operating systems, the sqlhosts information is kept in the following key in the Windows registry:

     ```
     HKEY_LOCAL_MACHINE\SOFTWARE\INFORMIX\SQLHOSTS
     ```
3. If the `sqlhosts` file or registry is not in the default location, set the environment variable INFORMIXSQLHOSTS.
   a. On UNIX operating systems, set the environment variable INFORMIXSQLHOSTS to the fully-qualified name of the `sqlhosts` file. On Windows operating systems, set the environment variable INFORMIXSQLHOSTS to the name of the Windows computer that stores the registry.
   b. Issue the following commands to recycle the DB2 instance and ensure that the environment variable is set in the program:

      ```
      db2stop
      db2start
      ```
4. Test the connection to ensure that the client software is able to connect to the Informix server. If the Informix **dbaccess** tool is on the federated server, use this tool to test the connection. Otherwise, run the Informix demo program to test the client setup.

The next task in this sequence of tasks is registering the Informix wrapper.

Related tasks:

- "Registering the Informix wrapper" on page 48
- "Tuning and troubleshooting the configuration to Informix data sources" on page 57

Registering the Informix wrapper

Registering the Informix wrapper is part of the larger task of adding Informix data sources to federated servers.

To specify the wrapper that will be used to access Informix data sources, use the CREATE WRAPPER statement. Every DB2 Server Edition (Enterprise, Personal, Workgroup) includes one wrapper for Informix called INFORMIX.

Procedure:

To specify the wrapper that you want to use to access Informix data sources, issue the CREATE WRAPPER statement.

For example:

```
CREATE WRAPPER INFORMIX
```

Recommendation: Use the default wrapper name INFORMIX. When you register the wrapper using the default name, the federated server automatically takes the default library name that is associated with that wrapper name. 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 use a name that is different from one of the default names, you must include the LIBRARY parameter in the CREATE WRAPPER statement.

Suppose that you have a federated server running on AIX, and you decide to use a wrapper name that is not one of the default names. You must include the LIBRARY parameter in the CREATE WRAPPER statement.

For example:

```
CREATE WRAPPER mywrapper LIBRARY 'libdb2informix.a'
```

The wrapper library names for Informix are:

Table 11. Informix wrapper library names

Operating system on your federated server	Wrapper library name
AIX	libdb2informix.a
HP-UX	libdb2informix.sl
Linux	libdb2informix.so
Solaris Operating Environment	libdb2informix.so
Windows NT and Windows 2000	db2informix.dll

The next task in this sequence of tasks is registering the server definitions for an Informix data source.

Related tasks:

- "Registering the server definitions for an Informix data source" on page 49

Related reference:

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*

Registering the server definitions for an Informix data source

Registering the server definitions for an Informix data source is part of the larger task of adding Informix data sources to federated servers.

In the federated database, you must define each Informix server that you want to access. You must first locate the node name of the Informix data source, and then use this node name when you register the server.

Procedure:

To register a server definition for an Informix data source:

1. Locate the node name in the Informix sqlhosts file or registry.

   Example sqlhosts file:

   ```
   inf724 onsoctcp anaconda inmx724
   inf731 onscotcp boa ifmx731
   inf92 onsoctcp python ifmx92
   ```

   The first value in each line is the *node_name*, such as `inf724`.

   The second value in each line is the *nettype*, or type of connection. In this example `onscotcp` 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 `inmx724`. The service name field depends on the *nettype* listed in the second value.

   Although the *node_name* is specified as an option in the CREATE SERVER SQL statement, it is required for Informix data sources.

   For more information about the format of this file and the meaning of these fields, see the Informix manual *Administrators Guide for Informix Dynamic Server*.

2. Issue the CREATE SERVER statement.

   For example:

```
CREATE SERVER server_name TYPE informix VERSION 9 WRAPPER INFORMIX
      OPTIONS (NODE 'node_name', DBNAME 'db_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 mapping for an Informix data source.

Related tasks:

- "Creating the user mapping for an Informix data source" on page 52

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*
- Appendix B, "Server options for federated systems", on page 367
- "CREATE SERVER statement - Examples for Informix wrapper" on page 50

CREATE SERVER statement - Examples for Informix wrapper

This topic provides several examples that show you how to use the CREATE SERVER statement to register servers for the Informix wrapper. This topic includes a complete example, which shows how to register a server with required parameters, and an example with additional server options.

Complete example:

The following example shows you how to register a server definition for an Informix wrapper by using the CREATE SERVER statement:

```
CREATE SERVER asia TYPE informix VERSION 9 WRAPPER INFORMIX
      OPTIONS (NODE 'abc', DBNAME 'sales', IUD_APP_SVPT_ENFORCE 'N')
```

asia A name you assign to the Informix database server. This name must be unique. Duplicate server 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 *9*
: The Informix database server version that you want to access. The supported Informix versions are 7, 8, and 9.

WRAPPER *INFORMIX*
: The name you specified in the CREATE WRAPPER statement.

NODE *'abc'*
The name of the node where Informix database server resides. Obtain the node name from the `sqlhosts` file. This value is case sensitive.

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

IUD_APP_SVPT_ENFORCE *'N'*
Specifies whether DB2 federated system should enforce detecting or building of application savepoint statements. Informix does not support application savepoint statements. When set to 'N', the federated server will allow INSERT, UPDATE, or DELETE statements on nicknames for Informix data sources.

The IUD_APP_SVPT_ENFORCE server option must be set to 'N' to enable replication to or from Informix data sources.

Although the application savepoint enforcement is specified as an option in the CREATE SERVER statement, it is required for Informix data sources.

Server options example:

When you create the server definition, you can specify additional server options in the CREATE SERVER statement. These server options include general server options and Informix-specific server options.

The following example shows an Informix server definition with additional server options:

```
CREATE SERVER asia TYPE informix VERSION 9 WRAPPER INFORMIX
      OPTIONS (NODE 'abc', DBNAME 'sales', FOLD_ID 'N', FOLD_PW 'N')
```

When the federated server connects to a data source, it tries to connect using all possible combinations of upper and lower case for the user ID and password, as well as the current case. 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 being locked out. You can prevent lock outs by specifying values for the `FOLD_ID` and `FOLD_PW` server options.

For example, you can set the `FOLD_ID` and `FOLD_PW` server options to 'N' (do not fold the user ID or password). If you establish these settings, then you must specify the user ID and password in the correct case. The advantage to setting these options to 'N' is that when an invalid user ID or password is specified, the wrapper will not keep trying the various combinations. This setting reduces the chance of exceeding the maximum number of failed login attempts and the ID getting locked out.

Related tasks:

- "Registering the server definitions for an Informix data source" on page 49

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*

Creating the user mapping for an Informix data source

Creating the user mapping for an Informix data source is part of the larger task of adding Informix data sources to federated servers.

When you attempt to access an Informix server, the federated server establishes a connection to the data source 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. 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 USERID SERVER INFORMIXSERVER
      OPTIONS (REMOTE_AUTHID 'remote_id', REMOTE_PASSWORD 'remote_password')
```

The next task in this sequence of tasks is testing the connection to the Informix server.

Related tasks:

- "Testing the connection to the Informix server" on page 54

Related reference:

- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*
- "CREATE USER MAPPING statement - Examples for Informix wrapper" on page 53

CREATE USER MAPPING statement - Examples for Informix wrapper

This topic provides examples that show you how to 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 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 (*VINCENT*) to an Informix server user ID and password (*'vinnie'* and *'close2call'*):

```
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 an Informix server.

SERVER *asia*
: Specifies the name of the Informix server that you registered in the CREATE SERVER statement.

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.

REMOTE_PASSWORD *'close2call'*
: Specifies the password 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.

Special register example:

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')
```

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.

Related tasks:

- “Creating the user mapping for an Informix data source” on page 52

Related reference:

- “CREATE USER MAPPING statement” in the *SQL Reference, Volume 2*

Testing the connection to the Informix server

Testing the connection to the Informix server is part of the larger task of adding Informix data sources to federated servers.

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:

1. Open a pass-through session to issue an SQL SELECT statement on the Informix system tables.

   For example:

   ```
   SET PASSTHRU server_name
   SELECT count(*) FROM informix.systables
   SET PASSTHRU RESET
   ```

   If the SQL SELECT statement returns a count, your server definition and your user mapping are set up properly.
2. If the SQL SELECT statement returns an error, you might need to:
   - Check the Informix server to make sure that it is configured for incoming connections.
   - Check your user mapping to make sure that the settings for the REMOTE_AUTHID and REMOTE_PASSWORD options are valid for the connections to the Informix server. Alter the user mapping, or create another user mapping as necessary.
   - Check the Informix Client SDK software on the DB2 federated server to make sure that it is installed and configured correctly to connect to the Informix server.
   - Check the settings of your DB2 federated variables to verify that they are correct for the Informix server. These variables include the system environment variables, the db2dj.ini variables, and the DB2 Profile Registry (db2set) variable.
   - Check your server definition. If necessary, drop it and create it again.

The next task in this sequence of tasks is registering nicknames for Informix tables, views, and synonyms.

Related concepts:

- "Server definitions and server options" on page 10

Related tasks:

- "Checking the federated server setup" in the *DB2 Information Integrator Installation Guide*
- "Checking the data source environment variables" in the *DB2 Information Integrator Installation Guide*
- "Registering nicknames for Informix tables, views, and synonyms" on page 55

Related reference:

- "ALTER USER MAPPING statement" in the *SQL Reference, Volume 2*

Registering nicknames for Informix tables, views, and synonyms

Registering nicknames for Informix tables, views, and synonyms is part of the larger task of adding Informix data sources to federated servers.

For each Informix server that you define, 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.

The federated database relies on catalog statistics for nicknamed objects to optimize query processing. These statistics are gathered when you register a nickname for a data source object by 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 that is useful to the optimizer is read from the data source catalogs and placed into the global catalog on the federated server. Because some or all of the data source catalog information might be used by the optimizer, update statistics (using the data source command that is equivalent to the DB2 **RUNSTATS** command) at the data source before you register a nickname.

Procedure:

To register a nickname, issue the CREATE NICKNAME statement.

For example:

```
CREATE NICKNAME informix_name FOR INFOSERVER."remote_schema"."remote.table"
```

Nicknames can be up to 128 characters in length.

Repeat this step for each Informix table, view, or synonym for which you want to create a nickname.

When you create the nickname, DB2 will use the connection to query the data source catalog. This query tests your connection to the data source by using the nickname. If the connection does not work, you will receive an error message.

Related reference:

- "RUNSTATS Command" in the *Command Reference*
- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*
- "CREATE NICKNAME statement - Examples for Informix wrapper" on page 56

CREATE NICKNAME statement - Examples for Informix wrapper

This topic provides an example that shows you how to use the CREATE NICKNAME statement to register a nickname for an Informix table, view, or synonym that you want to access.

This example shows how to specify a remote object for the Informix server under which the nickname is assigned:

```
CREATE NICKNAME JPSALES FOR asia."salesdata"."japan"
```

JPSALES
A unique nickname used to identify the Informix table, view, or synonym.

Note: 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.

asia."salesdata"."japan"
A three-part identifier for the remote object.

- *asia* is the name that you assigned to the Informix database server in the CREATE SERVER statement.
- *salesdata* is the name of the remote schema to which the table, view, or synonym belongs.
- *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.

Related tasks:

- "Registering nicknames for Informix tables, views, and synonyms" on page 55

Related reference:

- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*

Tuning and troubleshooting the configuration to Informix data sources

After you set up the configuration to Informix data sources, you might want to modify the configuration to improve performance. For example, you can set the DB2_DJ_COMM profile registry variable to improve performance when the Informix data source is accessed.

Improving performance by setting the FOLD_ID and FOLD_PW server options

When the federated server connects to a data source, it tries to connect using all possible combinations of upper and lower case for the user ID and password, as well as the current case. The server might make up to nine connect attempts before successfully connecting to the data source server. These attempts can slow down connect times.

Procedure:

To improve performance, specify values for the FOLD_ID and FOLD_PW server options by using the ALTER SERVER OPTION statement.

- Suppose all your Informix user IDs and passwords are in lowercase, then setting FOLD_ID and FOLD_PW to the value L (delimited by single quotes) could improve your connect time. For example:

  ```
  ALTER SERVER TYPE INFORMIX OPTIONS (ADD FOLD_ID 'L')
  ALTER SERVER TYPE INFORMIX OPTIONS (ADD FOLD_PW 'L')
  ```

- Because 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 and the ID getting locked out 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')
  ```

Improving performance by setting the DB2_DJ_COMM variable (UNIX)

If you find that it takes a long time to access the Informix server, you can improve the performance by setting the DB2_DJ_COMM DB2 profile registry variable to load the wrapper when the federated server initializes rather than when you attempt to access the data source.

Procedure:

To set the DB2_DJ_COMM variable:

1. Set the DB2_DJ_COMM variable to the wrapper library that corresponds to the wrapper that you specified. Suppose that your federated server is running AIX . The command to set the DB2_DJ_COMM variable is:

   ```
   db2set DB2_DJ_COMM='libdb2informix.a,libdb2informixF.a,libdb2informixU.a'
   ```

   The following table lists the DB2_DJ_COMM commands with the appropriate library names by operating system.

Table 12. Commands to set the DB2_DJ_COMM variable for Informix data sources

Federated server operating system	Command
AIX	DB2_DJ_COMM= ’libdb2informix.a’
HP-UX	DB2_DJ_COMM= ’libdb2informix.sl’
Linux	DB2_DJ_COMM= ’libdb2informix.so’
Solaris Operating Environment	DB2_DJ_COMM= ’libdb2informix.so’

2. Issue the following commands to recycle the DB2 instance:

   ```
   db2stop
   db2start
   ```

   By recycling the DB2 instance, you ensure that the DB2 instance accepts the variable changes that you made

Related tasks:

- “Adding Informix data sources to federated servers” on page 45

Related reference:

- “db2set - DB2 Profile Registry Command” in the *Command Reference*
- “ALTER SERVER statement” in the *SQL Reference, Volume 2*

Chapter 6. Configuring access to Oracle data sources

This chapter explains how to configure your federated server to access data that is stored in Oracle data sources.

This chapter lists the tasks that you need to perform, shows examples of the SQL statements that you need to issue, and provides tuning and troubleshooting information that you can use when you set up the configuration to Oracle data sources.

Adding Oracle data sources to federated servers

Configuring the federated server to access Oracle data sources involves supplying the server with information about the Oracle data sources and objects that you want to access.

You can configure access to Oracle data sources through the DB2 Control Center, through the DB2 Command Center, or through the DB2 command line processor.

The advantage of using the DB2 Control Center is that you do not need to type each statement and command. The DB2 Control Center provides the easiest method to configure access quickly to Oracle data sources. However, you cannot use the DB2 Control Center to initiate the following configuration tasks:

- Setting up and testing the Oracle client configuration file
- Testing the connection to the Oracle server to validate the server definition and user mappings
- Adding or dropping column options

Prerequisites:

- Access to the DB2 Command Center or the DB2 command line processor.
- A federated server and database that are set up to access Oracle data sources.
- The Oracle client software that is installed and configured on the federated server.
- The proper setup of the system environment variables, db2dj.ini variables, and DB2 Profile Registry (db2set) variables. The variables are: ORACLE_HOME, ORACLE_BASE, ORA_NLS, and TNS_ADMIN.

Procedure:

To add an Oracle data source to a federated server:

1. Set up and test the Oracle client configuration file.
2. Register the wrapper.
3. Register the server definition.
4. Create the user mappings.
5. Test the connection to the Oracle server.
6. Register nicknames for Oracle tables and views.

Related concepts:

- "Fast track to configuring your data sources" on page 1

Related tasks:

- "Checking the data source environment variables" in the *DB2 Information Integrator Installation Guide*
- "Setting up and testing the Oracle client configuration file" on page 60
- "Tuning and troubleshooting the configuration to Oracle data sources" on page 71

Setting up and testing the Oracle client configuration file

Setting up and testing the Oracle client configuration file is part of the larger task of adding Oracle data sources to federated servers.

The client configuration file is used to connect to Oracle databases, using the client libraries that are installed on the federated server. This 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.

Procedure:

To set up and test the Oracle client configuration file:

1. Use the utility that comes with the Oracle client software.

   See the installation documentation from Oracle for more information about using this utility. 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.

   The directory in which the tnsnames.ora file is created depends on the operating system running on your federated server.

- On UNIX operating systems, the default path and name of this file is $ORACLE_HOME/network/admin .
- On Windows operating systems, the default path and name of this file is %ORACLE_HOME%\NETWORK\ADMIN.

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 that is located in the sqllib/cfg directory, and set the TNS_ADMIN environment variable:
      ```
      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 by using the Oracle **sqlplus** tool to ensure that the client software is able to connect to the Oracle server.

The next task in this sequence of tasks is registering the Oracle wrapper.

Related tasks:
- "Registering the Oracle wrapper" on page 61
- "Tuning and troubleshooting the configuration to Oracle data sources" on page 71

Registering the Oracle wrapper

Registering the Oracle wrapper is part of the larger task of adding Oracle data sources to federated servers.

To specify the wrapper that will be used to access Oracle data sources, use the CREATE WRAPPER statement. Two wrappers for Oracle, SQLNET and NET8, are included with DB2 Information Integrator.

Procedure:

To specify the wrapper that you want to use to access Oracle data sources, issue the CREATE WRAPPER statement.

For example:
```
CREATE WRAPPER NET8
```

To determine which wrapper name (SQLNET or NET8) to use with the CREATE WRAPPER statement, see the Related reference at the end of this topic.

The `SQLNET` wrapper uses OCI 7 (Oracle Call Interface) API calls. The `NET8` wrapper uses OCI 8 API calls. If the Oracle 8 or Oracle 9 client is installed, you will experience better performance and functionality by using the `NET8` wrapper. Additionally, the `NET8` wrapper has LOB support. Because the OCI 7 does not support LOB data types, the `SQLNET` wrapper does not support Oracle LOB data types.

- The `SQLNET` wrapper maps Oracle LONG data types to DB2 for UNIX and Windows LOB data types.
- The `NET8` wrapper does not support Oracle LONG data types. It does map Oracle LOB data types to DB2 for UNIX and Windows LOB data types.

Recommendation: Use the default wrapper name (SQLNET or NET8). When you register the wrapper by using one of the default names, the federated server automatically takes the default library name associated with that 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 use a name that is different from one of the default names, you must include the LIBRARY parameter in the CREATE WRAPPER statement.

Suppose that you have a federated server running on AIX, and you decide to use a wrapper name that is not one of the default names. Examples of the CREATE WRAPPER statements for `SQLNET` and `NET8` are:

```
CREATE WRAPPER mywrapper LIBRARY 'libdb2sqlnet.a'
```

```
CREATE WRAPPER mywrapper LIBRARY 'libdb2net8.a'
```

See the Related reference at the end of this topic for a list of Oracle wrapper library names.

The next task in this sequence of tasks is registering the server definitions for an Oracle data source.

Related tasks:

- “Registering the server definitions for an Oracle data source” on page 64

Related reference:

- “CREATE WRAPPER statement” in the *SQL Reference, Volume 2*
- “Oracle wrappers and library names” on page 63

Oracle wrappers and library names

This topic provides the Oracle wrapper names and the Oracle library names that you can use when you register a wrapper to access Oracle data sources.

The following table lists the Oracle wrapper names to use depending on the Oracle client version and the operating system that you are using.

Table 13. Oracle wrappers by client version and operating system

Oracle client	Operating system	Wrapper to use
Oracle Version 7	AIX	SQLNET
	Windows NT and Windows 2000	SQLNET
	HP-UX, Linux, and Solaris Operating Environment	not applicable
Oracle Version 8	AIX	NET8
	Windows NT or Windows 2000	NET8 (recommended) or SQLNET
	HP-UX, Linux, and Solaris	NET8
Oracle Version 9	AIX	NET8
	Windows NT or Windows 2000	NET8 (recommended) or SQLNET
	HP-UX, Linux, and Solaris	NET8

The following table lists the Oracle wrapper library names to use depending on the operating system of your federated server.

Table 14. Oracle wrapper library names

Operating system on your federated server	Wrapper library names for SQLNET	Wrapper library names for NET8
AIX	libdb2sqlnet.a	libdb2net8.a
HP-UX	libdb2sqlnet.sl	libdb2net8.sl
Linux	libdb2sqlnet.so	libdb2net8.so
Solaris Operating Environment	libdb2sqlnet.so	libdb2net8.so
Windows NT and Windows 2000	db2sqlnet.dll	db2net8.dll

Related tasks:

- "Registering the server definitions for an Oracle data source" on page 64

Related reference:

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*

Registering the server definitions for an Oracle data source

Registering the server definitions for an Oracle data source is part of the larger task of adding Oracle data sources to federated servers.

In the federated database, you must define each Oracle server that you want to access. You must first locate the node name of the Oracle data source, and then use this node name when you register the server.

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. 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.

Related tasks:

- "Creating the user mappings for an Oracle data source" on page 66

Related reference:

- "ALTER SERVER statement" in the *SQL Reference, Volume 2*
- "CREATE SERVER statement" in the *SQL Reference, Volume 2*
- Appendix B, "Server options for federated systems", on page 367
- "CREATE SERVER statement - Examples for Oracle wrapper" on page 65

CREATE SERVER statement - Examples for Oracle wrapper

This topic provides examples that show you how to use the CREATE SERVER statement to register servers for the Oracle wrapper. This topic includes a complete example, which shows how to register a server with required parameters, and an example with additional server options.

Complete example:

The following example shows you how to register a server definition for an Oracle wrapper by using the CREATE SERVER statement:

```
CREATE SERVER oraserver TYPE oracle VERSION 8.1.7 WRAPPER net8
      OPTIONS (NODE 'paris_node')
```

oraserver
: A name that you assign to the Oracle database server. This name must be unique. Duplicate server names are not allowed.

TYPE *oracle*
: Specifies the type of data source server to which you are configuring access. The type parameter for the SQLNET and NET8 wrappers must be *oracle*.

VERSION *8.1.7*
: The version of Oracle database server that you want to access. The supported Oracle versions are 7.3.4, 8.x, and 9.x.

WRAPPER *net8*
: The 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.

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

Server option example:

When you create the server definition, you can specify additional server options in the CREATE SERVER statement. These server options include general server options and Oracle-specific server options.

DB2 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 this server option:

```
CREATE SERVER oraserver TYPE oracle VERSION 8.1.7 WRAPPER net8
      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 specific columns with the ALTER NICKNAME statement.

Related tasks:

- "Registering the server definitions for an Oracle data source" on page 64

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*

Creating the user mappings for an Oracle data source

Creating the user mappings for an Oracle data source is part of the larger task of adding Oracle data sources to federated servers.

When you attempt to access an Oracle server, the federated server establishes a connection to the data source 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. 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 map a local user ID to the Oracle server user ID and password, issue a CREATE USER MAPPING statement.

For example:

```
CREATE USER MAPPING FOR userid SERVER oraserver
       OPTIONS (REMOTE_AUTHID 'remote_id', REMOTE_PASSWORD 'remote_password')
```

The next task in this sequence of tasks is testing the connection to the Oracle server.

Related tasks:

- "Testing the connection to the Oracle server" on page 68

Related reference:

- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*
- "CREATE USER MAPPING statement - Examples for Oracle wrapper" on page 67

CREATE USER MAPPING statement - Examples for Oracle wrapper

This topic provides examples that show you how to 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 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 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 defined at an Oracle server.

SERVER *oraserver*
Specifies the name of the Oracle server that you defined in the CREATE SERVER statement.

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.

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.

Special register example:

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')
```

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.

Restriction: The user ID at the Oracle data source must have been created using the Oracle **create user** command with the 'identified by' clause, instead of the 'identified externally' clause.

Related tasks:

- "Creating the user mappings for an Oracle data source" on page 66

Related reference:

- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*

Testing the connection to the Oracle server

Testing the connection to the Oracle server is part of the larger task of adding Oracle data sources to federated servers.

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:

1. Open a pass-through session to issue an SQL SELECT statement on the Oracle system tables.

   For example:

   ```
   SET PASSTHRU remote_server_name
   SELECT count(*) FROM sys.all_tables
   SET PASSTHRU RESET
   ```

   If the SQL SELECT statement returns a count, your server definition and your user mapping are set up properly.
2. If the SQL SELECT statement returns an error, you might need to:

- Check the Oracle server to make sure that it is configured for incoming connections.
- Check your user mapping to make sure that the settings for the REMOTE_AUTHID and REMOTE_PASSWORD options are valid for the connections to the Oracle server. Alter the user mapping, or create another user mapping as necessary.
- Check the Oracle client software on the DB2 federated server to make sure that it is installed and configured correctly to connect to the Oracle server.
- Check the settings of your DB2 federated variables to verify that they are correct for the Oracle server. These variables include the system environment variables, the db2dj.ini variables, and the DB2 Profile Registry (db2set) variable.
- Check your server definition. If necessary, drop it and create it again.

The next task in this sequence of tasks is registering nicknames for Oracle tables and views.

Related concepts:

- "Server definitions and server options" on page 10

Related tasks:

- "Checking the federated server setup" in the *DB2 Information Integrator Installation Guide*
- "Checking the data source environment variables" in the *DB2 Information Integrator Installation Guide*
- "Registering nicknames for Oracle tables and views" on page 69

Related reference:

- "ALTER USER MAPPING statement" in the *SQL Reference, Volume 2*

Registering nicknames for Oracle tables and views

Registering nicknames for Oracle tables and views is part of the larger task of adding Oracle data sources to federated servers.

For each Oracle server that you define, 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.

The federated database relies on catalog statistics for nicknamed objects to optimize query processing. These statistics are gathered when you register a nickname for a data source object by 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 that is useful to the optimizer is read from the data source catalogs and placed into the global catalog on the federated server. Because some or all of the data source catalog information might be used by the optimizer, update statistics (using the data source command that is equivalent to the DB2 **RUNSTATS** command) at the data source before you register a nickname.

Procedure:

To register a nickname, issue the CREATE NICKNAME statement.

For example:

```
CREATE NICKNAME oracle_name FOR oraserver."remote_schema"."remote.table"
```

Nicknames can be up to 128 characters in length.

Repeat this step for each Oracle table or view for which you want to create a nickname.

When you create the nickname, DB2 will use the connection to query the data source catalog. This query tests your connection to the data source by using the nickname. If the connection does not work, you will receive an error message.

Related reference:

- "RUNSTATS Command" in the *Command Reference*
- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*
- "CREATE NICKNAME statement - Examples for Oracle wrapper" on page 70

CREATE NICKNAME statement - Examples for Oracle wrapper

This topic provides an example that shows you how to use the CREATE NICKNAME statement to register a nickname for an Oracle table or view that you want to access.

This example shows how to specify a remote object for the Oracle server under which the nickname is assigned:

```
CREATE NICKNAME PARISINV FOR oraserver."france"."inventory"
```

PARISINV
A unique nickname used to identify the Oracle table or view.

Note: 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.

oraserver."france"."inventory"
A three-part identifier for the remote object:

- *oraserver* is the name that you assigned to the Oracle database server in the CREATE SERVER statement.
- *france* is the name of the remote schema 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 folds the names of the Oracle schemas and tables to uppercase unless you enclose the names in quotation marks.

Related tasks:

- "Registering nicknames for Oracle tables and views" on page 69

Related reference:

- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*

Tuning and troubleshooting the configuration to Oracle data sources

After you set up the configuration to Oracle data sources, you can modify the configuration to improve performance. For example, you might want to set the DB2_DJ_COMM profile registry variable to improve performance when the Oracle data source is accessed.

Improving performance by setting the DB2_DJ_COMM variable (UNIX)

If you find that it takes a long time to access the Oracle server, you can improve the performance by setting the DB2_DJ_COMM variable. When you set the DB2_DJ_COMM, the federated server loads the wrapper upon initialization rather than when you attempt to access the data source.

Procedure:

To set the DB2_DJ_COMM variable:

1. Set the DB2_DJ_COMM variable to the wrapper library that corresponds to the wrapper that you specified. Suppose that your federated server is running AIX and the wrapper you are using is NET8. The command to set the DB2_DJ_COMM variable is:

   ```
   db2set DB2_DJ_COMM= 'libdb2net8.a'
   ```

The following table lists the valid Oracle library names.

Table 15. Oracle wrapper library names

Operating system on your federated server	SQLNET wrapper library names	NET8 wrapper library names
AIX	libdb2sqlnet.a	libdb2net8.a
HP-UX	libdb2sqlnet.sl	libdb2net8.sl
Linux	libdb2sqlnet.so	libdb2net8.so
Solaris Operating Environment	libdb2sqlnet.so	libdb2net8.so

2. Issue the following commands to recycle the DB2 instance:

   ```
   db2stop
   db2start
   ```

   By recycling the DB2 instance, you ensure that the DB2 instance accepts the variable changes that you made.

Connectivity problems

For each HOST in the DESCRIPTION section of the tnsnames.ora file, you might need to update the 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. Otherwise, you need an entry for the remote host. See your network administrator to determine how your network is configured. If you need to update the hosts file, the file location depends on the federated server operating system:

On UNIX federated servers
Update the /etc/hosts file.

On Windows federated servers
Update the x:\winnt\system32\drivers\etc\hosts file.

Related tasks:

- “Adding Oracle data sources to federated servers” on page 59

Related reference:

- “db2set - DB2 Profile Registry Command” in the *Command Reference*

Chapter 7. Configuring access to Sybase data sources

This chapter explains how to configure your federated server to access data that is stored in Sybase data sources.

This chapter lists the tasks that you need to perform, shows examples of the SQL statements that you need to issue, and provides tuning and troubleshooting information that you can use when you set up the configuration to Sybase data sources.

Adding Sybase data sources to federated servers

Configuring the federated server to access Sybase data sources involves supplying the server with information about the Sybase data sources and objects that you want to access.

You can configure access to Sybase data sources through the DB2 Control Center, through the DB2 Command Center, or through the DB2 command line processor.

The advantage of using the DB2 Control Center is that you do not need to type each statement and command. The DB2 Control Center provides the easiest method to configure access quickly to Sybase data sources. However, you cannot use the DB2 Control Center to initiate the following configuration tasks:

- Setting up and testing the Sybase client configuration file
- Testing the connection to the Sybase server to validate the server definition and user mappings
- Adding or dropping column options

Prerequisites:

- Access to the DB2 Command Center or the DB2 command line processor.
- A federated server and database that are set up to access Sybase data sources.
- The Sybase client software that is installed and configured on the federated server.
- The proper setup of the system environment variables, db2dj.ini variables, and DB2 Profile Registry (db2set) variables. The variables are: SYBASE and SYBASE_OCS.

Restriction:

The Sybase Open Client DB-Library wrapper called DBLIB is a read-only wrapper and does not support INSERT, UPDATE, or DELETE operations.

Procedure:

To add a Sybase data source to a federated server:
1. Set up and test the Sybase client configuration file.
2. Register the wrapper.
3. Register the server definition.
4. Create the user mappings.
5. Test the connection to the Sybase server.
6. Register nicknames for Sybase tables and views.

Related concepts:
- "Fast track to configuring your data sources" on page 1

Related tasks:
- "Checking the data source environment variables" in the *DB2 Information Integrator Installation Guide*
- "Setting up and testing the Sybase client configuration file" on page 74

Setting up and testing the Sybase client configuration file

Setting up and testing the Sybase client configuration file is part of the larger task of adding Sybase data sources to federated servers.

The client configuration file is used to connect to Sybase using the Sybase Open Client libraries that are installed on the federated server. This 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 DB2 federated server that will be used to connect to Sybase. The steps that you must use to set up and test this file depend on the operating system that you are running on your federated server.

Procedure:

To set up and test the client configuration file:

On UNIX operating systems:

1. Set up the client configuration file by using the utility that comes with the Sybase Open Client software. This file is created in the $SYBASE/interfaces directory. See the Sybase documentation for more information about using this utility.
2. Make the interfaces file accessible to the DB2 federated server instance by using one of the following methods:
   - Copy this file to the $HOME/sqllib directory of the DB2 federated instance.
   - Use the **ln** command to create a link from the /sqllib subdirectory to the interfaces file in the instance $HOME/sqllib directory. For example:

     ```
     ln -s -f /home/sybase/interfaces  /home/db2djinst1/sqllib
     ```
   - Use the IFILE server option to specify the full path to the Sybase interfaces file.
3. 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**.

On Windows operating systems:

1. Set up the client configuration file by using the utility that comes with the Sybase Open Client software. This file is created in the %SYBASE%\ini\sql.ini directory. See the Sybase documentation for more information about using this utility.
2. Make this sql.ini file accessible to the DB2 federated server instance by using copying this file to the c:\Program Files\IBM\SQLLIB directory of the DB2 federated instance.

   Because DB2 Information Integrator uses interfaces as the default name for the Sybase client configuration file, rename the Windows sql.ini file in the c:\Program Files\IBM\SQLLIB directory to interfaces.

   Required: If you do not rename the sql.ini file to interfaces, you must use the IFILE server option when you create the server definition.
3. 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**.

The next task in this sequence of tasks is registering the Sybase wrapper.

Related tasks:

- "Registering the Sybase wrapper" on page 76

Registering the Sybase wrapper

Registering the Sybase wrapper is part of the larger task of adding Sybase data sources to federated servers.

To specify the wrapper that will be used to access Sybase data sources, use the CREATE WRAPPER statement. Two wrappers for Sybase, the Open Client Client-Library wrapper called CTLIB and the Open Client DB-Library wrapper called DBLIB, are included with DB2 Information Integrator.

Procedure:

To specify the wrapper that you want to use to access Sybase data sources, issue the CREATE WRAPPER statement.

For example:

```
CREATE WRAPPER CTLIB
```

You can use either the CTLIB or DBLIB wrapper regardless of the operating system that is running on your federated server.

Recommendation: Use one of the default wrapper names (CTLIB or DBLIB). When you register the wrapper by using one of the default names, the federated server automatically takes the default library name that is associated with that 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 use a name that is different from one of the default names, you must include the LIBRARY parameter in the CREATE WRAPPER statement. Suppose that you have a federated server running on AIX and you decide to use a wrapper name that is not one of the default names. Examples of the CREATE WRAPPER statements for CTLIB and DBLIB are:

```
CREATE WRAPPER mywrapper LIBRARY 'libdb2ctlib.a'
```

```
CREATE WRAPPER mywrapper LIBRARY 'libdb2dblib.a'
```

The wrapper library names for Sybase are:

Table 16. Sybase wrapper library names

Operating system on your federated server	CTLIB wrapper library names	DBLIB wrapper library names
AIX	libdb2ctlib.a	libdb2dblib.a
HP-UX	libdb2ctlib.sl	libdb2dblib.sl
Linux	libdb2ctlib.so	libdb2dblib.so

Table 16. Sybase wrapper library names (continued)

Operating system on your federated server	CTLIB wrapper library names	DBLIB wrapper library names
Solaris Operating Environment	libdb2ctlib.so	libdb2dblib.so
Windows NT and Windows 2000	db2ctlib.dll	db2dblib.dll

The next task in this sequence of tasks is registering the server definitions for a Sybase data source.

Related tasks:

- "Registering the server definitions for a Sybase data source" on page 77

Related reference:

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*

Registering the server definitions for a Sybase data source

Registering the server definitions for a Sybase data source is part of the larger task of adding Sybase data sources to federated servers.

In the federated database, you must define each Sybase server that you want to access. You must first locate the node name of the Sybase data source, and then use this node name when you register the server.

Procedure:

To register a server definition for a Sybase data source:

1. Locate the node name in the Sybase interfaces file.

   Example interfaces file on UNIX operating systems:

   ```
   sybase119
   query tcp ether anaconda 4100
   ```

   Example interfaces file on Windows NT or Windows 2000 operating systems:

   ```
   [sybase119]
   query=TCP,anaconda,4100
   ```

   In these examples, the node name is sybase119. The node name is followed by the type of connection (TCP/IP) and the host name (anaconda).

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

2. Issue the CREATE SERVER statement.

   For example:

   ```
   CREATE SERVER server_name TYPE SYBASE VERSION 12.0 WRAPPER CTLIB
         OPTIONS (NODE 'sybnode', DBNAME 'sybdb')
   ```

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

Important: If you did not rename the sql.ini file to interfaces when you set up the Sybase client configuration file, you must use the IFILE server option when you register the server definition.

The next task in this sequence of tasks is creating a user mapping for a Sybase data source.

Related tasks:

- "Creating a user mapping for a Sybase data source" on page 80

Related reference:

- "ALTER SERVER statement" in the *SQL Reference, Volume 2*
- "CREATE SERVER statement" in the *SQL Reference, Volume 2*
- Appendix B, "Server options for federated systems", on page 367
- "CREATE SERVER statement - Examples for Sybase wrapper" on page 78

CREATE SERVER statement - Examples for Sybase wrapper

This topic provides examples that show you how to use the CREATE SERVER statement to register servers for the Sybase wrapper. This topic includes a complete example, which shows how to register a server with required parameters, and an example with additional server options.

Complete example:

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. This name must be unique. Duplicate server names are not allowed.

TYPE *SYBASE*
: Specifies Sybase as the type of data source to which you are configuring access. The TYPE parameter for the CTLIB and DBLIB wrappers must be *SYBASE*.

VERSION *12.0*
: The version of the Sybase database server software that you want to access. The supported versions are 11, 11.5, 11.9, 12, and 12.5.

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

NODE *'sybnode'*
: The name of the node where *SYBSERVER* 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. Obtain this name from the Sybase server. This value is case sensitive.

  Although the database name is specified as an option in the CREATE SERVER statement, it is 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 use the IFILE server option when you register the server definition.

Server option examples:

When you register the server, you can specify additional server options in the CREATE SERVER statement. These server options include general server options and Sybase-specific server options.

The following example shows how to use the TIMEOUT server option when you register a server with the CTLIB wrapper on a UNIX operating system:

```
CREATE SERVER SYBSERVER TYPE SYBASE
       VERSION 12.0 WRAPPER CTLIB
        OPTIONS (NODE 'sybnode', DBNAME 'sybdb',
         TIMEOUT '60', LOGIN_TIMEOUT '60', PACKET_SIZE '1024',
         IFILE '/home/sybase/interfaces')
```

The timeout value is 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 use the IFILE server option when you register a server on a Windows operating system:

```
CREATE SERVER SYBSERVER TYPE SYBASE
       VERSION 12.0 WRAPPER CTLIB
        OPTIONS (NODE 'sybnode', DBNAME 'sybdb',
         IFILE 'C:\Sybase\ini\sql.ini')
```

The additional Sybase-specific server options are:

IFILE
Specifies the full path and name of the Sybase Open Client interfaces file.

Use this server option if you did not copy or link the sql.ini file as $SQLLIB\interfaces (on UNIX systems) or as %SQLLIB%/interfaces (on Windows operating systems).

IGNORE_UDT
Specifies whether the federated server determines the built-in type that underlies a UDT without strong typing.

LOGIN_TIMEOUT
Specifies the length of time, in seconds, that DB2 Universal Database waits for a login response when making a connection attempt. The default behavior is to wait indefinitely for a response from the Sybase server.

PACKET_SIZE
Determines the packet size that Client-Library uses when sending Tabular Data Stream (TDS) packets. If an application needs to send or receive large amounts of text, image, or bulk data, a larger packet size might improve efficiency.

TIMEOUT
Specifies the length of time, in seconds, that DB2 Universal Database waits for a server response to a command. The default behavior is to wait indefinitely for a response from the Sybase server. The Sybase Open Client uses timeout thresholds to interrupt queries and responses that run for a long a period of time.

Related tasks:

- "Registering the server definitions for a Sybase data source" on page 77

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*

Creating a user mapping for a Sybase data source

Creating a user mapping for a Sybase data source is part of the larger task of adding Sybase data sources to federated servers.

When you attempt to access a Sybase server, the federated server establishes a connection to the data source 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. Create a user mapping for each user ID that will access the federated system to send distributed requests to the Sybase data source.

Procedure:

To map a local user ID to the Sybase server user ID and password, issue a CREATE USER MAPPING statement.

For example:

```
CREATE USER MAPPING FOR userid SERVER SYBSERVER
      OPTIONS (REMOTE_AUTHID 'remote_id', REMOTE_PASSWORD 'remote_password')
```

The next task in this sequence of tasks is testing the connection to the Sybase server.

Related tasks:

- “Testing the connection to the Sybase server” on page 82

Related reference:

- “CREATE USER MAPPING statement” in the *SQL Reference, Volume 2*
- “CREATE USER MAPPING statement - Examples for Sybase wrapper” on page 81

CREATE USER MAPPING statement - Examples for Sybase wrapper

This topic provides examples that show you how to use the CREATE USER MAPPING statement to map a federated server user ID to a Sybase server user ID and password. This topic includes a complete example with 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 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 name of the Sybase server that you defined in the CREATE SERVER statement.

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.

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.

Special register example:

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')
```

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.

Related tasks:

- “Creating a user mapping for a Sybase data source” on page 80

Related reference:

- “CREATE USER MAPPING statement” in the *SQL Reference, Volume 2*

Testing the connection to the Sybase server

Testing the connection to the Sybase server is part of the larger task of adding Sybase data sources to federated servers.

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:

1. Open a pass-through session to issue an SQL SELECT statement on the Sybase system tables.

   For example:

   ```
   SET PASSTHRU local_server_name
   SELECT count(*) FROM dbo.sysobjects
   SET PASSTHRU RESET
   ```

   Where *local_server_name* is the name you used to register the remote server in the federated database catalog. If the SQL SELECT statement returns a count, your server definition and your user mapping are set up properly.
2. If the SQL SELECT statement returns an error, you might need to:
   - Check the Sybase server to make sure that it is configured for incoming connections.
   - Check your user mapping to make sure that the settings for the REMOTE_AUTHID and REMOTE_PASSWORD options are valid for the connections to the Sybase server. Alter the user mapping, or create another user mapping as necessary.
   - Check the Sybase client software on the DB2 federated server to make sure that it is installed and configured correctly to connect to the Sybase server.
   - Check the settings of your DB2 federated variables to verify that they are correct for the Sybase server. These variables include the system environment variables, the db2dj.ini variables, and the DB2 Profile Registry (db2set) variable.
   - Check your server definition. If necessary, drop it and create it again.

The next task in this sequence of tasks is registering nicknames for Sybase tables and views.

Related concepts:

- "Server definitions and server options" on page 10

Related tasks:

- "Checking the federated server setup" in the *DB2 Information Integrator Installation Guide*
- "Checking the data source environment variables" in the *DB2 Information Integrator Installation Guide*
- "Registering nicknames for Sybase tables and views" on page 84

Related reference:

- "ALTER USER MAPPING statement" in the *SQL Reference, Volume 2*

Registering nicknames for Sybase tables and views

Registering nicknames for Sybase tables and views is part of the larger task of adding Sybase data sources to federated servers.

For each Sybase server that you define, 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.

The federated database relies on catalog statistics for nicknamed objects to optimize query processing. These statistics are gathered when you register a nickname for a data source object by 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 that is useful to the optimizer is read from the data source catalogs and placed into the global catalog on the federated server. Because some or all of the data source catalog information might be used by the optimizer, update statistics (using the data source command that is equivalent to the DB2 **RUNSTATS** command) at the data source before you register a nickname.

Procedure:

To register a nickname, issue the CREATE NICKNAME statement.

For example:

```
CREATE NICKNAME sybase_name FOR SYBSERVER."remote_schema"."remote.table"
```

Nicknames can be up to 128 characters in length.

Repeat this step for each Sybase table or view for which you want to create a nickname.

When you create the nickname, DB2 will use the connection to query the data source catalog. This query tests your connection to the data source by using the nickname. If the connection does not work, you will receive an error message.

Related reference:

- "RUNSTATS Command" in the *Command Reference*
- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*
- "CREATE NICKNAME statement - Examples for Sybase wrapper" on page 85

CREATE NICKNAME statement - Examples for Sybase wrapper

This topic provides an example that shows you how to use the CREATE NICKNAME statement to register a nickname for a Sybase table or view that you want to access.

This example shows how to specify a remote object for the Sybase server under which the nickname is assigned:

```
CREATE NICKNAME SYBSALES FOR SYBSERVER."salesdata"."europe"
```

SYBSALES
Is a unique nickname for the Sybase table or view.

The nickname is a two-part name—the schema and the nickname. If you omit the schema when creating the nickname, the schema of the nickname will be the authentication ID of the user creating the nickname.

SYBSERVER."salesdata"."europe"
Is a three-part identifier for the remote object.

- *SYBSERVER* is the name you assigned to the Sybase database server in the CREATE SERVER statement.
- *salesdata* is the name of the remote schema 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.

Related tasks:

- "Registering nicknames for Sybase tables and views" on page 84

Related reference:

- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*

Tuning and troubleshooting the configuration to Sybase data sources

After you set up the configuration to Sybase data sources, you might want to modify the configuration to improve performance. For example, you might want to set the DB2_DJ_COMM environment variable to improve performance when the Sybase data source is accessed.

Improving performance by setting the DB2_DJ_COMM environment variable (UNIX)

If you find that it takes a long time to access the Sybase server, you can improve the performance by setting the DB2_DJ_COMM environment variable. When you set the DB2_DJ_COMM environment variable, the federated server loads the wrapper upon initialization rather than when you attempt to access the data source.

Procedure:

To set the DB2_DJ_COMM environment variable:

1. Set the DB2_DJ_COMM environment variable to the wrapper library that corresponds to the wrapper that you specified. Suppose that your federated server is running AIX and the wrapper you are using is CTLIB. The command to set the DB2_DJ_COMM environment variable is:

   ```
   db2set DB2_DJ_COMM= 'libdb2ctlib.a'
   ```

   Consult the following table for the proper library name.

Table 17. Sybase wrapper library names

Operating system on your federated server	CTLIB wrapper library names	DBLIB wrapper library names
AIX	libdb2ctlib.a	libdb2dblib.a
HP-UX	libdb2ctlib.sl	libdb2dblib.sl
Linux	libdb2ctlib.so	libdb2dblib.so
Solaris Operating Environment	libdb2ctlib.so	libdb2dblib.so

2. Issue the following commands to recycle the DB2 instance:

   ```
   db2stop
   db2start
   ```

   By recycling the DB2 instance, you ensure that the DB2 instance accepts the variable changes that you made.

Using CTLIB instead of DBLIB

CT-Library supports dynamic prepare-and-execute of statements. This allows CT-Library applications to prepare a statement one time and execute it many times with different inputs. Preparing a statement one time eliminates the need to recompile the statement for each input parameter change. Although the DB2 application might not take advantage of dynamic SQL, federated query processing of remote queries uses dynamic SQL exclusively.

Resolving the sp_helpindex error

The federated system relies on one of the Sybase catalog stored procedures, `sp_helpindex`. If you receive the following SQL error, the Sybase catalog stored procedures might not be installed on the Sybase server.

```
SQL0204N "sp_helpindex" is an undefined name.
```

Have the Sybase administrator install the catalog stored procedures on the Sybase server.

Related tasks:

- "Adding Sybase data sources to federated servers" on page 73

Related reference:

- "db2set - DB2 Profile Registry Command" in the *Command Reference*

Chapter 8. Configuring access to Microsoft SQL Server data sources

This chapter explains how to configure your federated server to access data that is stored in Microsoft SQL Server databases.

This chapter lists the tasks that you need to perform, shows examples of the SQL statements that you need to issue, and provides tuning and troubleshooting information that you can use when you set up the configuration to Microsoft SQL Server data sources.

Adding Microsoft SQL Server data sources to federated servers

Configuring the federated server to access Microsoft SQL Server data sources involves supplying the federated server with information about the Microsoft SQL Server data sources and objects that you want to access.

You can configure access to Microsoft SQL Server data sources through the DB2 Control Center, through the DB2 Command Center, or through the DB2 command line processor.

The advantage of using the DB2 Control Center is that you do not need to type each statement and command. The DB2 Control Center provides the easiest method to configure access quickly to Microsoft SQL Server data sources. However, you cannot use the DB2 Control Center to initiate the following configuration tasks:

- Testing the connection to the Microsoft SQL Server server to validate the server definition and user mappings
- Adding or dropping column options

Prerequisites:

- Access to the DB2 Command Center or the DB2 command line processor.
- A federated server and database that are set up to access Microsoft SQL Server data sources.
- The Microsoft SQL Server ODBC driver that is installed and configured on the federated server.
- The proper setup of the system environment variables, db2dj.ini variables, and DB2 Profile Registry (db2set) variables. The variables are: DJXODBCTRACE, DJX_ODBC_LIBRARY_PATH, ODBCINI, DB2LIBPATH, and DB2ENVLIST.

Procedure:

To add a Microsoft SQL Server data source to a federated server:

1. Prepare the federated server and federated database.
   - On Windows, confirm that the ODBC System DSN is properly set up, and test the connection to the Microsoft SQL Server remote server.
   - On UNIX systems, update or create an odbc.ini file, and test the connection to the Microsoft SQL Server remote server.
2. Register the wrapper.
3. Register the server definition.
4. Create the user mappings.
5. Test the connection to the Microsoft SQL Server remote server.
6. Register nicknames for Microsoft SQL Server tables and views.

Related concepts:

- "Fast track to configuring your data sources" on page 1

Related tasks:

- "Checking the data source environment variables" in the *DB2 Information Integrator Installation Guide*
- "Preparing the federated server and database to access Microsoft SQL Server data sources" on page 90

Preparing the federated server and database to access Microsoft SQL Server data sources

Preparing the federated server and database to access Microsoft SQL Server data sources is part of the larger task of adding Microsoft SQL Server data sources to federated servers.

The steps that you need to follow to prepare the federated server and database to access Microsoft SQL Server data sources depend on the operating system that is running on your federated server.

Procedure:

To prepare the federated server and database:

On Windows:

1. Verify that the ODBC System DSN is set to connect to the Microsoft SQL Server data source by checking this setting in the Control Panel.
   a. From the **Start** menu, open the Control Panel.

   b. Double-click **ODBC Data Sources** to display the ODBC Data Source Administrator window.
   c. Click the System DSN tab, and locate an entry for the Microsoft SQL Server remote server.

      The entry is the value that you will use for the NODE server option when you register the server in the federated database.
2. From the ODBC Data Source Administrator window, select **Configure** to test the connection from the ODBC Systems DSN to the Microsoft SQL Server data source. Alternatively, you can test the connection by using the Microsoft SQL Server query tool.

On UNIX systems:
1. Verify that the odbc.ini file is updated (or if necessary created) on the federated server.

   Recommendation: Place the odbc.ini file or a copy of this file in the home directory of the DB2 instance owner.
2. Verify that the path to the odbc.ini is in the ODBCINI environment variable.
3. Verify that the appropriate symbolic link is created:
   - On HP-UX, the symbolic link is from `/usr/exe/libodbcinst.sl` to `$DJX_ODBC_LIBRARY_PATH/libodbcinst.sl`.
   - On Linux, the symbolic link is from `/usr/local/locale` to `$DJX_ODBC_LIBRARY_PATH/../locale`.
   - On Solaris Operating Environments, the symbolic link is from `$HOME/sqllib/locale` to `$DJX_ODBC_LIBRARY_PATH/../locale`. `$HOME` is the home directory of the DB2 instance owner.
4. Test the connection from the federated server to the Microsoft SQL server data source by using the DataDirect Connect ODBC **demoodbc** tool.
   a. From an operating system command prompt, issue the following command:
      ```
      export ODBCINI=$HOME/.odbc.ini
      ```
   b. Run the **/opt/odbc/odbc.sh** script. This script sets up several operating specific environment variables.
   c. Test the connection to the Microsoft SQL server data source by using the DataDirect Connect ODBC **demoodbc** tool. The **demoodbc** tool is located in the /demo subdirectory of the Connect ODBC libraries.

The next task in this sequence of tasks is registering the Microsoft SQL Server wrapper.

Related tasks:
- "Registering the Microsoft SQL Server wrapper" on page 92

Registering the Microsoft SQL Server wrapper

Registering the Microsoft SQL Server wrapper is part of the larger task of adding Microsoft SQL Server data sources to federated servers.

To specify the wrapper that you will use to access Microsoft SQL Server data sources, issue the CREATE WRAPPER statement. DB2 Information Integrator includes two wrappers for Microsoft SQL Server. The wrapper that you use depends on the operating system of your federated server.

- On UNIX systems, the default wrapper name is MSSQLODBC3 for the DataDirect Connect ODBC 3.7 (or later) driver.
- On Windows, the default wrapper name is DJXMSSQL3 for the ODBC 3.0 (or later) driver.

Procedure:

To specify the wrapper that you want to use to access Microsoft SQL Server data sources, issue the CREATE WRAPPER statement.

For example on Windows NT and Windows 2000:

```
CREATE WRAPPER DJXMSSQL3
```

Recommendation: Use one of the default wrapper names (DJXMSSQL3 or MSSQLODBC3). When you register the wrapper by using one of the default names, the federated server automatically takes the default library name that is associated with that 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 use a name that is different from one of the default names, you must include the LIBRARY parameter in the CREATE WRAPPER statement. Suppose that you have a federated server running on AIX, and you decide to use a wrapper name that is not one of the default names. The CREATE WRAPPER statement that you need to issue is:

```
CREATE WRAPPER mywrapper LIBRARY 'libdb2mssql3.a'
```

The value *mywrapper* is the name that you give to the wrapper instead of using the default wrapper name.

The wrapper library names for Microsoft SQL Server are:

Table 18. Microsoft SQL Server wrapper library names

Operating system on your federated server	Wrapper library name
AIX	libdb2mssql3.a

Table 18. Microsoft SQL Server wrapper library names (continued)

Operating system on your federated server	Wrapper library name
HP-UX	libdb2mssql3.sl
Linux	libdb2mssql3.so
Solaris Operating Environment	libdb2mssql3.so
Windows	db2mssql3.dll

The next task in this sequence of tasks is registering the server definitions for a Microsoft SQL Server data source.

Related tasks:

- "Registering the server definitions for a Microsoft SQL Server data source" on page 93

Related reference:

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*

Registering the server definitions for a Microsoft SQL Server data source

Registering the server definitions for a Microsoft SQL Server data source is part of the larger task of adding Microsoft SQL Server data sources to federated servers.

In the federated database, you must define each Microsoft SQL Server remote server that you want to access. You must first locate the node name of the Microsoft SQL Server remote server, and then use this node name when you register the server definition by issuing the CREATE SERVER statement.

Procedure:

To register a server definition for a Microsoft SQL Server data source:

1. Locate the node name.
   - If your federated server is using Windows NT or Windows 2000, the NODE is the System DSN name that you specified for the Microsoft SQL Server remote server that you are accessing.
   - If your federated server is using AIX, HP-UX, Linux, or Solaris Operating Environment, the NODE is defined in the .odbc.ini file.

     The following is an example of a .odbc.ini file on AIX.

     Example .odbc.ini file on AIX:

```
rawilson=MS SQL Server 7.0
medusa=MS SQL Server 7.0
[rawilson]
Driver=/opt/odbc/lib/ivmsss16.so
Description=MS SQL Server Driver for AIX
  Address=9.112.30.39,1433
[medusa]
Driver=/opt/odbc/lib/ivmsss16.so
Description=MS SQL Server Driver for AIX
Address=9.112.98.123,1433
```

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 [node_name] that describes each node.

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

2. Issue the CREATE SERVER statement.

   For example:

```
CREATE SERVER server_name TYPE MSSQLSERVER VERSION 7.0 WRAPPER djxmssql3
      OPTIONS (NODE 'sqlnode', DBNAME 'mssdb')
```

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 a user mapping for a Microsoft SQL Server data source.

Related tasks:

- “Creating a user mapping for a Microsoft SQL Server data source” on page 96

Related reference:

- “ALTER SERVER statement” in the *SQL Reference, Volume 2*
- “CREATE SERVER statement” in the *SQL Reference, Volume 2*
- Appendix B, “Server options for federated systems”, on page 367
- “CREATE SERVER statement - Examples for Microsoft SQL Server wrapper” on page 94

CREATE SERVER statement - Examples for Microsoft SQL Server wrapper

This topic provides examples that show you how to use the CREATE SERVER statement to register servers for the Microsoft SQL Server wrapper. This topic includes a complete example, which shows how to register a server with required parameters, and an example with additional server options.

Complete example:

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

```
CREATE SERVER sqlserver TYPE MSSQLSERVER VERSION 7.0 WRAPPER djxmssql3
      OPTIONS (NODE 'sqlnode', DBNAME 'africa')
```

sqlserver
: A name that you assign to the Microsoft SQL Server remote server. This name must be unique. Duplicate server names are not allowed.

TYPE *MSSQLSERVER*
: The type of data source to which you are configuring access. The TYPE parameter for the Microsoft SQL Server wrappers must be *MSSQLSERVER*.

VERSION *7.0*
: The version of Microsoft SQL Server database server software that you want to access. Supported versions are 6.5, 7.0, and 2000.

WRAPPER *djxmssql3*
: The wrapper name that you specified in the CREATE WRAPPER statement.

NODE *'sqlnode'*
: The name of the node where the Microsoft SQL Server remote server resides. 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 Microsoft SQL Server data sources.

DBNAME *'africa'*
: The name of the 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 Microsoft SQL Server data sources.

Server option examples:

When you register the server, you can specify additional server options in the CREATE SERVER statement. These server options include general server options and Microsoft SQL Server-specific server options.

The following example shows how to use the COLLATING_SEQUENCE server option:

```
CREATE SERVER sqlserver TYPE MSSQLSERVER VERSION 7.0 WRAPPER djxmssql3
      OPTIONS (NODE 'sqlnode', DBNAME 'africa', COLLATING_SEQUENCE 'I')
```

The COLLATING_SEQUENCE server option specifies whether the data source uses the same collating sequence as the federated server. On a Microsoft SQL Server database server that is running Windows NT or Windows 2000, the default collating sequence is case insensitive (for example, ’STEWART’ and ’StewART’ are considered equal). To guarantee correct results from the federated server, set the COLLATING_SEQUENCE server option to ’I’. This setting indicates that the Microsoft SQL Server data source is case insensitive.

Note: The federated server does not push down queries if the results that are returned from the data sources will be different from the results that are returned when processing the query at the federated server. When you set the COLLATING_SEQUENCE server option to ’I’, the federated server does not push down queries with string data or expressions and that include the following clauses, predicates, or functions:

- GROUP BY clauses
- DISTINCT clauses
- Basic predicates, such as equal to (=)
- Aggregate functions, such as MIN or MAX

Related tasks:

- “Registering the server definitions for a Microsoft SQL Server data source” on page 93

Related reference:

- “CREATE SERVER statement” in the *SQL Reference, Volume 2*

Creating a user mapping for a Microsoft SQL Server data source

Creating a user mapping for a Microsoft SQL Server data source is part of the larger task of adding Microsoft SQL Server data sources to federated servers.

When you attempt to access a Microsoft SQL Server data source, the federated server establishes a connection to the data source 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. Create a user mapping for each user ID that will access the federated system to send distributed requests.

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 userid SERVER sqlserver
      OPTIONS (REMOTE_AUTHID 'remote_id', REMOTE_PASSWORD 'remote_password')
```

The next task in this sequence of tasks is testing the connection to the Microsoft SQL Server remote server.

Related tasks:

- "Testing the connection to the Microsoft SQL Server remote server" on page 98

Related reference:

- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*
- "CREATE USER MAPPING statement - Examples for Microsoft SQL Server wrapper" on page 97

CREATE USER MAPPING statement - Examples for Microsoft SQL Server wrapper

This topic provides examples that show you how to 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 includes a complete example with 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 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 name of the Microsoft SQL Server remote server that you defined in the CREATE SERVER statement.

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.

Special register example:

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')
```

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.

Related tasks:

- "Creating a user mapping for a Microsoft SQL Server data source" on page 96

Related reference:

- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*

Testing the connection to the Microsoft SQL Server remote server

Testing the connection to the Microsoft SQL Server remote server is part of the larger task of adding Microsoft SQL Server data sources to federated servers.

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:

1. Open a pass-through session to issue an SQL SELECT statement on the Microsoft SQL Server system tables.

   For example:

   ```
   SET PASSTHRU remote_server_name
   SELECT count(*) FROM dbo.sysobjects
   SET PASSTHRU RESET
   ```

If the SQL SELECT statement returns a count, your server definition and your user mapping are set up properly.

2. If the SQL SELECT statement returns an error, you might need to:
   - Check the Microsoft SQL Server remote server to make sure that it is started.
   - Check the Microsoft SQL Server remote server to make sure that it is configured for incoming connections.
   - Check your user mapping to make sure that the settings for the REMOTE_AUTHID and REMOTE_PASSWORD options are valid for the connections to the Microsoft SQL Server remote server. Alter the user mapping, or create another user mapping as necessary.
   - Check the ODBC drivers on the DB2 federated server to make sure that they are installed and configured correctly to connect to the Microsoft SQL Server remote server.
   - Check the settings of your DB2 federated variables to verify that they are correct for the Microsoft SQL Server remote server. These variables include the system environment variables, the db2dj.ini variables, and the DB2 Profile Registry (db2set) variable.
   - Check your server definition. If necessary, drop it and create it again.

The next task in this sequence of tasks is registering nicknames for Microsoft SQL Server tables and views.

Related concepts:

- "Server definitions and server options" on page 10

Related tasks:

- "Checking the federated server setup" in the *DB2 Information Integrator Installation Guide*
- "Checking the data source environment variables" in the *DB2 Information Integrator Installation Guide*
- "Registering nicknames for Microsoft SQL Server tables and views" on page 99

Related reference:

- "ALTER USER MAPPING statement" in the *SQL Reference, Volume 2*

Registering nicknames for Microsoft SQL Server tables and views

Registering nicknames for Microsoft SQL Server tables and views is part of the larger task of adding Microsoft SQL Server data sources to federated servers.

For each Microsoft SQL Server remote server that you define, 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.

The federated database relies on catalog statistics for nicknamed objects to optimize query processing. These statistics are gathered when you register a nickname for a data source object by 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 that is useful to the optimizer is read from the data source catalogs and placed into the global catalog on the federated server. Because some or all of the data source catalog information might be used by the optimizer, update statistics (using the data source command that is equivalent to the DB2 **RUNSTATS** command) at the data source before you register a nickname.

Procedure:

To register a nickname, issue the CREATE NICKNAME statement.

For example:

```
CREATE NICKNAME mss_name FOR sqlserver."remote_schema"."remote.table"
```

Nicknames can be up to 128 characters in length.

Repeat this step for each Microsoft SQL Server table or view for which you want to create a nickname.

When you create the nickname, DB2 uses the connection to query the data source catalog tables (Microsoft SQL Server refers to these tables as system tables). This query tests your connection to the data source by using the nickname. If the connection does not work, you will receive an error message.

Related reference:

- “RUNSTATS Command” in the *Command Reference*
- “CREATE NICKNAME statement” in the *SQL Reference, Volume 2*
- “CREATE NICKNAME statement - Examples for Microsoft SQL Server wrapper” on page 101

CREATE NICKNAME statement - Examples for Microsoft SQL Server wrapper

This topic provides an example that shows you how to use the CREATE NICKNAME statement to register a nickname for a Microsoft SQL Server table or view that you want to access.

This example shows how to specify a remote object for the Microsoft SQL Server remote server under which the nickname is assigned:

```
CREATE NICKNAME cust_africa FOR sqlserver.customers.egypt
```

cust_africa
A unique nickname for the Microsoft SQL Server table or view.

Note: The nickname is a two-part name which includes the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authentication ID of the user creating the nickname.

sqlserver.customers.egypt
A three-part identifier for the remote object.

- *sqlserver* is the name that you assigned to the Microsoft SQL Server database server in the CREATE SERVER statement.
- *customers* is the name of the remote schema 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.

Related tasks:

- “Registering nicknames for Microsoft SQL Server tables and views” on page 99

Related reference:

- “CREATE NICKNAME statement” in the *SQL Reference, Volume 2*

Tuning and troubleshooting the configuration to Microsoft SQL Server data sources

After you set up the configuration to Microsoft SQL Server data sources, you might want to modify the configuration to improve performance. For example, you might want to set the DB2_DJ_COMM profile registry variable to improve performance when the federated server accesses the Microsoft SQL Server data source.

Improving performance by setting the DB2_DJ_COMM variable (UNIX)

If you find that it takes a long time to access the Microsoft SQL Server remote server, you can improve the performance by setting the DB2_DJ_COMM DB2 profile registry variable. When you set the DB2_DJ_COMM variable, the federated server loads the wrapper upon initialization rather than when you attempt to access the data source.

Procedure:

To set the DB2_DJ_COMM variable:

1. Set the DB2_DJ_COMM variable to the wrapper library that corresponds to the wrapper that you specified. Suppose that your federated server is running AIX and the wrapper that you are using is MSSQLODBC3. The command to set the DB2_DJ_COMM variable is:

   ```
   db2set DB2_DJ_COMM='libdb2mssql3.a'
   ```

   The following table lists the valid library names by supported operating system.

Table 19. Microsoft SQL Server wrapper library names

Operating system on your federated server	MSSQLODBC3 wrapper library names	DJXMSSQL3 wrapper library names
AIX	libdb2mssql3.a	none
HP-UX	libdb2mssql3.sl	none
Linux	libdb2mssql3.so	none
Solaris Operating Environment	libdb2mssql3.so	none
Windows NT and Windows 2000	none	db2mssql3.dll

2. Issue the following commands to recycle the DB2 instance:

   ```
   db2stop
   db2start
   ```

   By recycling the DB2 instance, you ensure that the DB2 instance accepts the variable changes that you made.

Obtaining ODBC traces

If you are experiencing problems when accessing the data source, you can obtain ODBC tracing information to analyze and resolve these problems. Activating a trace impacts system performance. Therefore, you should turn off tracing after you resolved the problems.

On Windows federated servers, use the trace tool that is provided by the ODBC Data Source Administrator to ensure that the ODBC tracing works properly.

On UNIX federated servers, set the DJXODBCTRACE variable in the db2dj.ini file. For example:

```
DJXODBCTRACE=/home/user1/trace_dir/filename.xxx
```

You also need to set tracing on for the .odbc.ini file. For example, suppose you are using the DataDirect ODBC 3.x driver. Find the example of the .odbc.ini file in the client directory. This file contains a sample of what is needed for trace files:

```
[ODBC]
Trace=0
TraceFile=/home/user1/trace_dir/filename.xxx
TraceDll==/opt/odbc/lib/odbctrac.so
InstallDir=/opt/odbc
```

The first line is set to Trace=0 when tracing is OFF, and this first line is set to Trace=1 when tracing is ON. The `TraceFile` should point to a path and file name that the instance has write access to. This path and file name should also match the line that is placed in the db2dj.ini file, `DJXODBCTRACE=/home/user1/trace_dir/filename.xxx`

Related tasks:

- “Adding Microsoft SQL Server data sources to federated servers” on page 89

Related reference:

- “db2set - DB2 Profile Registry Command” in the *Command Reference*

Chapter 9. Configuring access to ODBC data sources

This chapter explains how to configure your federated server to access data that is stored in ODBC data sources.

This chapter lists the tasks that you need to perform, shows examples of the SQL statements that you need to issue, and provides tuning and troubleshooting information that you can use when you set up the configuration to ODBC data sources.

Adding ODBC data sources to federated servers

DB2 Information Integrator provides wrappers that support specific data sources that are accessed through the ODBC API. Examples of these data sources include Oracle, Microsoft SQL Server, and Microsoft Excel. You will experience better performance if you use the wrappers specifically designed for those data sources. Data sources that are accessed through the ODBC API are referred to in this text as ODBC 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 DB2 Information Integrator.

The ODBC wrapper supports ODBC Version 3.x.

Configuring the federated server to access ODBC data sources involves supplying the federated server with information about the ODBC data sources and objects that you want to access.

You can configure access to ODBC data sources through the DB2 Control Center, through the DB2 Command Center, or through the DB2 command line processor.

The advantage of using the DB2 Control Center is that you do not need to type each statement and command. The DB2 Control Center provides the easiest method to configure access quickly to ODBC data sources.

You can use the ODBC wrapper on federated servers that use the following operating systems:

- AIX
- HP-UX

- Linux on Intel operating systems
- Solaris Operating Environment
- Windows NT, Windows 2000, Windows .NET

Prerequisites:

- Access to the DB2 Command Center or the DB2 command line processor.
- A federated server and database that are set up to access ODBC data sources.
- The ODBC driver that is installed and configured on the federated server.
- The proper setup of the system environment variables, db2dj.ini variables, and DB2 Profile Registry (db2set) variables. Check the vendor documentation for the required variables for your ODBC client. The LIBPATH variable might be required.

Restrictions:

- The ODBC wrapper does not support the following functions and statements:
  - LOCK TABLE statements on nicknames
  - Features deprecated in ODBC Version 3.x
  - X/Open or SQL/CLI drivers
  - Stored procedure nicknames
  - Statement-level atomicity enforcement using remote savepoint statements
  - 64–bit clients
- Positioned UPDATE and DELETE statements and certain complex, 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 ODBC wrapper supports read and write operations with most data sources.

Procedure:

To add an ODBC data source to a federated server:

1. Prepare the federated server and federated database.
2. Register the wrapper.
3. Register the server definition.
4. Create the user mappings.
5. Test the connection to the ODBC data source.
6. Register nicknames for ODBC data source tables and views.

Related concepts:

- “Fast track to configuring your data sources” on page 1

Related tasks:

- “Checking the federated server setup” in the *DB2 Information Integrator Installation Guide*
- “Preparing the federated server and database to access data sources through ODBC” on page 107

Preparing the federated server and database to access data sources through ODBC

Preparing the federated server and database to access data sources through ODBC is part of the larger task of adding ODBC data sources to federated servers.

The steps that you need to follow to prepare the federated server and database to access data sources through ODBC depend on the operating system that is running on your federated server.

Note: The ODBC driver and the operating system that you are using have unique library path locations.

Procedure:

To prepare the federated server and database:

On Windows:

1. Verify that the ODBC System DSN is set to connect to the ODBC data source. You can use the ODBC Data Source Administrator to configure the DSN. Check this setting in the Control Panel.
   a. From the **Start** menu, open the Control Panel.
   b. Double-click **ODBC Data Sources** to access the ODBC device manager.
   c. Click the System DSN tab to confirm that the System DSN that you defined for the ODBC driver appears on the list.

      The node name for the ODBC data source must be defined in the System DSN.
2. From the ODBC Data Source Administrator window, select **Configure** to test the connection from the ODBC Systems DSN to the ODBC data source.

On UNIX systems:

Consult the ODBC client vendor's documentation for instructions on how to configure the ODBC client.

The next task in this sequence of tasks is registering the ODBC wrapper.

Related tasks:

- "Registering the ODBC wrapper" on page 108

Registering the ODBC wrapper

Registering the ODBC wrapper is part of the larger task of adding ODBC data sources to federated servers.

You must issue the CREATE WRAPPER statement to register an ODBC wrapper.

Procedure:

To specify the wrapper that you want to use to access ODBC data sources, issue the CREATE WRAPPER statement.

For example:

```
CREATE WRAPPER odbc
```

Recommendation: Use the default wrapper name called ODBC when you issue the CREATE WRAPPER statement. When you register the wrapper that uses the default name, the federated server automatically takes the default library name that is associated with that 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 use a name that is different from the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement. Suppose that you have a federated server running on AIX, and you decide to use a wrapper name that is not the default name. An example of the CREATE WRAPPER statement that you need to issue is:

```
CREATE WRAPPER mywrapper
      LIBRARY 'libdb2rcodbc.a' OPTIONS (MODULE '/usr/lib/odbc.a')
```

MODULE *'/usr/lib/odbc.a'* is the full path of the library that contains the ODBC Driver Manager.

You need to register the ODBC wrapper only one time regardless of the number of ODBC data sources that you plan to access. You specify the data source location when you register the server definition. You specify the exact data source object when you register the nickname.

The next task in this sequence of tasks is registering the server definitions for an ODBC data source.

Related tasks:

- "Registering the server definitions for an ODBC data source" on page 110

Related reference:

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*
- "CREATE WRAPPER statement - Examples for ODBC wrapper" on page 109

CREATE WRAPPER statement - Examples for ODBC wrapper

This topic provides examples that show you how to use the CREATE WRAPPER statement to register wrappers for ODBC data sources.

Example for UNIX systems:

The following example shows you how to register a wrapper by issuing the CREATE WRAPPER statement on a UNIX operating system:

```
CREATE WRAPPER odbc OPTIONS (MODULE '/usr/lib/odbc.so')
```

In this example, *odbc* is the name that you assign to the wrapper that is being registered in the federated database. MODULE *'/usr/lib/odbc.so'* is the full path of the library that contains the ODBC Driver Manager.

You must specify the MODULE option on UNIX operating systems. On Windows, the MODULE option defaults to *'odbc32.dll'*.

Example for Windows:

The following example shows you how to register a wrapper by issuing the CREATE WRAPPER statement on a Windows operating system:

```
CREATE WRAPPER odbc LIBRARY 'db2rcodbc.dll'
```

In this example, *odbc* is the name that you assign to wrapper that is being registered in the federated database. LIBRARY *'db2rcodbc.dll'* is the library name for the ODBC wrapper.

The following table lists the wrapper library names for ODBC by operating system:

Table 20. ODBC wrapper library names

Operating system on your federated server	ODBC wrapper library names
AIX	libdb2rcodbc.a
HP-UX	libdb2rcodbc.sl
Linux	libdb2rcodbc.so
Solaris Operating Environment	libdb2rcodbc.so
Windows	db2rcodbc.dll

Related tasks:

- "Registering the ODBC wrapper" on page 108

Related reference:

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*

Registering the server definitions for an ODBC data source

Registering the server definitions for an ODBC data source is part of the larger task of adding ODBC data sources to federated servers.

In the federated database, you must define each ODBC data source server that you want to access.

Procedure:

To register a server definition for an ODBC data source:

Issue the CREATE SERVER statement.

For example:

```
CREATE SERVER server_name TYPE odbc
       VERSION 3.0 WRAPPER odbc_wrapper
        OPTIONS (NODE 'node_name')
```

Although NODE is specified as optional 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.

The next task in this sequence of tasks is creating a user mapping for an ODBC data source.

Related tasks:
- "Creating a user mapping for an ODBC data source" on page 112

Related reference:
- "ALTER SERVER statement" in the *SQL Reference, Volume 2*
- "CREATE SERVER statement" in the *SQL Reference, Volume 2*
- Appendix B, "Server options for federated systems", on page 367
- "CREATE SERVER statement - Examples of ODBC wrapper" on page 111

CREATE SERVER statement - Examples of ODBC wrapper

This topic provides examples that show you how to use the CREATE SERVER statement to register servers for the ODBC wrapper. This topic includes a complete example, which shows how to register a server with required parameters, and an example with additional server options.

Complete example:

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

```
CREATE SERVER server_name TYPE odbc
       VERSION 3.0 WRAPPER odbc_wrapper
        OPTIONS (NODE 'node_name', DBNAME 'venice')
```

server_name
: A name that you assign to the ODBC data source server. This name must be unique. Duplicate server names are not allowed.

TYPE *odbc*
: Specifies the type of data source to which you are configuring access. For the ODBC wrapper, the server type must be *odbc*.

VERSION *3.0*
: The version of the ODBC client that you want to access. All releases of the ODBC standard Version 3 are supported.

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

NODE *'node_name'*
: The name of the node (the system DSN name) that was assigned to the ODBC data source when the DSN was defined. This value is case sensitive. On Windows, this value must be the name of a system DSN in the ODBC Data Administration window. On UNIX, consult the ODBC client vendor documentation for information about the value to use.

Although the NODE is specified as an option in the CREATE SERVER statement, it is required for ODBC data sources.

DBNAME *'venice'*
The name of the database that you want to access. This value is case sensitive.

Server options example:

The following example shows how to use the DB2_TABLE_QUOTE_CHAR and DB2_ID_QUOTE_CHAR 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 supply the following server options in the CREATE SERVER statement:

- DB2_TABLE_QUOTE_CHAR ' '
- DB2_ID_QUOTE_CHAR ' '

For example:

```
CREATE SERVER mysql_server TYPE odbc
       VERSION 3.0 WRAPPER odbc_wrapper
        OPTIONS (NODE 'mysql', DB2_TABLE_QUOTE_CHAR ' ',
                 DB2_ID_QUOTE_CHAR ' ')
```

Related tasks:

- "Registering the server definitions for an ODBC data source" on page 110

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*

Creating a user mapping for an ODBC data source

Creating a user mapping for an ODBC data source is part of the larger task of adding ODBC data sources to federated servers.

When you attempt to access an ODBC data source, the federated server establishes a connection to the data source 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. Create a user mapping for each user ID that will access the federated system to send distributed requests.

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 userid SERVER server_name
      OPTIONS (REMOTE_AUTHID 'remote_id', REMOTE_PASSWORD 'remote_password')
```

The next task in this sequence of tasks is testing the connection to the ODBC data source server.

Related tasks:

- "Testing the connection to the ODBC data source server" on page 114

Related reference:

- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*
- "CREATE USER MAPPING statement - Examples for ODBC wrapper" on page 113

CREATE USER MAPPING statement - Examples for ODBC wrapper

This topic provides examples that show you how to use the CREATE USER MAPPING statement to map a local user ID to an ODBC data source user ID and password. This topic includes a complete example with 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 local user ID to an ODBC data source user ID and password:

```
CREATE USER MAPPING FOR arturo SERVER server_name
      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.

server_name
Specifies the name of the ODBC data source that you defined in the CREATE SERVER statement.

'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 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.

Special register example:

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

```
CREATE USER MAPPING FOR USER SERVER server_name
       OPTIONS (REMOTE_AUTHID 'art', REMOTE_PASSWORD 'red4blue')
```

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.

Related tasks:

- "Creating a user mapping for an ODBC data source" on page 112

Related reference:

- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*

Testing the connection to the ODBC data source server

Testing the connection to the ODBC data source server is part of the larger task of adding ODBC data sources to federated servers.

You can test the connection to the ODBC data source server by using the server definition and the user mappings that you defined.

Prerequisites:

The data source that you are using must support pass-through sessions.

Procedure:

To test the connection:

1. Open a pass-through session to issue an SQL SELECT statement on the ODBC data source system tables.

   For example:

   ```
   SET PASSTHRU server_name
   SELECT COUNT(*) FROM schema_name.table_name
   SET PASSTHRU RESET
   ```

The *server_name* is the name of the ODBC data source that you defined in the CREATE SERVER statement.

The *schema_name* is the name of the schema at the remote ODBC data source. If your ODBC data source does not support schemas, omit the schema from the statement.

The *table_name* is the name of the table at the remote ODBC data source.

If the SQL SELECT statement returns a count, your server definition and your user mappings are set up properly.

2. If the SQL SELECT statement returns an error, you might need to:
   - Verify that the data source is available.
   - If applicable, check the data source server to make sure that it is configured for incoming connections.
   - Check your user mapping to make sure that the settings for the REMOTE_AUTHID and REMOTE_PASSWORD options are valid for the connections to the ODBC data source. Alter the user mapping, or create another user mapping as necessary.
   - Check the ODBC driver on the DB2 federated server to make sure that it is installed and configured correctly to connect to the ODBC data source server. On Windows operating systems, use the ODBC Data Source Administrator tool to check the driver. On UNIX operating systems, consult the ODBC client vendor's documentation.
   - Check your server definition. If necessary, drop it and create it again.

The next task in this sequence of tasks is registering nicknames for ODBC data source tables and views.

Related concepts:

- "Server definitions and server options" on page 10

Related tasks:

- "Checking the federated server setup" in the *DB2 Information Integrator Installation Guide*
- "Checking the data source environment variables" in the *DB2 Information Integrator Installation Guide*
- "Registering nicknames for ODBC data source tables and views" on page 116

Related reference:

- "ALTER USER MAPPING statement" in the *SQL Reference, Volume 2*

Registering nicknames for ODBC data source tables and views

Registering nicknames for ODBC data source tables and views is part of the larger task of adding ODBC data sources to federated servers.

For each ODBC data source server that you define, 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.

In addition to registering nicknames for ODBC data source tables and views, you can also register nicknames for remote system tables.

For example, suppose that you define the nickname *cust_europe* to represent a Microsoft SQL Server table called *italy* with a schema name of *customers*. The SQL statement SELECT * FROM *cust_europe* is allowed from the federated server. However, the statement SELECT * FROM *server_name."customers"."italy"* is not allowed.

If your ODBC data source does not support schemas, omit the schema from the statement.

The federated database relies on catalog statistics for nicknamed objects to optimize query processing. These statistics are gathered when you register a nickname for a data source object by 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 that is useful to the optimizer is read from the data source catalogs and placed into the global catalog on the federated server. Because some or all of the data source catalog information might be used by the optimizer, update statistics (using the data source command that is equivalent to the DB2 **RUNSTATS** command) at the data source before you register a nickname.

Procedure:

To register a nickname, issue the CREATE NICKNAME statement.

For example:

```
CREATE NICKNAME odbc_name FOR server_name."remote_schema"."remote.table"
```

Nicknames can be up to 128 characters in length.

Repeat this step for each ODBC table or view for which you want to create a nickname.

When you create the nickname, DB2 will use the connection to query the data source catalog tables. This query tests your connection to the ODBC data source by using the nickname. If the connection does not work, you will receive an error message.

Related reference:

- “RUNSTATS Command” in the *Command Reference*
- “CREATE NICKNAME statement” in the *SQL Reference, Volume 2*
- “CREATE NICKNAME statement - Examples for ODBC wrapper” on page 117

CREATE NICKNAME statement - Examples for ODBC wrapper

This topic provides an example that shows you how to use the CREATE NICKNAME statement to register a nickname for an ODBC data source table or view that you want to access.

This example shows how to specify a remote object for the ODBC data source under which the nickname is assigned:

```
CREATE NICKNAME cust_europe FOR server_name."customers"."italy"
```

cust_europe
A unique nickname for the table or view. The nickname must be unique within the schema.

Note: The nickname is a two-part name that includes the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authentication ID of the user who registers the nickname.

server_name."customers"."italy"
A three-part identifier for the remote object.

- *server_name* is the name that you assigned to the ODBC database server in the CREATE SERVER statement.
- *customers* is the name of the remote schema to which the table or view belongs. If your ODBC data source does not support schemas, omit the schema from the CREATE NICKNAME statement.
- *italy* is the name of the remote table or view which you want to access.

ODBC data source objects might be case sensitive. Enclose both the remote schema name and the remote table name in quotation marks. Otherwise, DB2 folds these names to uppercase.

Related tasks:

- “Registering nicknames for ODBC data source tables and views” on page 116

Related reference:

- “CREATE NICKNAME statement” in the *SQL Reference, Volume 2*

Tuning and troubleshooting the configuration to ODBC data sources

After you set up the configuration to ODBC data sources, you might want to modify the configuration to improve performance. For example, you might want to set the DB2_DJ_COMM profile registry variable to improve performance when the federated server accesses the ODBC data source.

Improving performance by setting the DB2_DJ_COMM variable

If you find that it takes a long time to access the ODBC remote server, you can improve the performance by setting the DB2_DJ_COMM DB2 profile registry variable. When you set the DB2_DJ_COMM variable, the federated server loads the wrapper upon initialization rather than when you attempt to access the data source.

Procedure:

To set the DB2_DJ_COMM variable:

1. Set the DB2_DJ_COMM variable to the wrapper library that corresponds to the wrapper that you specified.

   For example:

   Suppose that your federated server uses Windows NT, and the wrapper that you are using is ODBC_WRAPPER. The command to set the DB2_DJ_COMM variable is:

   ```
   db2set DB2_DJ_COMM='db2rcodbc.dll'
   ```

   The DB2_DJ_COMM variable is added to the Windows Registry.

   The following table lists the proper ODBC library names by supported operating systems.

Table 21. ODBC wrapper library names

Operating system on your federated server	ODBC wrapper library names
AIX	libdb2rcodbc.a
HP-UX	libdb2rcodbc.sl
Linux	libdb2rcodbc.so
Solaris Operating Environment	libdb2rcodbc.so

Table 21. ODBC wrapper library names (continued)

Windows NT, Windows 2000, and Windows .NET	db2rcodbc.dll, db2rcodbcF.dll, db2rcodbcU.dll

2. Issue the following commands to recycle the DB2 instance:

   ```
   db2stop
   db2start
   ```

   By recycling the DB2 instance, you ensure that the DB2 instance accepts the changes that you made.

Obtaining ODBC traces

If you are experiencing problems when accessing the data source, you can obtain ODBC tracing information to analyze and resolve these problems. Activating a trace impacts system performance. Therefore, you should turn off tracing after you resolve the problems.

On Windows federated servers, use the trace tool provided by the ODBC Data Source Administrator to ensure that the ODBC tracing works properly. On UNIX operating systems, consult the ODBC client vendor’s documentation.

Related tasks:

- “Adding ODBC data sources to federated servers” on page 105

Related reference:

- “db2set - DB2 Profile Registry Command” in the *Command Reference*

Chapter 10. Configuring access to Teradata data sources

This chapter explains how to configure your federated server to access data that is stored in Teradata databases.

This chapter lists the tasks that you need to perform, shows examples of the SQL statements that you need to issue, and provides tuning and troubleshooting information that you can use when you set up the configuration to a Teradata data source.

Adding Teradata data sources to federated servers

To access Teradata data sources from a federated server, you must supply the federated server with information about the Teradata data sources and the objects that you want to access.

You can configure access to Teradata data sources through the DB2 Control Center, through the DB2 Command Center, or through the DB2 command line processor.

The advantage of using the DB2 Control Center is that you do not need to type each statement and command. The DB2 Control Center provides the easiest method to configure access quickly to Teradata data sources. However, you cannot use the DB2 Control Center to issue SQL statements.

You can add a Teradata wrapper to your federated server on any one of the supported operating systems:
- AIX Version 4.3 or later
- Windows NT and Windows 2000

Prerequisites:
- Access to the DB2 Command Center or the DB2 command line processor.
- A federated server and database that are set up to access Teradata data sources.
- Teradata client software that supports the Teradata Call-Level Interface, Version 2 (CLIv2) Release 04.06 or later and is installed and configured on the federated server.
- Access to one or more Teradata servers that are running Teradata release V2R3 or V2R4.

- System environment variables, db2dj.ini variables, and DB2 Profile Registry (db2set) variables that are properly set to access Teradata data sources.

  The db2dj.ini variables that you need to set up properly include:
  - The COPLIB and COPERR variables (AIX only).

    For example:

    ```
    COPLIB=coplib_directory
    COPERR=coperr_directory
    ```

    The *coplib_directory* is the fully qualified path of the directory in which the libcliv2.so file resides. The *coperr_directory* is the fully qualified path of the directory in which the errmsg.txt file resides.

    By default, the installation process places the libcliv2.so file and the errmsg.txt file in the same directory. However, you can specify during the installation process that the libcliv2.so file and the errmsg.txt file reside in different directories.
  - The Teradata NETRACE and COPANOMLOG variables (optional).

    You can enable Teradata tracing if you need to preserve a listing of SQL statements that are sent to the Teradata server.

    For example:

    ```
    NETRACE=1
    COPANOMLOG=trace_file
    ```

    The *trace_file* is the fully qualified name of the file that will contain the trace data.

    These variables enable the Teradata tracing facility only and do not affect the DB2 tracing.

Procedure:

To add a Teradata data source to a federated server:

1. Optional: Test the connection to the Teradata server.
2. Verify that the Teradata library is enabled for run-time linking (AIX).
3. Register the wrapper.
4. Register the server definition.
5. Create the user mappings.
6. Test the connection from the federated server to the Teradata server.
7. Register nicknames for Teradata tables and views.

Related concepts:

- "Fast track to configuring your data sources" on page 1

Related tasks:

- "Testing the connection to the Teradata server" on page 123
- "Checking the data source environment variables" in the *DB2 Information Integrator Installation Guide*

Testing the connection to the Teradata server

Testing the connection to the Teradata server is part of the larger task of adding Teradata data sources to federated servers.

Before you create a wrapper, server definition, or user mapping, you can test the connection to the Teradata server. Test the connection first to verify that the client software is properly set and to prevent errors when you issue the CREATE WRAPPER, CREATE SERVER, and CREATE USER MAPPING statements.

You can use the Basic Teradata Query (BTEQ) utility to submit an SQL query to verify that you can connect to the Teradata server. See the Teradata documentation for more information about the BTEQ utility.

Prerequisite:

Ensure that the BTEQ utility and the Teradata Data Connector Application Program Interface (PIOM) were installed during the Teradata client software installation process.

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 should see the query output on the screen. For example:

   ```
   *** Query completed. One row found. One column returned.
   *** Total elapsed time was 1 second.

       Count(*)
   ___________
            497
   ```

   If the connection is unsuccessful, 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.

The next task in this sequence of tasks is verifying that the Teradata library is enabled for run-time linking.

Related tasks:

- “Verifying that the Teradata library is enabled for run-time linking (AIX)” on page 124

Verifying that the Teradata library is enabled for run-time linking (AIX)

Verifying that the Teradata library is enabled for run-time linking is part of the larger task of adding Teradata data sources to federated servers.

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.

   By default, the installation process places this file in the /usr/lib directory.
2. Issue the following UNIX command.
   ```
   dump - H libcliv2.so | grep libtli.a
   ```
3. Check the file names that appear on the screen.

   If the libtli.a file name appears, the Teradata library is enabled for run-time linking.
4. If the libtli.a file name does not appear, issue the following UNIX commands.
   ```
   rtl_enable libcliv2.so -F libtli.a
   mv libcliv2.so libcliv2.so.old
   mv libcliv2.so.new libcliv2.so
   chmod a+r libcliv2.so
   ```

   These commands enable run-time linking for the Teradata library.

The next task in this sequence of tasks is registering the Teradata wrapper.

Related tasks:

- “Registering the Teradata wrapper” on page 125

Registering the Teradata wrapper

Registering the Teradata wrapper is part of the larger task of adding Teradata data sources to federated servers.

You must issue the CREATE WRAPPER statement to register the Teradata wrapper and to identify the wrapper library to the federated server.

The Teradata wrapper, which is called TERADATA, is included with DB2 Information Integrator.

Procedure:

To specify the wrapper that you want to use to access Teradata data sources, issue the CREATE WRAPPER statement.

For example:

```
CREATE WRAPPER TERADATA
```

Recommendation: Use the default wrapper name TERADATA when you issue the CREATE WRAPPER statement. When you register a wrapper that uses the default name, the federated server automatically takes the default library name that is associated with that 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 use a name that is different from the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement.

For example:

```
CREATE WRAPPER my_wrapper LIBRARY 'library_name'
```

The *my_wrapper* value is the name of the wrapper, and the *library_name* value is the library name for the Teradata wrapper on the operating system that you are using.

On AIX operating systems, the library name for the Teradata wrapper is `libdb2teradata.a`.

On Windows operating systems, the library name for the Teradata wrapper is `db2teradata.dll`.

The next task in this sequence of tasks is registering the server definition for a Teradata data source.

Related tasks:

- "Registering the server definitions for a Teradata data source" on page 126

Related reference:

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*

Registering the server definitions for a Teradata data source

Registering the server definitions for a Teradata data source is part of the larger task of adding Teradata data sources to federated servers.

In the federated database, you must define each Teradata server that you want to access. You must first locate the node name of the Teradata data source, and then use this node name when you register the server.

Procedure:

To register a server definition for a Teradata data source:

1. Locate the node name.
   a. Find the hosts file.

      On AIX operating systems, the hosts file is /etc/hosts.

      On Windows operating systems, the hosts file is *x:*\WINNT\system32\drivers\etc\hosts. *x:* is the drive where the \WINNT directory resides.
   b. 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.
   c. Find the line in the hosts file that contains this alias.
   d. Find the first non-numeric field on that line.

      Example 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, the **nodexyz** field is the node name.
2. Issue the CREATE SERVER statement.

   For example:

   ```
   CREATE SERVER server_name TYPE TERADATA VERSION 4 WRAPPER wrapper
          OPTIONS (NODE 'node_name')
   ```

You must specify a server name. The name that you specify must be unique.

You must set the TYPE parameter to *TERADATA* for all Teradata servers.

The Teradata wrapper supports all versions of both Teradata V2R3 and Teradata V2R4. You can specify the version number as one digit or as two digits with a decimal point. Examples of valid version numbers include 3, 3.0, 3.5, 4, 4.0, 4.4, and so on.

You must specify a name for the wrapper. The name that you specify must correspond to a Teradata wrapper that you registered with the CREATE WRAPPER statement.

You must also specify the name of the node where the Teradata server resides. This node name is case sensitive.

When you register a Teradata server definition, you can specify additional server options in the CREATE SERVER statement, if required.

After you register the server definition, you can add or drop server options by issuing the ALTER SERVER statement.

The next task in this sequence of tasks is creating the user mapping for a Teradata data source.

Related tasks:

- "Creating the user mapping for a Teradata data source" on page 128

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*
- "CREATE SERVER statement - Examples for Teradata wrapper" on page 127

CREATE SERVER statement - Examples for Teradata wrapper

This topic provides several examples that show you how to use the CREATE SERVER statement to register servers for the Teradata wrapper. This topic includes a complete example, which shows how to create a server with all required parameters, and an example with optional server options.

Complete example:

The following example shows you how to create a server definition for a Teradata wrapper by using the CREATE SERVER statement:

```
CREATE SERVER TERASERVER TYPE TERADATA VERSION 4 WRAPPER my_wrapper
       OPTIONS (NODE 'tera_node')
```

The server option *TERASERVER* specifies the name that you assign to the Teradata server. TYPE *TERADATA* specifies that you are configuring access to a Teradata data source. VERSION *4* is the version of the Teradata server software that you want to access. WRAPPER *my_wrapper* specifies the name of the Teradata wrapper that you registered through the CREATE WRAPPER statement. NODE *'tera_node'* is the name of the node where the Teradata server resides.

Server option example:

The following example shows a Teradata server definition with statistics for the optimizer:

```
CREATE SERVER TERASERVER1 TYPE TERADATA
       VERSION 4 WRAPPER WRAPPERNAME1
        OPTIONS (NODE 'tera_node1', CPU_RATIO '2.0', IO_RATIO '3.0')
```

In this example, *TERASERVER1* is the name of the Teradata server, *WRAPPERNAME1* is the wrapper name that you registered through the CREATE WRAPPER statement, and *'tera_node1'* is the name of the node where the Teradata server resides. The CPU_RATIO and IO_RATIO server options provide the following information to the optimizer:

- The CPU resources of the federated server are twice as powerful as the CPU resources of the Teradata server.
- The I/O devices of the federated server process data three times faster than the I/O devices of the Teradata server.

Related tasks:

- "Registering the server definitions for a Teradata data source" on page 126

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*

Creating the user mapping for a Teradata data source

Creating the user mapping for a Teradata data source is part of the larger task of adding Teradata data sources to federated servers.

When you attempt to access a Teradata server, the federated server establishes a connection to the data source 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. Create a user mapping for each user ID that will access the federated system to send distributed requests to the Teradata data source.

You must create user mappings for each Teradata server that you registered in the associated CREATE SERVER statement.

Procedure:

To map the federated user ID to the Teradata server user ID and password, issue a CREATE USER MAPPING statement.

For example:

```
CREATE USER MAPPING FOR USERID SERVER TERASERVER
      OPTIONS (REMOTE_AUTHID 'remote_id', REMOTE_PASSWORD 'remote_password')
```

Alternatively, you can create user mappings by using the Create User Mapping window of the DB2 Control Center.

The next task in this sequence of tasks is testing the connection from the federated server to the Teradata server.

Related tasks:

- "Testing the connection from the federated server to the Teradata server" on page 130

Related reference:

- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*
- "CREATE USER MAPPING statement - Examples for Teradata wrapper" on page 129

CREATE USER MAPPING statement - Examples for Teradata wrapper

This topic provides examples that show you how to use the CREATE USER MAPPING statement to map a local federated user ID to a Teradata server user ID and password. This topic includes a complete example with all 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 local federated user ID (*MICHAEL*) to a Teradata server user ID and password (*'mike'* and *'passxyz123'*):

```
CREATE USER MAPPING FOR MICHAEL SERVER TERASERVER
      OPTIONS (REMOTE_AUTHID 'mike', REMOTE_PASSWORD 'passxyz123')
```

The option *MICHAEL* specifies the federated user ID that you are mapping to a user ID that is defined at the Teradata server. SERVER *TERASERVER* specifies the name of the Teradata server that you defined in the CREATE SERVER statement. REMOTE_AUTHID *'mike'* is the user ID at the Teradata server to which you are mapping the local user ID called *MICHAEL*. REMOTE_PASSWORD *'passxyz123'* is the password that is associated with the REMOTE_AUTHID value of *'mike'*.

Special register example:

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

```
CREATE USER MAPPING FOR USER SERVER TERASERVER
      OPTIONS (REMOTE_AUTHID 'mike', REMOTE_PASSWORD 'passxyz123')
```

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.

Related tasks:

- "Creating the user mapping for a Teradata data source" on page 128

Related reference:

- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*

Testing the connection from the federated server to the Teradata server

Testing the connection from the federated server to the Teradata server is part of the larger task of adding Teradata data sources to federated servers.

You can test the connection from the federated server to the Teradata server by using the server definition and the user mapping that you defined.

Procedure:

To test the connection:

1. From the DB2 command line processor, open a pass-through session to issue an SQL SELECT statement on a Teradata system table.

   For example:

   ```
   SET PASSTHRU server_name
   SELECT count(*) FROM dbc.tables
   SET PASSTHRU RESET
   ```

   If the SQL SELECT statement returns a count, your server definition and your user mapping are set up properly.
2. If the SQL SELECT statement returns an error, you might need to:
   - Check the Teradata server to make sure that it is configured for incoming connections.
   - Check your user mapping to make sure that the settings for the REMOTE_AUTHID and REMOTE_PASSWORD options are valid for the connections to the Teradata server. Alter the user mapping, or create another user mapping as necessary.
   - Check the Teradata client software on the DB2 federated server to make sure that the software is correctly installed and configured to connect to the Teradata server.
   - Check the settings of your DB2 federated variables to verify that you can access the Teradata server. These variables include the system environment variables, the db2dj.ini variables, and the DB2 Profile Registry (db2set) variable.
   - Check your server definition. If necessary, drop the server definition and create it again.

   When you initiate a pass-through session to issue SQL statements on Teradata objects, you cannot submit an SQL PREPARE statement with an INTO parameter if the statement contains host variables.

The next task in this sequence of tasks is registering nicknames for Teradata tables and views.

Related concepts:

- "Server definitions and server options" on page 10

Related tasks:

- "Registering nicknames for Teradata tables and views" on page 132
- "Checking the federated server setup" in the *DB2 Information Integrator Installation Guide*
- "Checking the data source environment variables" in the *DB2 Information Integrator Installation Guide*

Related reference:

- “ALTER USER MAPPING statement” in the *SQL Reference, Volume 2*

Teradata nicknames on federated servers

You must create a nickname for each Teradata® table and view that you want to access on each Teradata server that you defined. Use these nicknames, instead of the names of the data source objects, when you query the Teradata servers.

The federated server connects to the Teradata data source by using the nickname that you assigned with the CREATE NICKNAME statement. The federated server then queries the data source catalog and verifies the connection to the data source. If the connection does not work, DB2® generates an error message.

The federated database relies on catalog statistics for nicknamed objects 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 from that data source. Information that is useful to the optimizer is read from the data source catalogs and placed into the global catalog on the federated server. Because some or all of the data source catalog information might be used by the optimizer, update the statistics at the data source before you create a nickname. Update these statistics at the data source by using a command or utility that is equivalent to the DB2 **RUNSTATS** command.

You cannot submit an SQL INSERT, UPDATE, or DELETE statement to a nickname that references an updatable Teradata view unless that SQL statement can be completely pushed down to the Teradata data source.

Related tasks:

- “Registering nicknames for Teradata tables and views” on page 132

Related reference:

- “RUNSTATS Command” in the *Command Reference*
- “CREATE NICKNAME statement - Examples for Teradata wrapper” on page 133

Registering nicknames for Teradata tables and views

Registering nicknames for Teradata tables and views is part of the larger task of adding Teradata data sources to federated servers.

For each Teradata server that you define, register a nickname for each table and view that you want to access.

Procedure:

To register a nickname, issue the CREATE NICKNAME statement.

For example:

```
CREATE NICKNAME TERANICKNAME FOR TERASERVER."remote_schema"."remote.table"
```

Recommendation: Because the federated database uses catalog statistics for nicknamed objects to optimize query processing, update the statistics at the Teradata data source before registering a nickname. You can use a command or utility that is equivalent to the DB2 **RUNSTATS** command.

Nicknames can be up to 128 characters in length.

You can specify the NUMERIC_STRING column option when you issue the CREATE NICKNAME statement. You can also specify this column option by using the ALTER NICKNAME statement.

Related concepts:

- "Teradata nicknames on federated servers" on page 132

Related reference:

- "RUNSTATS Command" in the *Command Reference*
- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*
- "CREATE NICKNAME statement - Examples for Teradata wrapper" on page 133

CREATE NICKNAME statement - Examples for Teradata wrapper

This topic provides an example that shows you how to use the CREATE NICKNAME statement to register a nickname for a Teradata table or view that you want to access.

This example shows how to specify a remote object for the Teradata server under which the nickname is assigned:

```
CREATE NICKNAME TERASALES FOR TERASERVER."salesdata"."europe"
```

TERASALES is the unique nickname that you assign for the Teradata table or view. A nickname is a two-part name: the schema and the actual nickname. If you omit the schema when you create the nickname, DB2 creates the nickname using your authentication ID as the schema.

TERASERVER."salesdata"."europe" specifies a three-part identifier for the remote object:

- *TERASERVER* is the name that you assigned to the Teradata database server in the CREATE SERVER statement.
- *salesdata* is the name of the remote schema to which the table or view belongs.
- *europe* is the name of the remote table or view that you want to access.

Related concepts:

- "Teradata nicknames on federated servers" on page 132

Related tasks:

- "Registering nicknames for Teradata tables and views" on page 132

Related reference:

- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*

Tuning and troubleshooting the configuration to Teradata data sources

After you set up the configuration to Teradata data sources, you can change the configuration to improve performance and to eliminate potential errors.

Improving access to the Teradata server by setting the DB2_DJ_COMM variable

You might want to set the DB2_DJ_COMM DB2 profile registry variable to improve performance if you find that it takes a long time to access the Teradata server. When you set the DB2_DJ_COMM variable, the federated server loads the wrapper upon initialization rather than when you attempt to access the data source.

Procedure:

To set the DB2_DJ_COMM variable:

1. Set the DB2_DJ_COMM variable to the wrapper library that corresponds to the wrapper that you specified. Use one of the following commands to set the DB2_DJ_COMM DB2 profile registry variable:

   On AIX:

   ```
   db2set DB2_DJ_COMM='libdb2teradata.a','libdb2teradataF.a','libdb2teradataU.a'
   ```

   On Windows:

   ```
   db2set DB2_DJ_COMM='db2teradata.dll','db2teradataF.dll','db2teradataU.dll'
   ```

2. Issue the following commands to recycle the DB2 instance:

```
db2stop
db2start
```

By recycling the DB2 instance, you ensure that the DB2 instance accepts the variable changes that you made.

Tuning and disabling Teradata access logging

The Teradata product provides an access logging feature that generates log entries when Teradata checks the specific security privileges of various users on one or more databases. Although access logging provides considerable and meaningful security information, this feature significantly increases processor usage and can degrade system performance.

If you need to improve system performance, evaluate the checking privilege rules that you defined for access logging. Then, terminate any unnecessary rules by defining END LOGGING statements.

For the best performance, turn off all access logging. Drop the Teradata DBC.AccLogRules macro and then force a trusted parallel application (TPA) reset to stop access logging completely.

See the Teradata documentation for more information.

Enabling run-time linking for libcliv2.so (AIX)

If you run the djxlinkTeradata.sh file to link to the Teradata shared library called `libcliv2.so`, you might receive an error message when you issue a CREATE NICKNAME statement.

An example of an error message that you might receive is:

```
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL30081N  A communication error has been detected.  Communication protocol
being used: "TCP/IP".  Communication API being used: "SOCKETS".  Location
where the error was detected: "9.112.26.28".  Communication function detecting
the error: "recv".  Protocol specific error code(s): "*", "*", "0".
SQLSTATE=08001
```

If you receive an error message, check the /sqllib/db2dump directory for any trap files. Trap file names begin with the letter `t` and end with a suffix of `000`. For example:

```
 t123456.000
```

Check the trace information in the trap file for any OsCall function references that indicate that the OsCall function caused the federated server to stop.

The following example shows trace information with an OsCall function reference that you might find in a trap file:

```
*** Start stack traceback ***

0x239690E0 OsCall + 0x28C
0x23973FB0 mtdpassn + 0x8A4
0x239795A4 mtdp + 0x208
0x2395A928 MTDPIO + 0x28C
0x239609C4 CLICON + 0xD50
0x23962350 DBCHCL + 0xC4
```

If you find an OsCall function reference in one of the trap files, issue the following UNIX commands:

```
cd /usr/lib
rtl_enable libcliv2.so -F libtli.a
mv libcliv2.so libcliv2.so.old
mv libcliv2.so.new libcliv2.so
chmod a+r libcliv2.so
```

These commands enable run-time linking for the `libcliv2.so` shared library.

Related tasks:

- “Adding Teradata data sources to federated servers” on page 121
- “Verifying that the Teradata library is enabled for run-time linking (AIX)” on page 124

Related reference:

- “db2set - DB2 Profile Registry Command” in the *Command Reference*

Chapter 11. Configuring access to OLE DB data sources

This chapter explains how to configure your federated server to access data that is stored in OLE DB data sources.

This chapter lists the tasks that you need to perform and shows examples of the SQL statements that you need to issue when you set up the configuration to OLE DB data sources.

Adding OLE DB data sources to federated servers

Microsoft OLE DB is a set of OLE/COM interfaces that provide applications with uniform access to data that is stored in diverse information sources. The OLE DB component DBMS architecture defines OLE DB consumers and OLE DB providers. An OLE DB consumer is any system or application that consumes OLE DB interfaces. An OLE DB provider is a component that exposes OLE DB interfaces.

The OLE DB wrapper enables you to access OLE DB providers that are compliant with Microsoft OLE DB 2.0 or later.

The OLE DB wrapper is supported on DB2 federated servers that run on the Windows NT or the Windows 2000 operating system.

You use the OLE DB wrapper to create table functions. You cannot use the wrapper to create nicknames on data source tables and views.

Configuring the federated server to access OLE DB data sources involves supplying the federated server with information about the OLE DB providers.

You can configure access to OLE DB data sources through the DB2 Command Center or through the DB2 command line processor.

After you configure access to the OLE DB data source, use the CREATE FUNCTION statement to register a user-defined OLE DB external table function in the federated database.

Prerequisites:

- Access to the DB2 Command Center or the DB2 command line processor.
- A federated server and database that are set up to access OLE DB data sources.

- The OLE DB 2.0 or later driver and an OLE DB provider that are installed and configured on the federated server.

Restriction:

The OLE DB wrapper is used only to assist in registering user-defined OLE DB external table functions. Unlike other wrappers, the OLE DB wrapper does not use nicknames to access data that is stored in data sources.

Procedure:

To add an OLE DB data source to a federated server:

1. Register the wrapper.
2. Register the server definition.
3. Create the user mappings.

Related concepts:

- "Fast track to configuring your data sources" on page 1

Related tasks:

- "Checking the federated server setup" in the *DB2 Information Integrator Installation Guide*
- "Registering the OLE DB wrapper" on page 138

Registering the OLE DB wrapper

Registering the OLE DB wrapper is part of the larger task of adding OLE data sources to federated servers.

You must issue the CREATE WRAPPER statement to register a wrapper that will access OLE DB data sources.

Procedure:

To register the wrapper, issue the CREATE WRAPPER statement.

For example:

```
CREATE WRAPPER OLEDB
```

Recommendation: Use the default wrapper name called OLEDB when you issue the CREATE WRAPPER statement. When you register the wrapper that uses the default name, the federated server automatically takes the default library name that is associated with that 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 use a name that is different from the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement. For example:

```
CREATE WRAPPER mywrapper LIBRARY 'db2oledb.dll'
```

The next task in this sequence of tasks is registering the server definitions for an OLE DB data source.

Related tasks:

- "Registering the server definitions for an OLE DB data source" on page 139

Related reference:

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*

Registering the server definitions for an OLE DB data source

Registering the server definitions for an OLE DB data source is part of the larger task of adding OLE DB data sources to federated servers.

In the federated database, you must define each OLE DB data source server that you want to access.

Procedure:

To register a server definition for an OLE DB data source:

Issue the CREATE SERVER statement.

For example:

```
CREATE SERVER server_name WRAPPER OLEDB
       OPTIONS (CONNECTSTRING 'Provider=Microsoft.Jet.OLEDB.4.0;
                              Data Source=c:\msdasdk\bin\oledb\nwind.mdb')
```

The next task in this sequence of tasks is creating a user mapping for an OLE DB data source.

Related tasks:

- "Creating a user mapping for an OLE DB data source" on page 141

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*
- "CREATE SERVER statement - Examples for OLE DB wrapper" on page 140

CREATE SERVER statement - Examples for OLE DB wrapper

This topic provides an example that shows you how to use the CREATE SERVER statement to register servers for the OLE DB wrapper.

The following example shows a CREATE SERVER statement:

```
CREATE SERVER Nwind WRAPPER OLEDB
OPTIONS (CONNECTSTRING 'Provider=Microsoft.Jet.OLEDB.4.0;
                Data Source=c:\msdasdk\bin\oledb\nwind.mdb',
         COLLATING_SEQUENCE 'Y')
```

Nwind A name that you assign to the OLE DB data source. This name must be unique. Duplicate server 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 initialization properties that are needed to connect to a data source.

The string 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 of the CONNECTSTRING option, see the *Microsoft OLE DB 2.0 Programmer's Reference and Data Access SDK*, Microsoft Press, 1998.

COLLATING_SEQUENCE *'Y'*
Specifies whether the data source uses the same collating sequence as the DB2 for UNIX and Windows collating sequence.

Valid values are 'Y' (the data source uses the DB2 for UNIX and Windows collating sequence) and 'N' (the data source uses a collating sequence that is different from the DB2 for UNIX and Windows collating sequence). The default value is 'N'.

Related tasks:

- "Registering the server definitions for an OLE DB data source" on page 139

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*

Creating a user mapping for an OLE DB data source

Creating a user mapping for an OLE data source is part of the larger task of adding OLE data sources to federated servers.

When you attempt to access an OLE data source, the federated server establishes a connection to the data source 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. Create a user mapping for each user ID that will access the federated system to send distributed requests.

Procedure:

To map a local user ID to the OLE data source user ID and password, issue a CREATE USER MAPPING statement.

For example:

```
CREATE USER MAPPING FOR userid SERVER server_name
      OPTIONS (REMOTE_AUTHID 'remote_id', REMOTE_PASSWORD 'remote_password')
```

Related reference:

- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*
- "CREATE USER MAPPING statement - Examples for OLE DB wrapper" on page 141

CREATE USER MAPPING statement - Examples for OLE DB wrapper

This topic provides examples that show you how to use the CREATE USER MAPPING statement to map a local user ID to an OLE data source user ID and password. This topic includes a complete example with 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 local 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 The local user ID that you are mapping to a user ID that is defined at the OLE DB data source.

SERVER *Nwind*

The name of the OLE DB server that you defined in the CREATE SERVER statement.

REMOTE_AUTHID *'lulu'*

The user ID at the OLE DB server to which you are mapping *laura*. This value is case sensitive.

REMOTE_PASSWORD *'raiders'*

The password that is associated with *'lulu'*. This value is case sensitive.

Special register example:

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')
```

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.

Related tasks:

- "Creating a user mapping for an OLE DB data source" on page 141

Related reference:

- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*

Chapter 12. Configuring access to Table-structured file data sources

This chapter explains what table-structured files are, how to add them as data sources to your federated system, and lists the error messages associated with the table-structured file wrapper.

What are table-structured files?

A table-structured file has a regular structure consisting of a series of records, where each record contains the same number of fields, separated by an arbitrary delimiter. Null values are represented by two delimiters next to each other.

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 drug's unique ID number. The second field is the name of the drug. The third field is the name of the manufacturer who produces the drug.

Related concepts:

- "Types of table-structured files" on page 143
- "How DB2 Information Integrator works with table-structured files" on page 144
- "What is Documentum?" on page 157
- "What is Excel?" on page 191
- "What is BLAST?" on page 205
- "What is XML?" on page 231

Related tasks:

- "Adding table-structured files to a federated system" on page 146

Types of table-structured files

Table-structured files can be sorted or unsorted.

Sorted files

DRUGDATA1.TXT contains sorted records. The file is sorted by the first field, the drug's unique ID number. 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.

```
332,DrugnameB,Manufacturer2
234,DrugnameA,Manufacturer1
333,DrugnameC,Manufacturer2
```

The wrapper can search sorted data files much more efficiently than non-sorted files.

Related concepts:

- "What are table-structured files?" on page 143
- "How DB2 Information Integrator works with table-structured files" on page 144

Related tasks:

- "Adding table-structured files to a federated system" on page 146

How DB2 Information Integrator works with table-structured files

Using a module called a wrapper, DB2® Information Integrator can process SQL statements that query data in a table-structured file as if it were contained in an ordinary relational table or view. This enables data in a table-structured file to be joined with relational data or data in other table-structured files. This process is illustrated in Figure 1 on page 145.

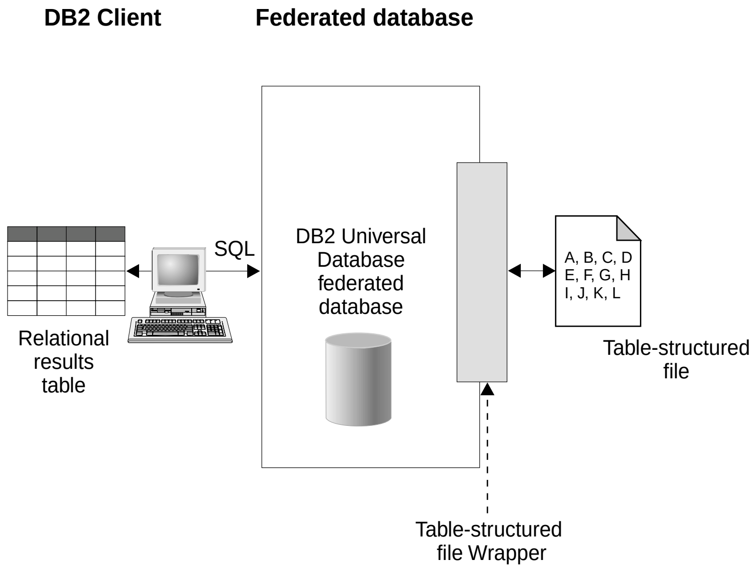


Figure 1. How the table–structured file wrapper works

For example, suppose that the table-structured file DRUGDATA2.TXT is located on your computer in your laboratory. Trying to query this data and match it up with other tables from other data sources that you use can be tedious.

After you register DRUGDATA2.TXT with DB2 Information Integrator, the file behaves as if it is a relational data source. You can now query the file together with other relational and non-relational data sources and analyze the data together.

For example, you could run the following query:

```
SELECT * FROM DRUGDATA2 ORDER BY DCODE
```

This query produces the following results.

Dcode	Drug	Manufacturer
234	DrugnameA	Manufacturer1
332	DrugnameB	Manufacturer2
333	DrugnameC	Manufacturer2

Related concepts:

- “What are table-structured files?” on page 143
- “Types of table-structured files” on page 143

Related tasks:

- "Adding table-structured files to a federated system" on page 146

Adding table-structured files to a federated system

Restrictions:

You cannot use pass-through sessions to access a table-structured file data source.

Procedure:

To add a data source for a table-structured file to a federated server:

1. Register the wrapper using the CREATE WRAPPER command.Register the wrapper using the CREATE WRAPPER command.
2. Optional: Set the DB2_DJ_COMM environment variable to improve query performance.
3. Register the server using the CREATE SERVER command.Register the server using the CREATE SERVER command.
4. Register nicknames using the CREATE NICKNAME command for all table-structured files.Register nicknames using the CREATE NICKNAME command for all table-structured files.

The commands can be run from the DB2 Command Line Processor.

Related tasks:

- "Registering the table-structured file wrapper" on page 146
- "Setting the DB2_DJ_COMM DB2 profile variable for the table-structured file wrapper" on page 147
- "Registering the server for table-structured files" on page 148
- "Registering nicknames for table-structured files" on page 149

Registering the table-structured file wrapper

Registering the table-structured file wrapper is part of the larger task of adding table-structured files to a federated system. You must register the wrapper in order to access a data source. Wrappers are mechanisms that federated servers use to communicate with and retrieve data from data sources. Wrappers are installed on your system as library files.

Procedure:

To register the wrapper, use the CREATE WRAPPER statement to specify which wrapper will be used to access table-structured files.

For example, to register a wrapper on AIX, run the following statement:

```
CREATE WRAPPER laboratory_flat_files LIBRARY 'libdb2lsfile.a'
  OPTIONS(DB2_FENCED 'N');
```

In this example, `laboratory_flat_files` is the name chosen for the wrapper. This name must be unique within the database in which the wrapper is being registered. The required library name for the table-structured file wrapper on AIX is `libdb2lsfile.a`.

The library name is installed as `libdb2lsfile.a` by default, but it might have been customized during installation. Check with your system administrator for the correct name.

For a table of default library filenames for the table-structured file wrapper by supported platform, see the related tasks section.

The next task in this sequence of tasks is setting the DB2_DJ_COMM environment variable for the table-structured file wrapper.

Related tasks:

- "Setting the DB2_DJ_COMM DB2 profile variable for the table-structured file wrapper" on page 147
- "After installing nonrelational wrappers" in the *DB2 Information Integrator Installation Guide*

Related reference:

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*

Setting the DB2_DJ_COMM DB2 profile variable for the table-structured file wrapper

Setting the DB2_DJ_COMM DB2 profile variable for the table-structured file wrapper is part of the larger task of adding table-structured files to a federated system. To improve performance when table-structured files are accessed, set the DB2_DJ_COMM DB2 profile variable. This variable determines whether the federated server loads the wrapper upon initialization.

Procedure:

To set the DB2_DJ_COMM DB2 profile variable, submit the db2set command with the wrapper library that corresponds to the wrapper that you specified in the associated CREATE WRAPPER statement.

For example:

```
db2set DB2_DJ_COMM='libdb2lsfile.a'
```

Ensure that there are no spaces on either side of the equal sign (=).

There is overhead associated with loading the wrapper libraries during database startup. To avoid this overhead, only specify libraries you intend to access.

The next task in this sequence of tasks is registering the server for table-structured files.

Related tasks:

- "Registering the server for table-structured files" on page 148

Registering the server for table-structured files

Registering the server for table-structured files is part of the larger task of adding table-structured files to a federated system. After the wrapper is registered, you must register a corresponding server.

Procedure:

To register the table-structured file server to the federated system, use the CREATE SERVER statement. For example:

```
CREATE SERVER biochem_lab WRAPPER laboratory_flat_files
```

In this example, biochem_lab is the name assigned to the table-structured file server. The name must be unique to the database in which the server is being registered.

The next task in this sequence of tasks is registering nicknames for table-structured files.

Related tasks:

- "Registering nicknames for table-structured files" on page 149

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*

Registering nicknames for table-structured files

Registering nicknames for table-structured files is part of the larger task of adding table-structured files to a federated system. After you register a server, you must register a corresponding nickname. Nicknames are used when you refer to a table-structured file data source in a query.

Nicknames are associated with your table-structured file in one of two ways:

- in a fixed manner using the FILE_PATH nickname option. When this option is used, the nickname represents data from a specific table-structured file.
- with a filename specified at query time 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.

Restrictions:

If a non-numeric 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 char than are allowed by the scale parameter of its column type, the excess data is truncated. The radix character is determined by the RADIXCHAR item of the LC_NUMERIC National Language Support category.

The maximum line length is 32768.

Files containing multibyte characters are not supported.

Procedure:

To register a nickname, use the CREATE NICKNAME statement for each table-structured file that you want to access.

There are no further tasks in this sequence of tasks.

Related tasks:

- "After installing nonrelational wrappers" in the *DB2 Information Integrator Installation Guide*

Related reference:

- "CREATE NICKNAME statement syntax - Table-structured file wrapper" on page 347
- "CREATE NICKNAME statement - Examples for table-structured file wrapper" on page 150

CREATE NICKNAME statement - Examples for table-structured file wrapper

This topic provides a complete example of using a CREATE NICKNAME statement to register nicknames for the table-structured file wrapper. It also includes examples for specific options.

Complete example:

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')
```

KEY COLUMN option examples:

These examples show that the column designated as the key is designated not nullable by adding the NOT NULL option to its definition 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')
```

This option is case-sensitive. However, DB2 folds column names to upper case unless the column is defined with double quotes. The following example will not work properly because the `empno` column will be folded to uppercase by DB2, and the `empno` key column will be submitted in lowercase. Thus the column designated as the key will not be found.

```
CREATE NICKNAME depart (
 empno char(6) NOT NULL)
 FOR SERVER DATASTORE
 OPTIONS(FILE_PATH'data.txt', SORTED 'Y', KEY_COLUMN 'empno');
```

Related tasks:

- “Registering nicknames for table-structured files” on page 149

Related reference:

- “CREATE NICKNAME statement syntax - Table-structured file wrapper” on page 347

File access control model for the table-structured file wrapper

The database management system will access table-structured files with the authority of the DB2 instance owner. The wrapper can only access files that can be read by this user ID (or group ID). The authorization ID of the application (the ID that establishes the connection to the federated database) is not relevant.

On DB2 Universal Database Enterprise Server Edition, 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 DB2 Universal Database node as long as it can be accessed from any node with a common path.

To access a table-structured file on a mapped drive if the network has a Windows NT or Windows 2000 domain configured, the DB2 service logon account must be an account from the domain that has access to he shared folder where data files reside.

To access a table-structured file on a mapped drive if the network doesn’t have a Windows NT or Windows 2000 domain, and your user logs on locally to each workstation, the DB2 service logon account should have the same user name and password as a valid user on the machine that shares that folder. That user must be on the permissions list for the shared folder with at least read access.

Related reference:

- “Access control for the Documentum wrapper” on page 184
- “File access control model for the Excel wrapper” on page 198

Optimization tips and considerations for the table-structured file wrapper

- The system can search sorted data files much more efficiently than non-sorted files.
- For sorted files, you can improve performance by specifying a value or range for the key column when submitting a query.
- Statistics for nicknames of table-structured files must be updated manually by updating the SYSSTAT and SYSCAT views.

Related reference:

- “Optimization tips for the BLAST wrapper” on page 228

Messages for the table-structured file wrapper

This section lists and describes messages you might encounter while working with the wrapper for table-structured files.

Table 22. Messages issued by the wrapper for table-structured files

Error Code	Message	Explanation
SQL0405N	The numeric literal ″<literal>″ is not valid because its value is out of range.	A column in the data file, or a predicate value in an SQL statement, contains a value that is out of the possible range for that data type. Correct the data file or redefine the column to a more appropriate type.
SQL0408N	A value is not compatible with the data type of it's assignment target. Target name is ″<column_name>″.	A column in the data file contains characters that are invalid for that data type. Correct the data file or redefine the column to a more appropriate type.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Data source path is NULL″.)	Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Key Column retrieval failure″.)	Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″STAT failed on data source. ERRNO = <error_number>″.)	Ensure that you have the proper directory permissions. Ensure that the file exists.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″No column info found″.)	Contact IBM Software Support.

Table 22. Messages issued by the wrapper for table-structured files (continued)

Error Code	Message	Explanation
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Unsupported operator″.)	Contact IBM Software Support.
SQL1816N	Wrapper ″<wrapper_name>″ cannot be used to access the ″type″ of data source (″<type>″ ″″) that you are trying to define to the federated database.	The server type was invalid. No server type should be specified in the CREATE SERVER statement. Remove the TYPE keyword and value and rerun it.
SQL1822N	Unexpected error code ″ERRNO = <error_number>″ received from data source ″<server_name>″. Associated text and tokens are ″Unable to read file″.	Check the value of the error number. Make sure that the file can be read by the DB2 instance owner. Then rerun the SQL command.
SQL1822N	Unexpected error code ″Data Error″ received from data source ″<server_name>″. Associated text and tokens are ″Data source is a non-standard file″.	The data source file is a directory, socket, or FIFO. Only standard files can be accessed as data source. Change the FILE_PATH option to point to a valid file and reissue the SQL command.
SQL1822N	Unexpected error code ″ERRNO = <error_number>″ received from data source ″<server_name>″. Associated text and tokens are ″File open error″.	The wrapper was unable to open the file. Check the error number to determine why the error occurred. Correct the problem with the data source and reissue the SQL command.
SQL1822N	Unexpected error code ″Data Error″ received from data source ″<server_name>″. Associated text and tokens are ″Key column missing″.	A record retrieved from the data source was missing the key field. The key column must not be null. Correct the data, or register the file with an unsorted nickname.

Table 22. Messages issued by the wrapper for table-structured files (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″Data Error″ received from data source ″<server_name>″. Associated text and tokens are ″File not sorted″.	The file was not sorted on the key column. Do one of the following: change the KEY_COLUMN option to point to the correct column; resort the data file; or register the nickname as an unsorted nickname.
SQL1822N	Unexpected error code ″Data Error″ received from data source ″<server_name>″. Associated text and tokens are ″Key exceeds definition size″.	The key column field read from the data source was larger than the DB2 column definition which could cause the wrapper search routines to function incorrectly. Correct the data or correct the nickname definition, and reregister the nickname.
SQL1822N	Unexpected error code ″Data Error″ received from data source ″<server_name>″. Associated text and tokens are ″Line in data file exceeds 32k″.	A line in the data file exceeded the maximum line length allowed by the wrapper. The line length cannot be greater than 32768. Shorten the length of the line in the data file.
SQL1823N	No data type mapping exists for data type ″<data_type>″ from server ″<server_name>″.	The nickname was defined with an unsupported data type. Redefine the nickname using only supported data types.
SQL1881N	″<option_name>″ is not a valid ″<component>″ option for ″<object_name>″.	The listed value is not a valid option for the listed object. Remove or change the invalid option then resubmit the SQL statement.
SQL1882N	The ″Nickname″ option ″COLUMN_DELIMITER″ cannot be set to ″<delimiter>″ for ″<nickname_name>″.	The column delimiter was more than one character long. Redefine the option with a single character. Then rerun the SQL statement command.
SQL1882N	The ″Nickname″ option ″KEY_COLUMN″ cannot be set to ″<column_name>″ for ″<nickname_name>″.	The column selected as the key column is not defined for this nickname. Correct the KEY_COLUMN option to be one of the sorted columns for this nickname, then reissue the SQL command.
SQL1882N	The ″Nickname″ option ″VALIDATE_DATA_FILE″ cannot be set to ″<option_value>″ for ″<nickname_name>″.	The option value was invalid. Valid values are ″Y″ or ″N″. Correct the option and register the nickname again.

Table 22. Messages issued by the wrapper for table-structured files (continued)

Error Code	Message	Explanation
SQL1883N	″<option_name>″ is a required ″<component>″ option for ″<object_name>″.	A required option for the wrapper was missing from the SQL statement. Add the required option and resubmit the SQL statement.
SQL30090N	Operation invalid for application execution environment. Reason code = ″21″.	You attempted a pass-through session. The table-structured file wrapper does not support pass-through sessions.

Related reference:

- "Messages for the Documentum wrapper" on page 184
- "Messages for the Excel wrapper" on page 198
- "Messages for the BLAST wrapper" on page 228
- "Messages for the XML wrapper" on page 251

Chapter 13. Configuring access to Documentum data sources

This chapter explains what Documentum is, how to add Documentum data sources to your federated system, and lists the error messages associated with the Documentum wrapper.

What is Documentum?

Documentum is document management software that provides management of document content and attributes such as check-in, check-out, workflow, and version management. The Documentum product is a three-tier, client-server system built on top of a relational database.

A Docbase is a Documentum repository that stores document content, attributes, relationships, versions, renditions, formats, workflow, and security. Documentum Query Language (DQL), an extended SQL dialect, is used to query Documentum data. A Docbase is the equivalent of an Oracle® instance or a DB2® database plus document content files. The metadata is stored in the underlying relational database management system (RDBMS), and the content is stored as binary large objects (BLOBs) in the database or as files stored within the file-system of the server system. For more information on Documentum, refer to the Documentum manuals.

The wrapper for Documentum allows you to add a Documentum data source to a DB2 federated system. By adding the Documentum data source to a federated system, you can use SQL statements to access and query objects and registered tables in a Documentum Docbase. You can then integrate this data with other data sources in your federated system without having to move the data out of the native data source. The Documentum wrapper uses a client library to interface with the Documentum server. The Documentum wrapper provides access to two versions of the Documentum server: EDMS 98 (also referred to as version 3) and 4i. Figure 2 on page 158 illustrates how the Documentum wrapper works.

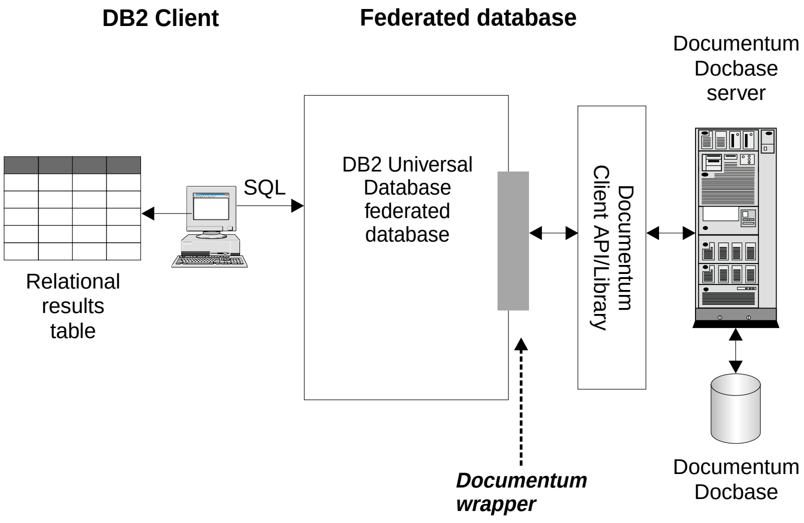

DB2 Client
Federated database
Documentum Docbase server
DB2 Universal Database federated database
SQL
Relational results table
Documentum Client API/Library
Documentum wrapper
Documentum Docbase

Figure 2. How the Documentum wrapper works

After the Documentum wrapper is registered, you can map Documentum Docbase objects and registered tables as relational tables. This is done by mapping Docbase attributes to column names in a DB2 relational table.

For example, Table 23 lists a subset of attributes for the Documentum Docbase default document type, dm_document, along with the associated data. You have determined that this attribute subset is important to you, and you would like to connect these attributes into your federated database system. You named this subset of data DrugAB_data.

Table 23. DrugAB_data

Title	Subject	Authors	Keywords
The effect of drug A on rabbits	Drug A	Curran, L.	rabbits, drug A
Toxicity results for drug A	Drug A	Abelite, P., McMurtrey, K.	toxicity, drug A
Drug B interactions	Drug B	DeNiro, R., Stone, S.	interactions, drug B
Chemical structure of drug B	Drug B	Boyslim, F.	structure, drug B

After you register the Documentum wrapper, the data can be queried using SQL statements.

The following query displays the titles and authors whose subject is Drug A. The result table is shown in Table 24 on page 159.

```
SELECT title, authors
 FROM drugAB_data
 WHERE subject = 'Drug A'
```

Table 24. Query results

Title	Authors
The effect of drug A on rabbits	Curran, L.
Toxicity results for drug A	Abelite, P., McMurtrey, K.

Related concepts:

- "What are table-structured files?" on page 143
- "What is Excel?" on page 191
- "What is BLAST?" on page 205
- "What is XML?" on page 231

Related tasks:

- "Adding Documentum to a federated system" on page 159

Adding Documentum to a federated system

Procedure:

To add the Documentum data source to a federated server:

1. Make the Documentum client library available to DB2.
2. Point to Documentum's client dmcl.ini file
3. Register the wrapper using the CREATE WRAPPER statement.Register the wrapper using the CREATE WRAPPER statement.
4. Optional: Set the DB2_DJ_COMM environment variable to improve query performance.
5. Register the server using the CREATE SERVER statement.Register the server using the CREATE SERVER statement.
6. Give users access to the data source by using the CREATE USER MAPPING statement.
7. Register nicknames using the CREATE NICKNAME statement.Register nicknames using the CREATE NICKNAME statement.
8. Create custom functions using the CREATE FUNCTION statement.

The statements can be run from the DB2 Command Line Processor. Once registered, you can run queries on the data source.

Related tasks:

- “Making the Documentum client library available to DB2 (AIX and Solaris Operating Environment only)” on page 160
- “Pointing to Documentum’s client dmcl.ini file” on page 161
- “Registering the Documentum wrapper” on page 162
- “Setting the DB2_DJ_COMM DB2 profile variable for the Documentum wrapper” on page 163
- “Registering the server for Documentum data sources” on page 164
- “Mapping users (Documentum wrapper)” on page 164
- “Registering nicknames for Documentum data sources” on page 165
- “Registering custom functions for Documentum data sources” on page 172

Making the Documentum client library available to DB2 (AIX and Solaris Operating Environment only)

Making the Documentum client library available to DB2 (AIX and Solaris Operating Environment only) is part of the larger task of adding Documentum to a federated system.

The client library must be available to DB2 in order for the wrapper to function correctly.

Prerequisites:

The Documentum wrapper uses Version 3.1.7a of the client library. If you are using Documentum 4i , you will need to acquire the older version of the client library from Documentum (if it is not already installed).

Procedure:

To make the Documentum client library available to DB2, copy the client library into the appropriate directory. See Table 25 for the client library names and copy to directories for each supported operating system. You can also make a symbolic link for the client library into the appropriate directory.

Table 25. Client library and copy to directory by operating system

Operating System	Client Library	Copy to directory
AIX	libdmcl.a	`sqllib/lib`
Solaris Operating Environment	libdmcl.so	`sqllib/lib`
Windows	dmcl32.dll	`x:\sqllib\bin`

The next task in this sequence of tasks is pointing to Documentum's client dmcl.ini file.

Related tasks:

- "Pointing to Documentum's client dmcl.ini file" on page 161

Pointing to Documentum's client dmcl.ini file

Pointing to Documentum's client dmcl.ini file is part of the larger task of adding Documentum to a federated system. Access to Documentum Docbases are controlled through the Documentum client's dmcl.ini file. A DB2 instance must have its environment variables set to the Documentum client's dmcl.ini file in order to gain access to a Documentum Docbase.

Procedure:

To set the environment variables:

1. Edit the db2dj.ini file, and set either the `DOCUMENTUM` or `DMCL_CONFIG` environment variable.

   The following examples shows how these variables would look on Unix operating systems.

   ```
   DOCUMENTUM=<path>
   ```

   or

   ```
   DMCL_CONFIG=<path>/dmcl.ini
   ```

   where <path> is the fully qualified directory that contains the dmcl.ini file that you want to use.

   The default path to the location of Documentum's dmcl.ini file is /pkgs/documentum. If both lines are included, DMCL_CONFIG is used. On Windows operating systems, a backslash would be used instead of the forward-slash to define the location of the dmcl.ini file.

   On AIX and Solaris Operating Environment, the db2dj.ini file is located in `$HOME/sqllib/cfg`.

   On Windows, the db2dj.ini file is in `x:\sqllib\cfg` where **x:** represents the drive on which the sqllib directory is located.

   Ensure that the name of a docbroker, to which all accessible Docbases for the DB2 instance report, is specified in the dmcl.ini file as shown in Figure 3 on page 162.

```
################# DOCUMENTUM CLIENT CONFIGURATION FILE #####################
#
# Copyright Documentum 1994.
# Version 3.1 of the Documentum Server.
#
# A generated client init file for the Documentum Server.
#
# The only REQUIRED information in this file is the
# [DOCBROKER_PRIMARY] section and an entry for host.
# The host value should be the name of host on which
# your network wide DocBroker is running

[DOCBROKER_PRIMARY]
host = server16.comp2.big.com
```

Figure 3. Sample dmcl.ini file with docbroker name specified

2. Recycle the DB2 instance by issuing the following commands:

   ```
   db2stop
   db2start
   ```

The next task in this sequence of tasks is registering the Documentum wrapper.

Related tasks:

- "Registering the Documentum wrapper" on page 162

Registering the Documentum wrapper

Registering the Documentum wrapper is part of the larger task of adding Documentum to a federated system. You must register the wrapper in order to access a data source. Wrappers are mechanisms that federated servers use to communicate with and retrieve data from data sources. Wrappers are installed on your system as library files.

Procedure:

To register the Documentum wrapper, submit the CREATE WRAPPER statement.

For example, to create a Documentum wrapper on AIX called `Dctm_Wrapper` from the default library file, `libdb2lsdctm.a`, submit the following statement:

```
CREATE WRAPPER Dctm_Wrapper LIBRARY 'libdb2lsdctm.a'
  OPTIONS(DB2_FENCED 'N');
```

For a table of default library filenames for the Documentum wrapper by supported platform, see the related tasks section.

The next task in this sequence of tasks is setting the DB2_DJ_COMM environment variable for the Documentum wrapper.

Related tasks:

- "Setting the DB2_DJ_COMM DB2 profile variable for the Documentum wrapper" on page 163
- "After installing nonrelational wrappers" in the *DB2 Information Integrator Installation Guide*

Related reference:

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*

Setting the DB2_DJ_COMM DB2 profile variable for the Documentum wrapper

Setting the DB2_DJ_COMM DB2 profile variable for the Documentum wrapper is part of the larger task of adding Documentum to a federated system. To improve performance when Documentum data sources are accessed, set the DB2_DJ_COMM DB2 profile variable. This variable determines whether the federated server loads the wrapper upon initialization.

Procedure:

To set the DB2_DJ_COMM DB2 profile variable, submit the `db2set` command with the wrapper library that corresponds to the wrapper that you specified in the associated CREATE WRAPPER statement.

For example:

```
db2set DB2_DJ_COMM='libdb2lsdctm.a'
```

Ensure that there are no spaces on either side of the equal sign (=).

There is overhead associated with loading the wrapper libraries during database startup. To avoid this overhead, only specify libraries you intend to access.

The next task in this sequence of tasks is registering the server for Documentum data sources.

Related tasks:

- "Registering the server for Documentum data sources" on page 164

Registering the server for Documentum data sources

Registering the server for Documentum data sources is part of the larger task of adding Documentum to a federated system. After the wrapper is registered, you must register a corresponding server.

Restrictions:

All servers running on the same instance of DB2 must share the same configuration parameters in the Documentum `dmcl.ini` file.

Procedure:

To register the Documentum server to the federated system, use the CREATE SERVER statement.

For example, suppose there is a server called Dctm_Server1 for the Dctm_Wrapper wrapper created in the associated CREATE WRAPPER statement. Suppose that server contains a Docbase that runs on AIX and uses Oracle to store data. To register the server, submit the following statement:

```
CREATE SERVER Dctm_Server1
 TYPE   DCTM
 VERSION 3
 WRAPPER Dctm_Wrapper
 OPTIONS( NODE 'Dctm_Docbase',
    OS_TYPE 'AIX',
    RDBMS_TYPE 'ORACLE');
```

The next task in this sequence of tasks is mapping users.

Related tasks:

- “Mapping users (Documentum wrapper)” on page 164

Related reference:

- “CREATE SERVER statement” in the *SQL Reference, Volume 2*
- “CREATE SERVER statement arguments and options - Documentum wrapper” on page 357

Mapping users (Documentum wrapper)

Mapping users (Documentum wrapper) is part of the larger task of adding Documentum to a federated system. You must map users to the previously defined servers to give them access to the data source.

Procedure:

To map users to your federated servers, use the CREATE USER MAPPING statement.

For example, the following CREATE USER MAPPING statement maps user `Chuck` to user `Charles` on the `Dctm_Server1` server.

```
CREATE USER MAPPING FOR Chuck SERVER Dctm_Server1
OPTIONS(REMOTE_AUTHID 'Charles', REMOTE_PASSWORD 'Charles_pw');
```

You can also define your own user mapping. In the following example, USER is a keyword meaning the current user, not a user named `USER`.

```
CREATE USER MAPPING FOR USER SERVER Dctm_Server1
OPTIONS(REMOTE_AUTHID 'Lisa', REMOTE_PASSWORD 'Lisa_pw')
```

The next task in this sequence of tasks is registering nicknames for Documentum data sources.

Related tasks:

- "Registering nicknames for Documentum data sources" on page 165

Related reference:

- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*
- "CREATE USER MAPPING statement options - Documentum wrapper" on page 361

Registering nicknames for Documentum data sources

Registering nicknames for Documentum data sources is part of the larger task of adding Documentum to a federated system. After you have registered a server and mapped your user information to the server, you must register corresponding nicknames. Nicknames are used when you refer to a Documentum data source in a query.

Restrictions:

- Pass-through sessions are not supported.
- For each connection to a DB2 database made by a DB2 application, the Documentum wrapper can support a maximum of 10 simultaneous Documentum sessions, and each such session can simultaneously manage up to 10 Documentum queries. A single DB2 application can have several queries in progress simultaneously; the lifetime of a query begins when it is submitted to DB2 and ends when the corresponding cursor over the result set is closed. At any given time, across the entire set of queries in progress at that time, no more than 10 nicknames from one Documentum server may

be referenced. Nicknames mentioned in more than one query, or referenced multiple times in a single query, must be counted once for each time they appear.

Procedure:

To register nicknames, use the CREATE NICKNAME statement to create a nickname for each Docbase for each object type or registered table of interest.

Understanding pseudo columns

The CREATE NICKNAME statement also defines 6 pseudo columns. These columns are used to access object content and other information

The pseudo-columns and their definitions are listed in Table 26.

Table 26. Pseudo column names and definitions.

Pseudo column name	Definition
GET_FILE	VARCHAR (255)
GET_FILE_DEL	VARCHAR (255)
GET_RENDITION	VARCHAR (255)
GET_RENDITION_DEL	VARCHAR (255)
HITS	INTEGER
SCORE	DOUBLE

Table 27 lists pseudo columns for SELECT clauses.

Table 27. Pseudo columns for SELECT clauses

Pseudo column name	**Description**
GET_FILE	Retrieves the content file for the current row in addition to the column values. The extension for the content file is its Documentum format name. If a file of the same name exists, it will be overwritten. GET_FILE attempts to get the object's base format. Its value in the row is the fully qualified file name of the file or the string ″no_content.″ For example: `SELECT object_name, get_file FROM ...` The content file is placed in the server directory that is specified by the Server's CONTENT_DIR option. It is also placed in a subdirectory named with your DB2 local name. The subdirectory will be created if it doesn't exist. It's extension will be its DOS extension defined in the Docbase for the document's format type. For example, ″.doc″, for MS Word documents.
GET_FILE_DEL	This function is the same as GET_FILE except GET_FILE_DEL first deletes the file retrieved for the previous row, if any, in that query. Its value in the row is the fully qualified file name of the file or the string ″no_content.″

Table 27. Pseudo columns for SELECT clauses (continued)

Pseudo column name	Description

GET_RENDITION	Retrieves the content file of that rendition, a copy of the original document in a different format, for the current row in addition to the column values.
	
	The extension for the content file is its Documentum format name. If a file of the same name exists, it will be overwritten.
	
	To specify the rendition format, a predicate of the form DCTM.RENDITION_FORMAT(<format) = 1 must be specified in the WHERE clause.
	
	For example:
	`SELECT object_name, get_rendition`
	`FROM ...`
	`WHERE DCTM.RENDITION_FORMAT('pdf')=1`
	
	GET_RENDITION attempts to get the named rendition of the object. Its value in the row is the fully qualified file name of the file or the string ″no_content.″
	
	The content file is placed in the server directory that is specified by the Server’s CONTENT_DIR option. It is also placed in a subdirectory named with your DB2 local name. The subdirectory will be created if it doesn’t exist.
	
	It’s extension will be its DOS extension defined in the Docbase for the document’s format type. For example, ″.doc″, for MS Word documents.

GET_RENDITION_DEL	This function is the same as GET_RENDITION except GET_RENDITION_DEL first deletes the file retrieved for the previous row, if any, in that query. Its value in the row is the fully qualified file name of the file or the string ″no_content.″

Table 28 on page 169 lists pseudo columns for SELECT clauses in queries that contain search clauses.

Table 28. Pseudo columns for SELECT clauses in queries that contain search clauses

Pseudo column name	Description

HITS	Contains an integer number that represents the number of places in the document in which the search criteria was matched.
	
	For example:
	`SELECT r_object_id, object_name, hits`
	`  FROM std_doc`
	`  WHERE DCTM.SEARCH_WORDS ('''workflow'' OR ''flowchart''')=1`
	
	For each document returned, the number of occurrences of the words ″workflow″ and ″flowchart″ within the document's content are summed and returned as the HITS value.
	
	The HITS pseudo column is appropriate when the documents have only one content file. This is the typical case. This pseudo column can be used in a WHERE clause qualification for a SELECT statement. However, it must also be specified in the SELECT clause.

SCORE	Contains a document's relevance ranking.
	
	Use this pseudo column in conjunction with the Documentum's ACCRUE concept operator. Both return a number that indicates how many of the specified words were found in each returned document.
	
	For example:
	`SELECT object_name, score`
	` FROM std_doc`
	` WHERE`
	`DCTM.SEARCH_TOPIC('<ACCRUE>("document","management","workflow")')=1`
	`  AND SCORE >=75`
	
	The statement returns all documents that have either two or three of the specified words in their content. If a document has only one of the words, it is assigned a score of 50 and therefore fails the WHERE clause criteria and is not returned. If two of the three words are found, a document is assigned a score of 75. If all three words are found, the document's score is 88.
	
	The SCORE pseudo column is used for documents that have one content file. This is the typical case.
	
	SCORE can be in a SELECT clause only if the WHERE contains a SEARCH_WORDS() or SEARCH_TOPIC() function. In a WHERE clause, it is used in conjunction with the ACCRUE concept operator.
	
	For information on the ACCRUE concept operator, see the Documentum documentation.

The next task in this sequence of tasks is registering custom functions for Documentum data sources.

Related concepts:

- "What is the CreateNicknameFile utility for the Documentum wrapper?" on page 180

Related tasks:

- "Registering custom functions for Documentum data sources" on page 172

Related reference:

- "CREATE NICKNAME statement syntax - Documentum wrapper" on page 338
- "CREATE NICKNAME statement - Example for Documentum wrapper" on page 170

CREATE NICKNAME statement - Example for Documentum wrapper

The following CREATE NICKNAME statement defines the nickname std_doc. Std_doc is associated with a Documentum Docbase with an object type of dm_document. Table 29 maps the Documentum attributes and data types to DB2 relational column names and data types that are then used to construct the CREATE NICKNAME statement.

Table 29. Mapping of Documentum attributes to DB2 columns for the std_doc nickname

Documentum attribute name	Documentum data type	DB2 column name	DB2 data type	Repeats?	Nullable?
object_name	string(255)	object_name	varchar	No	No
r_object_id	ID	object_id	char(16)	No	No
r_object_type	string(32)	object_type	varchar	No	No
title	string(255)	title	varchar	No	No
subject	string(128)	subject	varchar	No	No
authors	string(32)	author	varchar	Yes	Yes
keywords	string(32)	keyword	varchar	Yes	Yes
r_creation_date	time	creation_date	timestamp	No	Yes
r_modify_date	time	modified_date	timestamp	No	Yes
a_status	string(16)	status	varchar	No	No
a_content_type	string(32)	content_type	varchar	No	No
r_content_size	double	content_size	integer	No	No
owner_name	string(32)	owner_name	varchar	No	Yes

Table 30 describes each Documentum attribute used in the nickname.

Table 30. Description of Documentum attributes for the std_doc nickname

Documentum attribute name	Description
object_name	The user-defined name of the object.
r_object_id	The unique object identifier for this object, set at creation time.
r_object_type	The object's type, set when the object is created.
title	The user-defined title of the object.
subject	The user-defined subject of the object.
authors	The user-defined list of the authors for the object.
keywords	The list of user-defined keywords for the object.
r_creation_date	The date and time that the object was created.
r_modify_date	The date and time that the object was last modified.
a_status	Set by server when a router task is forwarded. The value is taken from the values assigned to attached_task_status in the router object.
a_content_type	The file format of the object's content.
r_content_size	The number of bytes in the content. For multi-page documents, this attribute records the size of the first content associated with the document.
owner_name	The name of the object's owner (the user who created the object).

Table 29 on page 170 translates into the following CREATE NICKNAME statement.

```
CREATE NICKNAME std_doc (
  object_name varchar(255) not null,
  object_id char(16) not null OPTIONS(REMOTE_NAME 'r_object_id'),
  object_type varchar(32) not null OPTIONS(REMOTE_NAME 'r_object_type'),
  title varchar(255) not null,
  subject varchar(128) not null,
  author varchar(32) OPTIONS(REMOTE_NAME 'authors', IS_REPEATING 'Y'),
  keyword varchar(32) OPTIONS(REMOTE_NAME 'keywords', IS_REPEATING 'Y'),
  creation_date timestamp OPTIONS(REMOTE_NAME 'r_creation_date'),
  modifed_date timestamp OPTIONS(REMOTE_NAME 'r_modify_date'),
  status varchar(16) not null OPTIONS(REMOTE_NAME 'a_status'),
  content_type varchar(32) not null OPTIONS(REMOTE_NAME 'a_content_type'),
  content_size integer not null OPTIONS(REMOTE_NAME 'r_content_size'),
  owner_name varchar(32))
FOR SERVER Dctm_Server2 OPTIONS (REMOTE_OBJECT 'dm_document', IS_REG_TABLE 'N')
```

After you submit the CREATE NICKNAME statement, you can use the nickname std_doc to query your federated system. You can also join the std_doc nickname with other nicknames and tables in the federated system.

In the catalog, the number of columns for this nickname is 6 more than what is being specified in the CREATE NICKNAME statement due to the pseudo columns.

You can use the CreateNicknameFile utility to automatically map Documentum types to DB2 types and to create an initial CREATE NICKNAME statement.

Related concepts:

- "What is the CreateNicknameFile utility for the Documentum wrapper?" on page 180

Related tasks:

- "Registering nicknames for Documentum data sources" on page 165

Related reference:

- "CREATE NICKNAME statement syntax - Documentum wrapper" on page 338

Registering custom functions for Documentum data sources

Registering custom functions for Documentum data sources is part of the larger task of adding Documentum to a federated system. You must use the CREATE FUNCTION statement to register several custom functions. You can use these functions to access some of the unique capabilities of Documentum, such as full-text searching and retrieving document content within queries.

Custom functions for predicates are listed in Table 31 on page 174.

References to the TOPIC function are to Documentum function provided as part of its third-party full-text indexing system from Verity, Inc

Restrictions:

Because DB2 does not support the Boolean type, most of the custom functions (except for USER) used in the WHERE clause must do a check for ″=1″ because these functions are defined to return an integer.

For example,

```
"... WHERE DCTM.ANY_EQ(authors,'Dave Winters')=1"
```

Procedure:

To register custom functions, use the CREATE FUNCTION statement.

All custom functions must be registered with the schema name DCTM. The fully-qualified name of each function is DCTM.<function_name>.

The following example registers the ANY_EQ custom function.

```
CREATE FUNCTION DCTM.ANY_EQ (CHAR(), CHAR()) RETURNS INTEGER
 AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION
```

You must register each custom function one time for each DB2 database that has the Documentum wrapper installed.

To assist you in registering custom functions, the sample file, `create_function_mappings.ddl`, is provided in the `sqllib/samples/lifesci/dctm` directory. This file contains definitions for each custom function. You can run this ddl file to register the custom functions for each DB2 database that has the Documentum wrapper installed.

Custom function string argument rules

All arguments passed as strings must adhere to the following rules:

- Each string is enclosed in single quotes.
- Single quotes within strings are expressed by two single quotes.

Using custom functions in queries

The following examples illustrate the use of the custom functions in queries.

To display the object name and author from the std_doc nickname for documents that have one or more authors named 'Dave Winters':

```
SELECT object_name,authors FROM std_doc
WHERE DCTM.ANY_EQ(authors,'Dave Winters')=1
```

To display the object name and author from the std_doc nickname for documents that have one or more authors named 'Dave Winters' or 'Jon Doe':

```
SELECT object_name,authors FROM std_doc
WHERE DCTM.ANY_IN(authors,'Dave Winters','Jon Doe')=1
```

To display the object name and r_object_id, and to retrieve the content file, from the std_doc nickname for documents containing strings like 'Dave Win%' in the authors column:

```
SELECT object_name, r_object_id, get_file FROM std_doc
WHERE DCTM.ANY_LIKE(authors,'Dave Win%')=1
```

Custom function table

Table 31 lists the custom functions for predicates.

Table 31. Custom functions for predicates

Function name	Description
ANY_EQ(arg1, arg2)	Tests a repeating attribute for any value equal to the specified value. Takes two required arguments: **arg1** Specifies the name of a column that represents a repeating attribute. **arg2** Specifies the value to be compared. For example: `... WHERE DCTM.ANY_EQ(authors,'Dave Winters')=1`
ANY_NE(arg1, arg2)	Tests a repeating attribute for any value not equal to the specified value. Takes two required arguments: **arg1** Specifies the name of a column that represents a repeating attribute. **arg2** Specifies the value to be compared. For example: `... WHERE DCTM.ANY_NE(authors,'Dave Winters')=1`
ANY_LT(arg1, arg2)	Tests a repeating attribute for any value less than the specified value. Takes two required arguments: **arg1** Specifies the name of a column that represents a repeating attribute. **arg2** Specifies the value to be compared. For example: `... WHERE DCTM.ANY_LT(num_approvers,4)=1`
ANY_GT(arg1, arg2)	Tests a repeating attribute for any value greater than the specified value. Takes two required arguments: **arg1** Specifies the name of a column that represents a repeating attribute. **arg2** Specifies the value to be compared. For example: `... WHERE DCTM.ANY_GT(num_approvers,3)=1`

Table 31. Custom functions for predicates (continued)

Function name	Description
ANY_LE(arg1, arg2)	Tests a repeating attribute for any value less than or equal to the specified value. Takes two required arguments: **arg1** Specifies the name of a column that represents a repeating attribute. **arg2** Specifies the value to be compared. For example: `... WHERE DCTM.ANY_LE(num_approvers,2)=1`
ANY_GE(arg1, arg2)	Tests a repeating attribute for any value greater than or equal to the specified value. Takes two required arguments: **arg1** Specifies the name of a column that represents a repeating attribute. **arg2** Specifies the value to be compared. For example: `... WHERE DCTM.ANY_GE(num_approvers,1)=1`
ANY_IN(arg1, arg2 – arg11)	Tests a repeating attribute for any of ten values in a specified list of values. Takes 3–11 arguments of the same data type: **arg1** Specifies the name of a column that represents a repeating attribute. **arg2–arg11** Specifies a comma-separated list of values to be compared. For example: `... WHERE DCTM.ANY_IN(authors,'Crick','Watson')=1` The maximum number of values in an ANY_IN custom function for repeating attributes is 10 for a single statement. Multiple statements can be OR'd.

Table 31. Custom functions for predicates (continued)

Function name	Description

ANY_LIKE(arg1, arg2)	Tests a repeating attribute for any value like the specified value. Takes two required arguments:
	arg1 Specifies the name of a column that represents a repeating attribute.
	arg2 Specifies the pattern being compared with sub-strings in single quotes.
	For example:
	`... WHERE DCTM.ANY_LIKE(authors,'Dave Win%')=1`
	`OR DCTM.ANY_LIKE(keywords,'%\_%')=1`
	The escape clause is not supported in ANY_LIKE() predicates.

ANY_NOT_LIKE(arg1, arg2)	Tests a repeating attribute for any value not like the specified value. Takes two required arguments:
	arg1 Specifies the name of a column that represents a repeating attribute.
	arg2 Specifies the pattern being compared with sub-strings in single quotes.
	For example:
	`... WHERE DCTM.ANY_NOT_LIKE(authors,'Dave Win%')=1`
	`OR DCTM.ANY_NOT_LIKE(keywords,'%\_%')=1`
	The escape clause is not supported in ANY_NOT_LIKE() predicates.

ANY_NULL(arg)	Tests a repeating attribute for IS NULL. Takes one required argument that is the name of the repeating attribute or single-valued DATE or TIMESTAMP attribute. For example: `... WHERE DCTM.ANY_NULL(authors)=1`
ANY_NOT_NULL(arg)	Tests a repeating attribute for IS NOT NULL. Takes one required argument that is the name of the repeating attribute. For example: `... WHERE DCTM.ANY_NOT_NULL(authors)=1`

Table 31. Custom functions for predicates (continued)

Function name	Description

ANY_SAME_INDEX(arg1 – arg10)	Tests repeating attributes for values at the same index of each attribute. Takes two to ten of the other ANY_xx() functions.
	
	The following example checks whether a document has at least one author named Ken who is not affiliated with UCD.
	`... WHERE DCTM.ANY_SAME_INDEX(`
	`ANY_EQ(author_name,'Ken'),`
	`DCTM.ANY_NE(author_affiliation,'UCD'))=1`
	
	The maximum number of tests for values at the same index of repeating attributes is 10. The tests must be AND tests that are evaluated left to right.

CABINET(arg) and CABINET_TREE(arg)	Takes one required argument that is the fully-qualified name of a Docbase cabinet.
	
	For example:
	`... WHERE DCTM.CABINET('/Tools')=1`
	`... WHERE DCTM.CABINET_TREE('/MyDocs')=1`
	
	Use multiple instances of CABINET and CABINET_TREE to specify multiple cabinets.
	
	For example:
	`... WHERE DCTM.CABINET('/Tools')=1`
	`OR DCTM.CABINET_TREE('/Parts')=1`

FOLDER(arg) and FOLDER_TREE(arg)	Takes one required argument that is the fully-qualified name of a Docbase folder or cabinet.
	
	For example:
	`... DCTM.FOLDER('/Tools/Drills')=1`
	`... DCTM.FOLDER_TREE('/MyDocs/WhitePapers')=1`
	
	Use multiple instances of FOLDER and FOLDER_TREE to specify multiple folders.
	
	For example:
	`... DCTM.FOLDER('/Tools/Drills')=1`
	`OR DCTM.FOLDER_TREE('/Animals/Horses')=1`

Table 31. Custom functions for predicates (continued)

Function name	Description

RENDITION_FORMAT (format)	Works with the GET_RENDITION and GET_RENDITION_DEL pseudo columns to establish the format of the rendition to be retrieved. Takes a single character string argument specifying the format.
	
	The following example retrieves a document in PDF format:
	`SELECT get_rendition`
	`FROM ....`
	`WHERE DCTM.RENDITION_FORMAT('pdf')=1`

USER(1)	Compares a value to the Documentum author ID of the current user. Due to a limitation of DB2, the custom function USER is defined with an integer argument that is not used. For example: `... WHERE approver = DCTM.USER(1)` To make the Documentum author ID correspond to the DB2 author ID, use the CREATE USER MAPPING statement.

SEARCH_WORDS(arg)	Takes one required string argument that is a list of individual words enclosed in single quotes, separated by AND, OR, or NOT, and using parentheses to control precedence. Words cannot contain white space and must be enclosed in single quotes.
	
	For example:
	`... DCTM.SEARCH_WORDS('''yeast''`
	`AND (''bread'' OR ''cake'')`
	`AND NOT ''wedding''' )=1`

SEARCH_TOPIC(arg)	Takes one required string argument which is a Verity TOPIC query statement that is to be passed to Documentum and Verity verbatim. For example: `... WHERE DCTM.SEARCH_TOPIC('"quick"')=1`

There are no further tasks in this sequence of tasks.

Related reference:

- "CREATE FUNCTION (Sourced or Template) statement" in the *SQL Reference, Volume 2*

Documentum data source – Example queries

After you register the wrapper, you can run SQL queries on the Documentum data source. This section provides several example queries.

To run queries, 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.

The Documentum server and DB2 process the LIKE predicate differently. When a LIKE predicate is pushed down to the Documentum server, the Documentum semantics apply. In the following example when column c1 contains a zero-length string, the predicate will be true for Documentum and false for DB2.

```
c1 LIKE '%'
```

The following query displays all of the Docbase documents for documents named ’Test Document’:

```
SELECT object_name
FROM std_doc
WHERE object_name='Test Document';
```

The following query uses the custom function ANY_EQ to display all the documents where one of the authors is ’Joe Doe’.

```
SELECT object_name
FROM std_doc
WHERE DCTM.ANY_EQ(author,'Joe Doe')=1
```

The following query uses the FOLDER_TREE function and the SEARCH_WORDS function to find all documents in the Approved cabinet that contain the text ″protein″.

```
SELECT object_name
FROM std_doc
WHERE DCTM.FOLDER_TREE('/Approved')=1
      AND DCTM.SEARCH_WORDS('protein')=1
```

The following query uses the GET_FILE pseudo column and the FOLDER_TREE and ANY_IN custom functions to retrieve the name of the files, on the DB2 server, into which the content has been placed for all documents in the Approved cabinet that have any of the authors listed.

```
SELECT object_name, object_id, get_file
FROM std_doc
WHERE DCTM.FOLDER_TREE('/Approved')=1
      AND DCTM.ANY_IN(author, 'Mary Black', 'Joe Carson', 'Peter Miller')=1
```

Related reference:

- “Excel data source – Example queries” on page 195

What is the CreateNicknameFile utility for the Documentum wrapper?

You can use a Docbasic utility named CreateNicknameFile, available for free download, to create an ASCII file that contains a complete definition of any Docbase object or registered table. You can edit the output file to:

- Define custom local names for columns and attributes. The local and remote names are initially the names as they are known in the Docbase.
- Delete unwanted columns and attributes. The only predefined Documentum document type (dm_document) has 59 attributes in EDMS98 and 76 attributes in 4i. Most of these contain metadata for low-level document management and application development. Deleting the attributes that are not of interest can make SELECT * SQL statements more useful without impacting performance.
- Add a value for the FOLDERS option to restrict searches on this nickname to particular Documentum folders.
- Change DATE mappings to TIMESTAMP if that is desired. The utility generates a mapping from DQL DATE to DB2® DATE because that seems the most useful.
- Change CHAR mappings to VARCHAR or vice-versa depending on application insight.

You must install the utility in a Docbase and run it from a Documentum Windows® graphical user interface. The files that the utility generates are specific to the Docbase in which it is installed.

Related tasks:

- “Installing the CreateNicknameFile utility (Documentum wrapper)” on page 181
- “Configuring the CreateNicknameFile utility (Documentum wrapper)” on page 181
- “Mapping the DM_ID object type in Documentum registered tables” on page 182

Installing the CreateNicknameFile utility (Documentum wrapper)

The CreateNicknameFile utility can assist you in writing CREATE NICKNAME statements for your Documentum data sources.

Procedure:

To install the utility:

1. Download the CreateNicknameFile utility from the download section of the DB2 Information Integrator product website.
2. Use the EDMS98 Workspace graphical user interface or the 4i Desktop Client to import the utility, named CreateNicknameFile.txt. You can import the utility as a procedure type into any Docbase cabinet or folder, and you can give it any name you want.
3. Check the **Can be run by user** box on the properties dialog for the newly imported CreateNicknameFile.txt object.

Related concepts:

- "What is the CreateNicknameFile utility for the Documentum wrapper?" on page 180

Related tasks:

- "Configuring the CreateNicknameFile utility (Documentum wrapper)" on page 181
- "Mapping the DM_ID object type in Documentum registered tables" on page 182

Configuring the CreateNicknameFile utility (Documentum wrapper)

The CreateNicknameFile utility can assist you in writing CREATE NICKNAME statements for your Documentum data sources.

Prerequisites:

You must install the CreateNicknameFile utility before it can be configured.

Procedure:

To configure the utility after you install it:

1. Double-click on the utility's icon to run it.
2. Type the Documentum Document/object-type name. The default is dm_document.

Specify dm_registered as the name if you need to create a nickname file for a registered table. If you specify dm_registered, you will also be prompted for the fully-qualified table name in <owner>.<table_name> format. You can use dm_dbo for the owner name if the table is owned by the Docbase owner (the typical case).

The utility assumes a naming convention for the names of nicknames for registered tables. The convention is to prefix the table name with ″rt_″ to indicate ″registered table″. You can change the nickname proposed by the utility if you don't want to use this convention.

3. Type the server name associated with the nickname you are creating.
4. Type the name of the nickname.

   The names of nickname should be self-explanatory and must be unique within the DB2 instance. The utility assumes a naming convention of <server_name>.<object_type> because the same <object_type> might need to be defined to multiple servers. You can change the nickname proposed by the utility if you don't want to follow this convention.
5. Type the name of the output file.

   The default is C:\Temp\nickname.txt. The directory to receive the output file must already exist and writeable to by you.

After you answer the prompts, the nickname file is created and opens in a text editor.

Related concepts:

- "What is the CreateNicknameFile utility for the Documentum wrapper?" on page 180

Related tasks:

- "Registering nicknames for Documentum data sources" on page 165
- "Installing the CreateNicknameFile utility (Documentum wrapper)" on page 181

Mapping the DM_ID object type in Documentum registered tables

The column definitions created by the CreateNicknameFile utility are compliant with the requirements of the Documentum wrapper, including the correct mapping of each data type to the corresponding DB2 data type. The only exception is that Documentum does not support the DM_ID data type in registered tables. The utility assumes that a column in a registered table is used to contain an object ID if it is defined as a string, is 16 characters long, and has a name ending with ″_id″. In the case of the DM_ID data type, the utility maps the column to the DB2 CHAR(16) data type. In all other cases, all string/varchar columns are mapped to the DB2 VARCHAR data type.

Procedure:

To ensure proper data type mapping:

1. Examine the column data type definitions in the output file created by the CreateNicknameFile utility.
2. If the utility mapped a data type of a Documentum column to an incorrect DB2 data type, change the DB2 data type before using the file to register the nickname to DB2.

Related concepts:

- "What is the CreateNicknameFile utility for the Documentum wrapper?" on page 180

Related tasks:

- "Installing the CreateNicknameFile utility (Documentum wrapper)" on page 181
- "Configuring the CreateNicknameFile utility (Documentum wrapper)" on page 181

Dual defining repeating attributes (Documentum wrapper)

To maximize the query capabilities of the wrapper, each attribute must be defined as its true equivalent DB2 data type. That is, Documentum integers must be defined as DB2 integers and so forth. However, these definitions prevent the return of multiple values for non-VARCHAR repeating attributes. For such columns, only the last value is returned.

This restriction exists because, whenever possible, the wrapper returns only one row of results per Docbase object. This restriction is an issue only when repeating attributes are selected. However, you can define a second column for the same remote repeating attribute but with a data type of VARCHAR.

This column name would be used in the SELECT list to return all values as a delimiter-separated list of all its values. (Each column's DELIMITER option specifies the delimiter to be used.)

You should standardize the local names of the multi-value columns. You can standardize the local names of each multi-value column by adding a prefix of ″m_″ to the local name of the column that is defined as its true data type.

For example, suppose you have a nickname column of a Documentum repeating attribute called approval_dates defined with the data type TIMESTAMP. You can create a second nickname column called m_approval_dates and define it as a VARCHAR data type. You can then use m_approval_dates in a SELECT list to return all approval dates in a delimiter-separated list.

You do not need to use dual definitions for repeating attributes whose true data type is VARCHAR.

Access control for the Documentum wrapper

Queries are subject to your permissions in the Docbase. Only those documents to which you have at least read access are included in query results.

Related reference:

- “File access control model for the table-structured file wrapper” on page 151
- “File access control model for the Excel wrapper” on page 198

Messages for the Documentum wrapper

This section lists and describes messages you might encounter while working with the wrapper for Documentum.

Table 32. Messages issued by the wrapper for Documentum

Error Code	Message	Explanation
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″sqlno_crule_save_plans [100]:rc (-2144272209) Empty plan list detect″.)	The SQL query submitted to DB2 could not be processed by the wrapper. Correct the syntax and resubmit.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″dmAPI exec failed: [DM_QUERY_E_BAD_QUAL] error: ″The attribute qualifier, A0, for attribute <column_name>, is not a valid qualifier.″″.)	An incorrect Documentum type or registered table was entered for the REMOTE_OBJECT nickname option. Change the nickname to use the correct Documentum object type or registered table.

Table 32. Messages issued by the wrapper for Documentum (continued)

Error Code	Message	Explanation
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Invalid null column specified″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Nickname specification is empty″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″The Output object is empty or incomplete″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Unexpected number of columns requested″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″No column information found″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Unsupported column type requested″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Incorrect Column definition″.)	Internal programming error. Contact IBM Software Support.

Table 32. Messages issued by the wrapper for Documentum (continued)

Error Code	Message	Explanation
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Inconsistent type; DB2 request != nickname type″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Output parameter is not NULL″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Query output variable is not NULL″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Invalid timestamp length″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Inconsistent number of columns″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Could not access data when converting values″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Failed to initialize the DMCL client″.)	The Documentum client cannot initialize. Contact your system administrator.

Table 32. Messages issued by the wrapper for Documentum (continued)

Error Code	Message	Explanation
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Get_User returned NULL″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Get_Local_User returned NULL″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Begin Transaction failed″.)	Documentum reported that begintrans failed. Contact your system administrator.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Input parameter was not NULL″.)	Internal programming error. Contact IBM Software Support.
SQL901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Dctm functions must be like DCTM.function(...) =1″.)	You did not use =1 as the RHS of the predicate for a Dctm function. Correct the syntax and run the query again.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Invalid column number requested″.)	Internal programming error. Contact IBM Software Support.
SQL1881N	″DELIMITER″ is not a valid ″COLUMN″ option for ″<column-name>″	The DELIMITER option was specified for column <column-name>, but the IS_REPEATING option was not specified.

Table 32. Messages issued by the wrapper for Documentum (continued)

Error Code	Message	Explanation
SQL1882N	The ″SERVER″ option ″RDBMS_TYPE″ cannot be set to ″<option-value>″ for ″<server-name>″.	The value specified for the RDBMS_TYPE server option is invalid. It must be one of the following: DB2, INFORMIX, ORACLE, SQLSERVER or SYBASE.
SQL1882N	The ″SERVER″ option ″TRANSACTIONS″ cannot be set to ″<option-value>″ for ″<server-name>″.	The value specified for the TRANSACTIONS server option is invalid. It must be one of the following: NONE, QUERY, PASSTHRU or ALL.
SQL1882N	The ″NICKNAME″ option ″IS_REG_TABLE″ cannot be set to ″<option-value>″ for ″<nickname>″.	The value specified for the IS_REG_TABLE nickname option is invalid. It must be one of the following: ’Y’ or ’N’.
SQL1882N	The ″NICKNAME″ option ″ALL_VERSIONS″ cannot be set to ″<option-value>″ for ″<nickname>″.	The value specified for the ALL_VERSIONS nickname option is invalid. It must be one of the following: ’Y’ or ’N’.
SQL1882N	The ″SERVER″ option ″OS_TYPE″ cannot be set to ″<option-value>″ for ″<server-name>″	The value specified for the OS_TYPE server option is invalid. It must be: AIX, HPUX, SOLARIS or WINDOWS.
SQL1882N	The ″NICKNAME″ option ″FOLDERS″ cannot be set to ″<option-value>″ for ″<nickname>″	The value specified for the FOLDERS nickname option is invalid. It cannot be specified for a table where IS_REG_TABLE is ’Y’.
SQL1882N	The ″NICKNAME″ option ″VERSIONS″ cannot be set to ″<option-value>″ for ″<nickname>″	The value specified for the VERSIONS nickname option is invalid. It must be one of the following: ’Y’ or ’N’. Moreover, VERSIONS ’Y’ cannot be specified for a table where IS_REG_TABLE is ’Y’.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Invalid column name, IS_REG_TABLE, or IS_REPEATING specified in nickname″	Check the nickname statement for the correct specification of the IS_REG_TABLE, IS_REPEATING, REMOTE_NAME options, and column names.
SQL30090N	Operation invalid for application execution environment. Reason code = ″db2dj.ini missing DOCUMENTUM or DMCL_CONFIG env var″	The required environment variables are not set. Set them in the db2dj.ini file.

Table 32. Messages issued by the wrapper for Documentum (continued)

Error Code	Message	Explanation
SQL30090N	Operation invalid for application execution environment. Reason code = ″Failed to open log file for debugging″	The log file used for troubleshooting is not accessible. Contact your system administrator.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Only one search condition may be specified″	Only one custom search function may be specified per query.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Failed to create content directory″	Make sure the destination directory is writable by the DB2 agent.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Failed to change permissions on content file″	Make sure the target content directory is writable by the db2 agent.

Related reference:

- “Messages for the table-structured file wrapper” on page 152
- “Messages for the Excel wrapper” on page 198
- “Messages for the BLAST wrapper” on page 228
- “Messages for the XML wrapper” on page 251

Chapter 14. Configuring access to Excel data sources

This chapter explains what Excel is, how to add Excel data sources to your federated system, and lists the error messages associated with the Excel wrapper.

What is Excel?

An Excel spreadsheet or workbook is a file created using the Microsoft® (MS) Excel application and has a file extension of xls. DB2® Information Integrator supports spreadsheets from Excel 97 and Excel 2000. Figure 4 illustrates how the Excel wrapper connects your spreadsheets to your federated system.

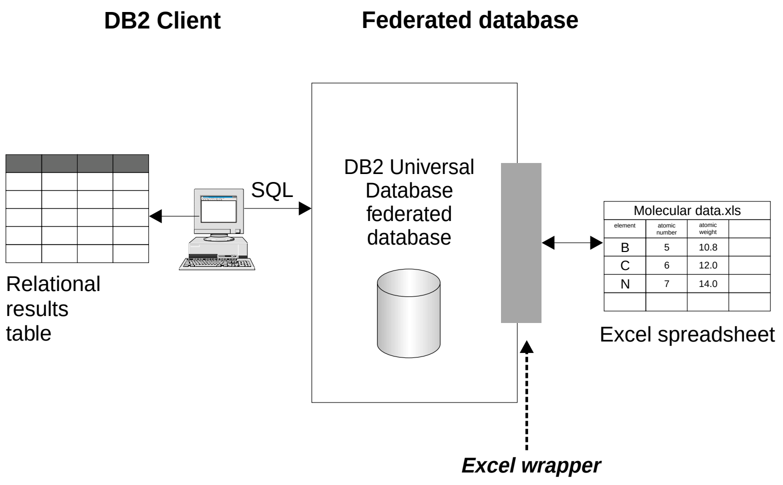


Figure 4. How the Excel wrapper works

The Excel wrapper uses the CREATE NICKNAME statement to map the columns in your Excel spreadsheet to columns in your DB2 federated system. Table 33 shows sample spreadsheet data that is stored in a file called Compound_Master.xls.

Table 33. Sample spreadsheet for Compound_Master.xls

	A	**B**	**C**	**D**
1	compound_A	1.23	367	tested
2	compound_G		210	
3	compound_F	0.000425536	174	tested

Table 33. Sample spreadsheet for Compound_Master.xls (continued)

	A	B	C	D
4	compound_Y	1.00256		tested
5	compound_Q		1024	
6	compound_B	33.5362		
7	compound_S	0.96723	67	tested
8				
9	compound_O	1.2		tested

This information is usually not available to you through standard SQL commands. When the Excel wrapper is installed and registered, you can access this information as if it were a standard 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 Table 34.

Table 34. 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	

Related concepts:

- “What are table-structured files?” on page 143
- “What is Documentum?” on page 157
- “What is BLAST?” on page 205
- “What is XML?” on page 231

Related tasks:

- “Adding Excel to a federated system” on page 193

Adding Excel to a federated system

Procedure:

To add the Excel data source to a federated system:

1. Register the wrapper using the CREATE WRAPPER statement.Register the wrapper using the CREATE WRAPPER statement.
2. Register the server using the CREATE SERVER statement.Register the server using the CREATE SERVER statement.
3. Register nicknames using the CREATE NICKNAME statement for each Excel spreadsheet you want to access.

The commands can be run from the DB2 Command Line Processor.

Related tasks:

- "Registering the Excel wrapper" on page 193
- "Registering the server for an Excel data source" on page 194
- "Registering nicknames for Excel data sources" on page 194

Registering the Excel wrapper

Registering the Excel wrapper is part of the larger task of adding Excel to a federated system. You must register the wrapper in order to access a data source. Wrappers are mechanisms that federated servers use to communicate with and retrieve data from data sources. Wrappers are installed on your system as library files.

Restrictions:

- The Excel wrappers are only available for Microsoft Windows operating systems that support DB2 Universal Database Enterprise Server Edition.
- The MS Excel application must be installed on the server where DB2 Information Integrator is installed before an Excel wrapper can be utilized.
- Pass-through sessions are not allowed.

Procedure:

To register the Excel data source wrapper, submit a CREATE WRAPPER statement.

To create an Excel wrapper for Excel 97 called `Excel_9x_Wrapper` using the library file `db2lsxls.dll`, submit the following statement:

```
CREATE WRAPPER Excel_9x_Wrapper LIBRARY 'db2lsxls.dll'
  OPTIONS(DB2_FENCED 'N');
```

The next task in this sequence of tasks is registering the server for an Excel data source.

Related tasks:

- "Registering the server for an Excel data source" on page 194

Related reference:

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*

Registering the server for an Excel data source

Registering the server for an Excel data source is part of the larger task of adding Excel to a federated system. After the wrapper is registered, you must register a corresponding server.

Procedure:

To register the Excel server to the federated system, use the CREATE SERVER statement.

For example, to create a server called `biochem_lab`, with a node name of `biochem_node1` that registers the server for the Excel_2000_Wrapper wrapper created using the CREATE WRAPPER statement, submit the following statement:

```
CREATE SERVER biochem_lab WRAPPER Excel_2000_Wrapper;
```

The next task in this sequence of tasks is registering nicknames for Excel data sources.

Related tasks:

- "Registering nicknames for Excel data sources" on page 194

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*
- "CREATE SERVER statement arguments - Excel wrapper" on page 359

Registering nicknames for Excel data sources

Registering nicknames for Excel data sources is part of the larger task of adding Excel to a federated system. After you register a server, you must register a corresponding nickname. Nicknames are used when you refer to an Excel data source in a query.

Restrictions:

The wrapper supported date range of the DATE data type is January 1, 1970 to January 18, 2038.

Procedure:

To map the Excel data source to relational tables, create a nickname using the CREATE NICKNAME statement.

The statement in the following example creates the Compounds nickname from the Excel spreadsheet file named `CompoundMaster.xls`. The file contains three columns of data that are being defined to the federated system as `Compound_ID`, `CompoundName`, and `MolWeight`.

```
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:E5');
```

There are no further tasks in this sequence of tasks.

Related reference:

- "CREATE NICKNAME statement syntax - Excel wrapper" on page 341

Excel data source – Example queries

This topic lists several sample Excel spreadsheet queries using the example nickname `Compounds`.

To run queries, 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.

The following query displays all `compound_ID`'s where the molecular weight is greater than 2000:

```
SELECT compound_ID
FROM Compounds
WHERE MolWeight > 200;
```

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;
```

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;
```

Related reference:

- "Documentum data source – Example queries" on page 179
- "Excel data source – Sample scenario" on page 196

Excel data source – Sample scenario

This section demonstrates a sample implementation of the Excel_2000 wrapper accessing an Excel 2000 spreadsheet located in the `C:\Data` directory. The scenario registers the wrapper, registers a server and registers one nickname, that will be used to access the spreadsheet. The statements shown in the scenario are entered using the DB2 Command Line Processor. After the wrapper is registered, you can run queries on the spreadsheet.

The scenario starts with a compound spreadsheet, called `Compund_Master.xls`, with 4 columns and 9 rows. The fully-qualified path name to the file is `C:\Data\Compound_Master.xls`. The contents are show in Table 35.

Table 35. Sample spreadsheet Compound_Master.xls

	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	1.00256		tested
5	compound_Q		1024	
6	compound_B	33.5362		
7	compound_S	0.96723	67	tested
8				
9	compound_O	1.2		tested

Procedure:

To access an Excel 2000 spreadsheet using the Excel wrapper:

1. Register the Excel_2000 wrapper:

   ```
   db2 => CREATE WRAPPER Excel_2000 LIBRARY 'db2lsxls.dll'
      OPTIONS(DB2_FENCED 'N')
   ```

2. Register the server:

   ```
   db2 => CREATE SERVER biochem_lab WRAPPER Excel_2000
   ```

3. Register a nickname that refers to the Excel spreadsheet:

   ```
   db2 => CREATE NICKNAME Compound_Master (compound_name VARCHAR(40),
   weight FLOAT, mol_count INTEGER, was_tested VARCHAR(20))
   FOR biochem_lab
   OPTIONS ( FILE_PATH 'C:\Data\Compound_Master.xls')
   ```

The registration process is complete. The Excel data source is now part of the federated system, and can be used in SQL queries.

The following examples show sample SQL queries and results obtained using the Excel data source.

- Sample SQL query: ″Give me all the compound data where mol_count is greater than 100″

  ```
  SELECT * FROM compound_master WHERE mol_count > 100
  ```

  Result: All fields for rows 1, 2, 3, 5, and 7.
- Sample SQL query: ″Give me the compound_name and mol_count for all compounds where the mol_count has not yet been determined.

  ```
  SELECT compound_name, mol_count FROM compound_master
  WHERE mol_count IS NULL
  ```

  Result: Fields compound_name & mol_count of rows 4, 6 and 9 from the spreadsheet.
- Sample SQL query: ″Count the number of compounds that have not been tested and the weight is greater than 1.″

  ```
  SELECT count(*) FROM compound_master
  WHERE was_tested IS NULL AND weight > 1
  ```

  Result: The record count of 1 which represents the single row 6 from the spreadsheet that meets the criteria.
- Sample SQL query: ″Give me the compound_name and mol_count for all compounds where the mol_count has been determined and is less than the average mol_count.″

  ```
  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 sub-query returns the average 368 to the main query which then returns Table 36:

Table 36. Query results

COMPOUND_NAME	MOL_COUNT
compound_A	367
compound_G	210
compound_F	174
compound_S	67

Related tasks:

- "Adding Excel to a federated system" on page 193

Related reference:

- "Excel data source – Example queries" on page 195

File access control model for the Excel wrapper

The database management system accesses Excel files with the authority of the LOG ON AS property of the DB2 database service. This setting can be viewed in the LOG ON properties page for the DB2 instance. The properties page is accessed through the Windows NT Services control panel.

Related reference:

- "File access control model for the table-structured file wrapper" on page 151
- "Access control for the Documentum wrapper" on page 184

Messages for the Excel wrapper

This section lists and describes messages you might encounter while working with the wrapper for Excel.

Table 37. Messages issued by the wrapper for Excel

Error Code	Message	Explanation
SQL1817N	The CREATE SERVER statement does not identify the ″VERSION″ of data source that you want defined to the federated database.	The VERSION parameter was not specified during the CREATE SERVER statement. Correct the SQL statement and run it again.

Table 37. Messages issued by the wrapper for Excel (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″-1000.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Memory allocation error″	Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1001.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Unknown option″.	The option specified in the DDL statement is not supported. Correct the SQL statement and run it again.
SQL1822N	Unexpected error code ″-1002.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Creation of DELTA object failed″.	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1100.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Wrapper options are not supported″	Wrapper OPTIONS are not supported by this wrapper. Correct the SQL statement and run it again.
SQL1822N	Unexpected error code ″-1200.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″<option> is an unsupported Server option″.	The specified option is not supported by this wrapper. Correct the SQL statement and run it again.
SQL1822N	Unexpected error code ″-1201.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error obtaining server name″	An internal program error has occurred. Contact IBM Software Support.

Table 37. Messages issued by the wrapper for Excel (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″-1209. <internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″ Error converting VARCHAR data″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1211.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″ Error converting INTEGER data″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1212.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″ Error converting FLOAT data″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1400.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″<option> is an unsupported User option″	The specified option is not supported by this wrapper. Correct the SQL statement and run it again.
SQL1822N	Unexpected error code ″-1401.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Creation of USER Delta object failed″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1500.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″<option> is an unsupported Nickname option″	The specified option is not supported by this wrapper. Correct the SQL statement and run it again.

Table 37. Messages issued by the wrapper for Excel (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″-1501.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Required option PATH not specified″	The PATH option is required to register the NICKNAME. Correct the SQL statement and run it again.
SQL1822N	Unexpected error code ″-1502.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Creation of NICKNAME Delta object failed″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1503.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error obtaining Nickname column type″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1504.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error obtaining Nickname column type name″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1505.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″received from data source ″Excel Wrapper″.	The specified <data type> is not supported by this wrapper. Correct the SQL statement and run it again.
SQL1822N	Unexpected error code ″-1506.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error obtaining Nickname column info″	An internal program error has occurred. Contact IBM Software Support.

Table 37. Messages issued by the wrapper for Excel (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″-1507.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″<option> option cannot be dropped″	The specified option cannot be dropped because it is a required option.
SQL1822N	Unexpected error code ″-1508.VANI″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Column names cannot be altered″	The altering of column names is not permitted by the Excel wrapper.
SQL1822N	Unexpected error code. ″-1509.VCTS″ received from data source ″Excel Wrapper″. Associated text and tokens are ″No column info found″.	The column information is not found.
SQL1822N	Unexpected error code ″-1701.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error parsing SQL″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1702.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error accessing NICKNAME object″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1703.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error building data storage area″	An internal program error has occurred. Contact IBM Software Support.

Table 37. Messages issued by the wrapper for Excel (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″-1704.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error linking SQL to Nickname Data″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1705.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Excel application startup failed″	The startup of the Excel application failed. Confirm that Excel is installed on the system and has been registered with the correct version of the wrapper. Check the LOG ON AS property for the DB2 instance in the Windows NT Services control panel. The Excel application will be accessed using this authority. Confirm that this user has appropriate rights or change this property to an authorized account, then restart DB2 and run the SQL query again.
SQL1822N	Unexpected error code ″-1706.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error opening source spreadsheet″	A problem occurred while opening the spreadsheet referenced by the nickname in the SQL query. Ensure that the file exists in the PATH specified during the CREATE NICKNAME statement during registration.
SQL1822N	Unexpected error code ″-1707.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error accessing DL output storage area″	An internal program error occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1708.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Excel application end failed″	An internal program error occurred. If this error persists after repeated queries, contact IBM Software Support.

Table 37. Messages issued by the wrapper for Excel (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″-1711.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error during fetch, possible data/col type mismatch″	The data fetched during the SQL query was of a different data type than the data type specified during the registration of the nickname. Correct the data in the source spreadsheet or correct the registered data type in the nickname. If this does not correct the problem, contact IBM Software Support.
SQL1822N	Unexpected error code ″-1900.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Memory allocation error″	An internal program error has occurred. Contact IBM Software Support.

Related reference:

- "Messages for the table-structured file wrapper" on page 152
- "Messages for the Documentum wrapper" on page 184
- "Messages for the BLAST wrapper" on page 228
- "Messages for the XML wrapper" on page 251

Chapter 15. Configuring access to BLAST data sources

This chapter explains what BLAST is, how to add BLAST data sources to your federated system, and lists the error messages associated with the BLAST wrapper.

What is BLAST?

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 an executable, blastall, 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. These are described in Table 38.

Table 38. 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.

Table 38. BLAST search types supported by the BLAST wrapper (continued)

BLAST search type	Description
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.
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.

Figure 5 on page 207 shows how BLAST works with your federated system.

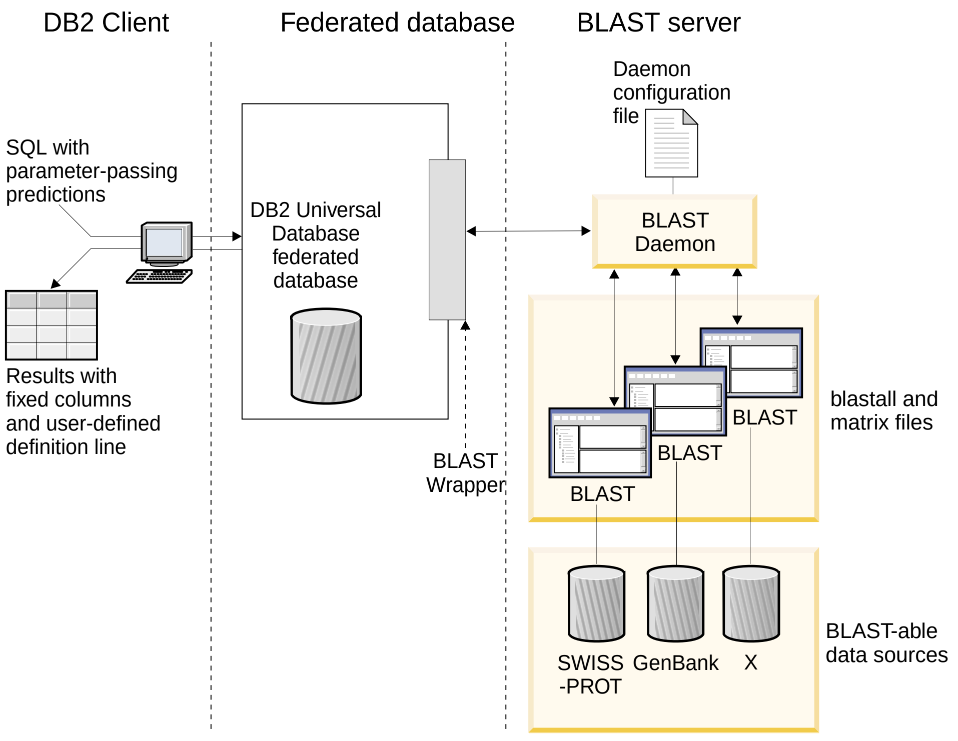

DB2 Client
Federated database
BLAST server
Daemon configuration file
SQL with parameter-passing predictions
DB2 Universal Database federated database
BLAST Daemon
Results with fixed columns and user-defined definition line
BLAST
BLAST
BLAST
blastall and matrix files
BLAST Wrapper
SWISS -PROT
GenBank
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 DB2® Universal Database federated database 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 a separate machine from the machine with the federated system. 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 system 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.

Figure 6 is a sample fasta file containing four definition line and nucleotide sequence records.

```
>7:4986 PMON5744
GTTCTTCCCAGTGCCCAAGTCCATTCTGACATCAATGAAGAAGGTAAAATCCCTGCGTGATCCCTCTGCC
AAGATGTCGAAATCAGACCCGGATAAACTAGCTGCTGTCAGAATAACAGACAGCCCGGAGGAGATCGTGC
AGAAGTTCCGCAAGGCTGTGACGGACTTCACCTCGGAGGTCACCTACGACCCGGCCAGGCGAGGAGGCGT
GTCCAACTTGGTGGCCATCCACGCGGCAGTGACCGGACTCCCGGTGGAGGAGGTGGTCCGCCGAAGTGCT
GGCATCAACACCGCTGGCTACAAGTTGGTGGTGGCGGAGGCTGTGATTGAGAGATTTGCACCAATTAAGA
GTGAAATTGAAAAACTGAAGAGGAACAAGGACCACCTAGAGAAGGTTTTACAAGTTGGGTCGGCAAAAGC
CAAAGAATTAGCATATCCCGTGTGCCAGGAGGTGAAGAAATTGGTGGGGTTTCTATAGGCAGTCTCACCT
AGTCCCAGAAAATGTTTTTTATCTTGTGGTCTGCTTGCACACTCAGTCTAATAAAGGCAGCTTTCCTAAG
ACGCCAACAATTCCAGTTTGGGGATGCTTAGTTTACT
>8:9747 PMON5699
AAGAAGTTCTTGTTAGAACTTTCCACCTCCGGCTTCCCCTCCACCTCTCTTACTGTCCCAACCTTCTGAG
ACGCTTTTTCTCCTCCCGAGGATTTATCTCTTTCTCTCTCTCTCTCTCTCTCTTTTTTTTTTTTCCCCT
TTTCCCCCCCCGAGGCTGGTTTTGCTTTGGGGAGGGGGGGTTTTTTAAAGGGGCCGGGGGGGCCCCCTTT
CTCCCCCCTAATGGGGTTAATTAATAATGGGGGGGGGGGTTTTTTTTTTTTAAACCCCTATTTGGTCCGG
CCCGGGGATTTCCCCCCCCCCCCCCTTGCCCGGTTCCGGGGCCCGGAGGAGGGGGGGAAAAGGGCGGGAA
CCTTTGGTAGTTTCCCCTCGGAAAAAAATTTTTCGGGGGGGAAAACCTCCCT
>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 transforms the fasta file to a BLAST-able data set. The data is now ready for querying by SQL through a federated system with the BLAST wrapper installed and registered.

The following query, sent by you or an application at the client end, is transformed by the BLAST wrapper. It then 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 Table 39.

Table 39. 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.

Related concepts:

- "What are table-structured files?" on page 143
- "What is Documentum?" on page 157
- "What is Excel?" on page 191
- "What is XML?" on page 231

Related tasks:

- "Adding BLAST to a federated system" on page 210

Adding BLAST to a federated system

Procedure:

To add the BLAST data source to a federated server:

1. Verify that the correct version of the blastall executable and matrix files are installed.
2. Configure the BLAST daemon.Configure the BLAST daemon.
3. Start the BLAST daemon.Start the BLAST daemon.
4. Register the wrapper using the CREATE WRAPPER statement.Register the wrapper using the CREATE WRAPPER statement.
5. Optional: Set the DB2_DJ_COMM environment variable to improve query performance.
6. Register the server using the CREATE SERVER statement.Register the server using the CREATE SERVER statement.
7. Register nicknames using the CREATE NICKNAME statement.Register nicknames using the CREATE NICKNAME statement.

The statements can be run from the DB2 command line processor. After the BLAST wrapper is added to your federated system, you can run queries on the BLAST data source.

Related tasks:

- "Verifying that the correct version of the blastall executable and matrix files are installed" on page 211
- "Configuring the BLAST daemon" on page 211
- "Starting the BLAST daemon" on page 214
- "Registering the BLAST wrapper" on page 215
- "Setting the DB2_DJ_COMM DB2 profile variable for the BLAST wrapper" on page 216
- "Registering the server for a BLAST data source" on page 217
- "Registering nicknames for BLAST data sources" on page 217

Verifying that the correct version of the blastall executable and matrix files are installed

Verifying that the correct version of the blastall executable and matrix files are installed is part of the larger task of adding BLAST to a federated system.

Prerequisites:

Verify that you have the latest version of the blastall executable and BLOSUM62, BLOSUM80, PAM30, and PAM70 matrix files installed on your BLAST server machine. If you don't, you must install the binary files and the matrix files. The matrix files must be in the same directory as the blastall executable.

Procedure:

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

1. Run a BLAST search from the command line and note the version number located in the output file.
2. Check this product's website for versions of BLAST that have been tested with this wrapper to ensure you have a supported version.

The next task in this sequence of tasks is configuring the BLAST daemon.

Related tasks:

- "Configuring the BLAST daemon" on page 211

Configuring the BLAST daemon

Configuring the BLAST daemon is part of the larger task of adding BLAST to a federated system.

The BLAST wrapper requires a BLAST daemon to be running on your UNIX-based machine accessible via TCP/IP from your DB2 Universal Database federated system. The daemon runs separately from the wrapper and DB2 Universal Database and listens for BLAST job requests from the wrapper. The daemon executable file, `db2blast_daemon`, can reside in any directory on the BLAST server machine.

During DB2 Universal Database installation, the daemon executable is placed in the `/usr/opt/db2_08_01/bin` directory on AIX, and in the `/opt/IBM/db2/V8.1/bin` directory on the other Unix platforms, of the machine on which the federated server is being installed. If, in your environment, BLAST runs on a different machine, you must copy the daemon to a location of your choice on that machine.

The BLAST daemon must have:

- Execute access to the blastall binary 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 must have read access to both the data file and the BLAST index files generated by the formatdb program.

The BLAST daemon requires a configuration file. A sample daemon configuration file, named `BLAST_DAEMON.config`, is placed in the directory `DB2PATH/samples/lifesci`, where `DB2PATH` is the directory in which DB2 Universal Database is installed. `BLAST_DAEMON.config` is the default name for the file.

Copy the configuration file to any location accessible to the daemon, rename it if you want, and edit it to work with your data source. By default the blast_daemon looks for its configuration information in the working directory from which it was started.

Procedure:

To configure the daemon, specify the following options in the configuration file. For options requiring 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 will listen 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 will create its log file. This file will contain 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 will be 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 will create the temporary file to store the BLAST output data. Data will be read from this file and passed back to the wrapper via the network connection, at which point the daemon cleans up the temporary file.

BLASTALL_PATH
This is the fully-qualified name of the BLAST executable file on the machine running the daemon.

database specification entry
Specifies the location of a BLAST-able data source. For the daemon to function properly, you must specify each entry name used in the configuration file in the DATASOURCE option of the CREATE NICKNAME statement when you create the nickname for the data source.

The configuration file must contain at least one database specification entry in the following form:

```
entry_name = path to BLAST-able_data_source
```

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
```

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 file name of the file that contains the original Fasta-formatted data. The three index files must have the same root file name as the file containing the original Fasta-formatted data.

The first line in the configuration file must be an equal sign. If the equal sign is missing, the daemon will not start up. An error message will indicate that the DAEMON_PORT was not specified.

The last line in the configuration file must be terminated by a new line. If it is not, you will receive an error message when you attempt to run your first BLAST query using the data source listed on the last line. The sample configuration file provided does not have the last line terminated by a newline. For it to run properly, you will need to terminate the last line with by a new line.

Example:

The following example shows the contents of a sample configuration file, with the required options and BLAST-able data source specification for GenBank and SWISS-PROT.

```
=
DAEMON_PORT=4007
MAX_PENDING_REQUESTS=10
DAEMON_LOGFILE_DIR=./
Q_SEQ_DIR_PATH=./
BLAST_OUT_DIR_PATH=./
BLASTALL_PATH=./blastall
genbank=/dsk/1/nucl_data/genbank
swissprot=/dsk/1/prot_data/swissprot
```

The next task in this sequence of tasks is starting the BLAST daemon.

Related tasks:

- “Starting the BLAST daemon” on page 214

Related reference:

- “CREATE NICKNAME statement syntax - BLAST wrapper” on page 336

Starting the BLAST daemon

Starting the BLAST daemon is part of the larger task of adding BLAST to a federated system. Before you can access BLAST data sources, you must have the BLAST daemon running.

Prerequisites:

Before you start the BLAST daemon, you must have write access to all paths listed under the `DAEMON_LOGFILE_DIR`, `BLAST_OUT_DIR_PATH`, and `Q_SEQ_DIR_PATH` entries in the configuration file.

Procedure:

To start the BLAST daemon if you are in the daemon installation directory, did not change the name of the daemon configuration file, and the configuration file is in the same directory as the daemon executable file, type the following command at the command line:

```
db2blast_daemon
```

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

To start the BLAST daemon if you changed the name of the daemon configuration file or are not in the directory in which the daemon configuration file is located, you must use the -c option on the wrapper daemon command to point the daemon executable to the new name or location.

For example, the following command causes the wrapper daemon to look for its configuration information in a file called `BLAST_D.config` in the subdirectory `cfg`.

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

The next task in this sequence of tasks is registering the BLAST wrapper.

Related tasks:

- “Registering the BLAST wrapper” on page 215

Registering the BLAST wrapper

Registering the BLAST wrapper is part of the larger task of adding BLAST to a federated system. You must register the wrapper in order to access a data source. Wrappers are mechanisms that federated servers use to communicate with and retrieve data from data sources. Wrappers are installed on your system as library files.

Procedure:

To register the BLAST wrapper, submit the CREATE WRAPPER statement.

For example, to create a BLAST wrapper on AIX called `my_blast` from the default library file, `libdb2lsblast.a`, submit the following statement:

```
CREATE WRAPPER my_blast LIBRARY 'libdb2lsblast.a'
  OPTIONS(DB2_FENCED 'N');
```

For a table of default library filenames for the BLAST wrapper by supported platform, see the related tasks section.

The next task in this sequence of tasks is setting the DB2_DJ_COMM environment variable for the BLAST wrapper.

Related tasks:

- "Setting the DB2_DJ_COMM DB2 profile variable for the BLAST wrapper" on page 216
- "After installing nonrelational wrappers" in the *DB2 Information Integrator Installation Guide*

Related reference:

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*

Setting the DB2_DJ_COMM DB2 profile variable for the BLAST wrapper

Setting the DB2_DJ_COMM DB2 profile variable for the BLAST wrapper is part of the larger task of adding BLAST to a federated system. To improve performance when BLAST data sources are accessed, set the DB2_DJ_COMM DB2 profile variable. This variable determines whether the federated server loads the wrapper upon initialization.

Procedure:

To set the DB2_DJ_COMM DB2 profile variable, submit the `db2set` command with the wrapper library that corresponds to the wrapper that you specified in the associated CREATE WRAPPER statement.

For example:

```
db2set DB2_DJ_COMM='libdb2lsblast.a'
```

Ensure that there are no spaces on either side of the equal sign (=).

There is overhead associated with loading the wrapper libraries during database startup. To avoid this overhead, only specify libraries you intend to access.

The next task in this sequence of tasks is registering the server for a BLAST data source.

Related tasks:

- "Registering the server for a BLAST data source" on page 217

Registering the server for a BLAST data source

Registering the server for a BLAST data source is part of the larger task of adding BLAST to a federated system. After the wrapper is registered, you must register a corresponding server.

Procedure:

To register the BLAST server to the federated system, use the CREATE SERVER statement.

For each machine on which the BLAST executable and daemon are installed in your environment, you must register one server for each type of BLAST search you want to run using that instance of the BLAST executable and daemon.

For example, to register a server called blast_server1 for the my_blast wrapper created using the CREATE WRAPPER statement that will be used for BLASTn searches, submit the following statement:

```
CREATE SERVER blast_server1
 TYPE blastn
   VERSION 2.1.2
   WRAPPER my_blast
   OPTIONS (NODE 'big_rs.company.com', DAEMON_PORT '4007')
```

The next task in this sequence of tasks is registering nicknames for BLAST data sources.

Related tasks:

- "Registering nicknames for BLAST data sources" on page 217

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*
- "CREATE SERVER statement arguments - BLAST wrapper" on page 357

Registering nicknames for BLAST data sources

Registering nicknames for BLAST data sources is part of the larger task of adding BLAST to a federated system. After you register a server, you must register a corresponding nickname. Nicknames are used when you refer to a BLAST data source in a query.

Procedure:

To register a BLAST nickname, use the CREATE NICKNAME statement. . Since each type of BLAST search is handled by a separate server, you must define a separate nickname for each type of BLAST search that you want to run on a given BLAST-able data source.

The nickname specifies column information for the definition line portion of the data source. All other columns are fixed. For more information on definition line parsing, see “Definition line parsing”. For more information on fixed columns, see “Fixed columns”.

Definition line parsing

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

If you are interested in including the definition line information in your results table, you must specify the definition line columns in the CREATE NICKNAME statement. Each column specification must specify an INDEX option. The DELIMITER option must be specified for each column, except for the last column specified if you want that column to contain the remainder of the definition line.

The definition line fields must be of type integer, float, double, or varchar.

If data are found in the Accession Number field of a BLAST hit, these data are inserted before data in the Definition field of that BLAST hit. The resulting definition line that contains the Accession Number data followed by the Definition field data is parsed by the wrapper.

Fixed columns

The CREATE NICKNAME statement automatically creates fixed columns. The fixed columns do not appear in the CREATE NICKNAME statement, but are part of the nickname definition and can be referenced in SQL queries. There are two types of fixed columns, input and output.

Input fixed columns

Input fixed columns are used as parameter-passing predicates in SQL queries. They pass standard BLAST switches to BLAST. BLAST then runs on the specified data source using these switches. Input fixed columns can also be referenced in the query select list and returned as part of the results table. Input fixed columns are listed in Table 40.

Table 40. Input fixed columns

Name	Data type	Allowed operators	Description
BlastSeq	varchar(32000)	=	Passes the query sequence to the BLAST wrapper.

Table 40. Input fixed columns (continued)

Name	Data type	Allowed operators	Description
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 expect values that should be returned from blastall.
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.
GapAlign	char(1)	=	Indicates to the wrapper whether gapped alignments are permitted in the BLAST output.
Matrix	varchar(50)	=	Determines which substitution matrix is used by blastall to determine the degree of similarity between pairings of amino acids. Only those BLAST search types that compare amino acids to amino acids 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.

Table 40. Input fixed columns (continued)

Name	Data type	Allowed operators	Description
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.
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.
NumberOfAlignments	integer	=	Specifies how many HSP alignments to include in the BLAST output.
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.
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.
WordSize	integer	=	Indicates to blastall the length of the initial hits that blastall initially searches in the database.

Table 40. Input fixed columns (continued)

Name	Data type	Allowed operators	Description
ThresholdEx	integer	=	Indicates the score threshold below which BLAST does not attempt to extend a hit any further.

The supported BLAST search types and switches for each input fixed column are listed in Table 41

Table 41. BLAST search types and switches supported by the input fixed columns

Name	BLAST search types	BLAST switch	Req?	Default
BlastSeq	n, p, x, tn, tx	–l	Y	N/A
E_Value	n, p, x, tn, tx	–e	N	10
QueryStrands	n	S	N	3
GapAlign	n, p, x, tn, tx	–g	N	T
Matrix	p, x, tn, tx	–n	N	BLOSUM62
NMisMatchPenalty	n	–q	N	–3
NMatchReward	n	–r	N	1
FilterSequence	n, p, x, tn, tx	–F	N	T
NumberOfAlignments	n, p, x, tn, tx	–b	N	250
GapCost	n, p, x, tn, tx	–G	N	11
ExtendedGapCost	n, p, x, tn, tx	–E	N	1
WordSize (for Blastn, a value less than 7 is invalid)	n, p, x, tn, tx	–W	N	11 –BLASTn 3 –BLASTp
ThresholdEx	n, p, x, tn, tx	–f	N	0

Output fixed columns

Output fixed columns are returned in the query results table and can be used as predicates. Output fixed columns are listed in Table 42.

Table 42. Output fixed columns

Name	Data type	Description
Score	double	The computed score for an HSP as reported in the BLAST results.
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.
Length	integer	The length of the hit sequence as reported in the BLAST results.
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_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_GAPS	integer	The percent gaps in the alignment defined as the number of gaps divided by the alignment length.
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_HIT_FRAME	integer	The reading frame of the alignment in the hit sequence. Only available for blastx, tblastn, and tblastx type servers.
HSP_Q_Start	integer	The numeric position of the first homologous nucleotide or amino acid on the query sequence.

Table 42. Output fixed columns (continued)

Name	Data type	Description
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.
HSP_H_Start	integer	The numeric position of the first homologous nucleotide or amino acid on the hit sequence.
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.
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.

There are no further tasks in this sequence of tasks.

Related reference:

- "CREATE NICKNAME statement syntax - BLAST wrapper" on page 336
- "CREATE NICKNAME statement - Examples for BLAST wrapper" on page 223

CREATE NICKNAME statement - Examples for BLAST wrapper

The following CREATE NICKNAME statement defines the nickname genbank.

It assumes the definition field in a BLAST result contains the following information:

```
>276342 15:8924 PMON5426
```

where:

276342 The accession field of the BLAST result.

15:8924 PMON5426
The definition field in a BLAST result containing an organism number followed by an experiment number and then a unique identifier.

With this information, the following nickname is created:

```
CREATE NICKNAME genbank (
     acc_num integer   OPTIONS(INDEX '1', DELIMITER ' '),
     org_num integer   OPTIONS(INDEX '2', DELIMITER ':'),
     exp_num  integer   OPTIONS(INDEX '3', DELIMITER ' '),
     u_id  varchar(10)  OPTIONS(INDEX '4'))
     FOR SERVER blast_server1
       OPTIONS(DATASOURCE 'genbank', TIMEOUT '300');
```

The column acc_num would contain 276342, the column org_num would contain 15, the column exp_num would contain 8924, and the column u_id would contain PMON5426.

After you submit the CREATE NICKNAME statement, you can use the nickname genbank to query your federated system. You can also join the genbank nickname with other nicknames and tables in your federated system.

Related tasks:

- “Registering nicknames for BLAST data sources” on page 217

Related reference:

- “CREATE NICKNAME statement syntax - BLAST wrapper” on page 336

Setting up TurboBlast to work with the BLAST wrapper

Restrictions:

TurboBlast does not support certain blastall command options. For example, the gapped alignment option -g F is not supported. If you specify F for the value of the GapAlign’s column in your BLAST nickname, TurboBlast generates an error. For a complete list of unsupported options, refer to the*TurboBlast 2.0 User Guide.*

Procedure:

To set up TurboBlast to work with the BLAST wrapper:

1. Installed and configure the BLAST wrapper. Run a query on a blastable database to test your setup.
2. The BLAST wrapper and TurboBlast support AIX, Linux, Solaris and Windows NT/2000 platforms. The BLAST daemon is not available on

Windows NT/2000 operating systems. The daemon will work with TurboBlast on Windows NT/2000 when the BLAST daemon is available on those operating systems.

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 machine on which you have your BLAST daemon. The BLAST daemon can invoke the `tblastall` command.
4. Be sure to test the TurboBlast system after you've installed and configured TurboBlast. Follow the instructions in the *TurboBlast 2.0 Installation and Reference Guide*.
5. Change your `BLAST_DAEMON.config` file as follows:
   a. Specify the `BLASTALL_PATH` parameter as the complete path of `tblastall`. For example:
      `BLASTALL_PATH=/home/blasttst/turboblast/TBlast-2.1/tblastall`
   b. Specify the blastable database specification entry as the blastable database name you used to upload your blastable database to TurboBlast. The database names are shown when you enter the `listdatabase -l` command under TurboBlast. This TurboBlast database name should be used instead of the path to the blastable data source. For example: `genbank=<the genbank database name in TurboBlast>`
6. Restart the BLAST daemon. The blast daemon invokes `tblastall` instead of `blastall` to do search work on the blastable databases.
7. The log files related to `tblastall` are written to the `DAEMON_LOGFILE_DIR` specified in your `BLAST_DEAMON.config` file. Also check the `STDERR.log` and `STDOUT.log` produced by the blast daemon in the same directory.

Constructing BLAST SQL queries

SQL for BLAST data sources must contain only special input predicates used to pass standard BLAST switches to the blastall executable file.

Restrictions:

To be valid, every query passed to the BLAST wrapper must contain at least the `BlastSeq` input predicate. All other predicates are optional.

Procedure:

To construct a BLAST query, use the input predicates in the WHERE clause of your SQL statement.

The following example shows three input predicates: BlastSeq, GapCost, and NMisMatchPenalty.

```
Select * from blast b where
BlastSeq = 'GTCCAGCC...' AND
GapCost = -10 AND
NMisMatchPenalty = -4;
```

Related tasks:

- "Registering nicknames for BLAST data sources" on page 217

Related reference:

- "BLAST data source – Example queries" on page 226

BLAST data source – Example queries

Several sample BLAST queries are provided to illustrate how queries are constructed for BLAST data sources.

To run queries, use the examples as a guide.

In these queries, the name used for each nickname indicates the type of BLAST search and the data source. This is done so that the registration statements do not need to be listed with each sample query. Also, some of the queries make use of other hypothetical data sources so that these examples can illustrate the behavior of the wrapper when joined with other data sources.

Query 1

```
select *
from blastn_genbank
where BlastSeq =
 'caaccccctccagccgagttgtcaatggcgaggaagctgttccccac';
```

When this SQL statement is executed, the wrapper will perform a BLASTn search of GenBank using the indicated sequence. The wrapper will return all of the available columns, including both the input parameter columns and the BLAST result columns.

Query 2

```
select *
from blastn_genbank
where BlastSeq =
 'caaccccctccagccgagttgtcaatggcgaggaagctgttccccac'
 and GapCost = 8 and NmisMatchPenalty = -4;
```

When this SQL statement is executed, the wrapper will perform a BLASTn search of GenBank using the indicated sequence. In addition, the wrapper will pass the two indicated parameters to the daemon, and they will be passed to the blastall command line. The wrapper will return all of the available columns, including both the input parameter columns and the BLAST result columns.

Query 3

```
select blp.*
from blastp_swissprot blp, protein_db prdb
where prdb.keyword = 'malic enzyme'
 and blp.BlastSeq = prdb.sequence;
```

When this SQL statement is executed, the wrapper will perform zero or more BLASTp searches of SWISS-PROT, depending on the number of sequences returned from a hypothetical protein sequence database. This statement will be broken into two separate queries by DB2, and one BLASTp search will be run for each row that is returned from the hypothetical protein database. The wrapper will return all of the available columns, including both the input parameter columns and the BLAST result columns.

Query 4

```
select Score, E_Value, HSP_Info, HSP_Q_Seq, HSP_H_Seq, HSP_Midline
from blastx_swissprot
where BlastSeq = 'gagttgtcaatggcgagg'
 and GapCost = 8;
```

When this SQL statement is executed, the wrapper will perform a BLASTx search of SWISS-PROT using the indicated sequence. In this case, blastall will translate 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 will be passed to the daemon and then to blastall via the command line. The wrapper will return only those columns that are specifically requested in the query.

Query 5

```
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';
```

When this SQL statement is executed, the wrapper will perform zero or more tBLASTx searches of GenBank, depending on the number of sequences returned from a hypothetical gene expression database. The statement will be broken into two separate queries by DB2, and one tBLASTx search will be 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 and perform the homology search 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 will be passed to the daemon and then to blastall via the command line. The wrapper will return only those columns that are specifically requested in the query.

Related reference:

- "Documentum data source – Example queries" on page 179
- "Excel data source – Example queries" on page 195

Optimization tips for the BLAST wrapper

Running both the wrapper and the daemon on the same server can eliminate potential network communication bottlenecks.

Related reference:

- "Optimization tips and considerations for the table-structured file wrapper" on page 151

Messages for the BLAST wrapper

This section lists and describes messages that you might encounter when working with the wrapper for BLAST.

Table 43. Messages issued by the wrapper for BLAST

Error Code	Message	Explanation
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″sqlno_crule_save_plans [100]:rc (–2144272209) Empty plan list detect″.)	The SQL query submitted to DB2 could not be processed by the wrapper. Correct the syntax and resubmit.

Table 43. Messages issued by the wrapper for BLAST (continued)

Error Code	Message	Explanation
SQL1816N	Wrapper ″BLAST_WRAPPER″ cannot be used to access the ″type″ of data source (″<server type>″ ″″) that you are trying to define to the federated database.	The CREATE SERVER statement used an invalid TYPE. The type must be one of the supported BLAST types.
SQL1817N	The CREATE SERVER statement does not identify the ″version″ of data source that you want defined to the federated database.	The CREATE SERVER statement did not specify the version.
SQL1822N	Unexpected error code ″Unspecified Error″ received from data source ″Blast Wrapper″. Associated text and tokens are ″Unable to connect to daemon″.	The blast wrapper was not able to connect to the daemon. The daemon might not be running. It might be misconfigured. The machine that it is running on might be unreachable.
SQL1822N	Unexpected error code ″Unspecified Error″ received from data source ″Blast Wrapper″. Associated text and tokens are ″Blast daemon timeout expired″.	No results were received from the daemon before the timeout as specified on the CREATE NICKNAME statement elapsed. Increase the timeout or check to see if there is a problem with the daemon.
SQL1822N	Unexpected error code ″Unspecified Error″ received from data source ″Blast Wrapper″. Associated text and tokens are ″Blast Daemon Failed″.	The daemon stopped communicating or the results returned were not properly formatted.
SQL1822N	Unexpected error code ″Unspecified Error″ received from data source ″Blast Wrapper″. Associated text and tokens are ″Unknown error from the blast daemon″.	The blast wrapper received an error code from the daemon that it doesn’t recognize. The daemon version might not be compatible with the wrapper version.
SQL1822N	Unexpected error code ″Unspecified Error″ received from data source ″Blast Wrapper″. Associated text and tokens are ″Column rename not allowed″.	An ALTER NICKNAME statement was issued trying to rename one of the columns. Renaming a column is not allowed.

Table 43. Messages issued by the wrapper for BLAST (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″Unspecified Error″ received from data source ″Blast Wrapper″. Associated text and tokens are ″XML parser error″.	The Xerces parser is in an invalid state or has thrown an exception.
SQL1823N	No data type mapping exists for data type ″<data type name>″ from server ″<server name>″.	The data type specified is not supported by this column.
SQL1881N	″DEFAULT″ is not a valid ″COLUMN″ option for ″<column-name>″	The DEFAULT option was used on a column that does not support it. Output only columns and definition line columns do not have default values.
SQL1882N	The ″COLUMN″ option ″DEFAULT″ cannot be set to ″<option-value>″ for ″<column-name>″.	The value specified for the DEFAULT option is of an incompatible type for the column or is incorrectly formatted.

Related reference:

- “Messages for the table-structured file wrapper” on page 152
- “Messages for the Documentum wrapper” on page 184
- “Messages for the Excel wrapper” on page 198
- “Messages for the XML wrapper” on page 251

Chapter 16. Configuring access to XML data sources

This chapter explains what XML is, how to add XML data sources to your federated system, and lists the error messages associated with the XML wrapper.

What is XML?

The Extensible Markup Language (XML) is a universal format for structured documents and data. XML files have a file extension of `xml`. Like HTML, XML uses tags (words bracketed by < and >) for structuring data in the document. A sample XML document is shown in Figure 7.

```
<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 7. Sample XML document

How the XML wrapper works

The XML wrapper enables the use of SQL to query the following types of data:

- External XML documents that are stored in a single file
- Multiple files in a directory path
- Remote XML files that are referenced with a Uniform Reference Identifier (URI)
- Relational columns

Figure 8 shows how the XML wrapper works with your federated system.

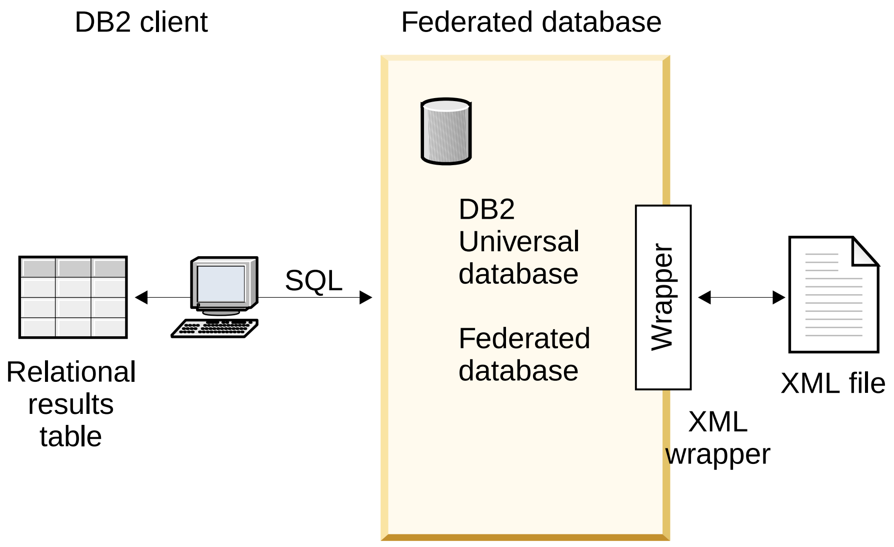


Figure 8. How the XML wrapper works

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.

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, shown in Figure 7 on page 231, is mapped into a set of nicknames, how parent and child relationships are established by using primary and foreign keys, how XPath expressions are used to define individual tuples and columns within each element of the document, and how 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 Figure 9.

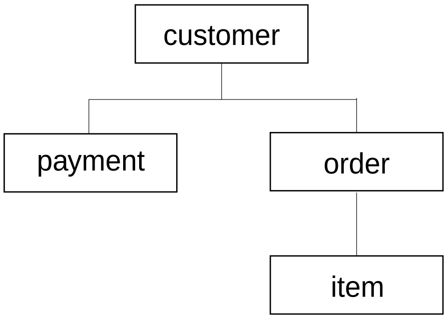


Figure 9. Tree structure of the sample XML document

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 primary and foreign key nickname column options. 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 representing individual tuples, you create one XPath expression. In this example, all the customer elements are referenced by using the `//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, and 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, with XPath expressions that define individual tuples and columns within each element of the document. You can then run SQL queries on the XML document.

Related concepts:

- "What are table-structured files?" on page 143
- "What is Documentum?" on page 157
- "What is Excel?" on page 191
- "What is BLAST?" on page 205
- "Data associations between nicknames and XML documents" on page 238

Related tasks:

- “Adding XML to a federated system” on page 235

Adding XML to a federated system

You can use an XML data source with your federated server by registering an XML wrapper. After you register an XML wrapper, you then register a corresponding server and nicknames to enable your federated server to retrieve and process XML data.

Procedure:

To add an XML data source to a federated server:

1. Register the wrapper using the CREATE WRAPPER statement.Register the wrapper by using the CREATE WRAPPER statement.
2. Optional: Set the DB2_DJ_COMM environment variable to improve query performance.
3. Register the server using the CREATE SERVER statement.Register the server by using the CREATE SERVER statement.
4. Register nicknames using the CREATE NICKNAME statement.Register nicknames by using the CREATE NICKNAME statement.
5. Create federated views for non-root nicknames

   Root nicknames identify the elements at the top level of an XML document. Nonroot nicknames identify the elements at the lower levels within that XML document.

You can run the statements from the DB2 Control Center or from a DB2 command line processor. After you add the XML wrapper to your federated system, you can run queries on an XML data source.

Related tasks:

- “Registering the XML wrapper” on page 236
- “Setting the DB2_DJ_COMM DB2 profile variable for the XML wrapper” on page 236
- “Registering the server for an XML data source” on page 237
- “Registering nicknames for XML data sources” on page 243
- “Creating federated views for nonroot nicknames (XML wrapper)” on page 249

Registering the XML wrapper

Registering the XML wrapper is part of the larger task of adding XML to a federated system. You must register the wrapper to access a data source. Wrappers are mechanisms that federated servers use to communicate with and retrieve data from the data sources. Wrappers are installed on your system as library files.

You can use the XML wrapper on the following operating systems:

- AIX
- HP-UX
- Linux
- Solaris Operating Environment
- Windows NT

Procedure:

To register the XML wrapper, submit the CREATE WRAPPER statement.

For example, to register an XML wrapper on AIX called `my_xml` from the default library file, `libdb2lsxml.a`, issue the following statement:

```
CREATE WRAPPER my_xml LIBRARY 'libdb2lsxml.a';
```

The next task in this sequence of tasks is setting the DB2_DJ_COMM environment variable for the XML wrapper.

Related tasks:

- "Setting the DB2_DJ_COMM DB2 profile variable for the XML wrapper" on page 236
- "After installing nonrelational wrappers" in the *DB2 Information Integrator Installation Guide*

Related reference:

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*

Setting the DB2_DJ_COMM DB2 profile variable for the XML wrapper

Setting the DB2_DJ_COMM DB2 profile variable for the XML wrapper is part of the larger task of adding XML to a federated system. To improve performance when you are accessing XML documents, you can optionally set the DB2_DJ_COMM DB2 profile variable. This variable determines whether the federated server loads the wrapper during initialization.

Procedure:

To set the DB2_DJ_COMM DB2 profile variable, submit the **db2set** command with the wrapper library that corresponds to the wrapper that you specified in the associated CREATE WRAPPER statement.

For example:

```
db2set DB2_DJ_COMM='libdb2lsxml.a'
```

Ensure that there are no spaces on either side of the equal sign (=).

Processor usage of your system increases when the federated server loads the wrapper libraries during database startup. To avoid excessive usage, specify only the libraries that you intend to access.

The next task in this sequence of tasks is registering the server for an XML data source.

Related concepts:

- "Environment Variables and the Profile Registry" in the *Administration Guide: Implementation*

Related tasks:

- "Registering the server for an XML data source" on page 237

Related reference:

- "db2set - DB2 Profile Registry Command" in the *Command Reference*

Registering the server for an XML data source

Registering the server for an XML data source is part of the larger task of adding XML to a federated system. After you register the wrapper, you must register a corresponding server.

Restrictions:

The XML wrapper does not use the TYPE and VERSION keywords. An error occurs if these keywords are used in the CREATE SERVER statement.

The XML wrapper does not support pass-through sessions to the federated system.

Procedure:

To register the XML server to the federated system, issue the CREATE SERVER statement.

For example:

```
CREATE SERVER xml_server WRAPPER my_xml;
```

The next task in this sequence of tasks is registering nicknames for XML data sources.

Related tasks:

- "Registering nicknames for XML data sources" on page 243

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*

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.

Note: 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 of 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.

Regardless of the type of data that you are using (data in a fixed format or data from source names that are specified at query time), you can specify the STREAMING option so that the XML wrapper separates the XML document data into fragments. 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 but the processing time required to run the entire query increases depending on the memory capacity of your server. Therefore, use the STREAMING option to parse large XML documents (documents of 50 megabytes or more) only.

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.

Related concepts:

- “What is XML?” on page 231
- “The cost model facility for the XML wrapper” on page 240
- “Optimization tips for the XML cost model facility” on page 241

Related tasks:

- “Registering nicknames for XML data sources” on page 243

Related reference:

- “CREATE NICKNAME statement syntax - XML wrapper” on page 351
- “CREATE NICKNAME statement - Examples for XML wrapper” on page 244

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 parameters as nickname option values to support the cost model facility:

- INSTANCE_PARSE_TIME
- XPATH_EVAL_TIME

You can use the default values for these parameters. Or you can set the values for these parameters to optimize queries on the root and nonroot nicknames that you create.

The INSTANCE_PARSE_TIME parameter is the amount of time (in milliseconds) that is required to read and parse one row-producing root element of the root nickname (for example, customers), including all contained row-producing nonroot elements (for example, all 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 parameter 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.

Related concepts:
- "What is XML?" on page 231
- "Data associations between nicknames and XML documents" on page 238
- "Optimization tips for the XML cost model facility" on page 241

Related reference:
- "CREATE NICKNAME statement syntax - XML wrapper" on page 351
- "CREATE NICKNAME statement - Examples for XML wrapper" on page 244

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 parameters of the CREATE NICKNAME statement:
- INSTANCE_PARSE_TIME
- XPATH_EVAL_TIME

You can specify values for these parameters when you issue a CREATE NICKNAME statement to register a nickname for an XML data source.

The cost model facility uses these parameter values when determining the amount of time required to parse data in each row of an XML source document and to evaluate the nickname's XPath expression.

You can use the default values for these parameters. 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:

Assume that your XML document has a relational schema with four nicknames:

- customers
- orders
- payments
- items

Also, assume that the customers nickname is the root nickname.

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;
```

Note the amount of time (in milliseconds) that is required to run each query by using the **db2batch** command or equivalent command or utility. (You can use the **db2batch** command to obtain an output file that contains the time required to run queries.) Also, note 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 parameters:

```
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 the example, `customers`), use the calculated values for the INSTANCE_PARSE_TIME and XPATH_EVAL_TIME parameters.

For nonroot nicknames, (in the 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 parameters also depend on the complexity of your XML source documents and on the speed of the processor that you are using.

Related concepts:

- "What is XML?" on page 231
- "Data associations between nicknames and XML documents" on page 238
- "The cost model facility for the XML wrapper" on page 240

Related reference:

- "db2batch - Benchmark Tool Command" in the *Command Reference*

Registering nicknames for XML data sources

Registering nicknames for XML data sources is part of the larger task of adding XML to a federated system. 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.

Prerequisite:

The database code page must match the character set of the XML source files.

Restriction:

Namespaces are not supported.

Procedure:

To register nicknames for XML data sources, issue a CREATE NICKNAME statement.

The next task in this sequence of tasks is creating federated views for nonroot nicknames (XML wrapper).

Related tasks:

- "Creating federated views for nonroot nicknames (XML wrapper)" on page 249

Related reference:

- “CREATE NICKNAME statement syntax - XML wrapper” on page 351
- “CREATE NICKNAME statement - Examples for XML wrapper” on page 244

CREATE NICKNAME statement - Examples for XML wrapper

This topic provides several examples that show you how to use the CREATE NICKNAME statement to register nicknames for the XML wrapper. This topic includes a complete example, which shows how to create parent and child nicknames, examples for specific column options, and examples that show the use of views.

Complete example:

The following example shows how to create nicknames for XML data sources by using the sample XML file shown in Figure 10.

```
<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 10. Sample XML file

To create the parent nickname, `customers`, 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)  OPTIONS(PRIMARY_KEY 'YES'))
   FOR SERVER xml_server
   OPTIONS(DIRECTORY_PATH '/home/db2user',
         XPATH '//customer', STREAMING 'YES');
```

This statement creates the customers nickname over multiple XML files under the specified directory path, /home/db2user. The STREAMING option indicates that the XML source data is separated and processed by node (in this example, by customer record).

You can now create nicknames for the children of the customers nickname (orders, payments, and items).

Issue the following nickname statement to create the orders 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)   OPTIONS(FOREIGN_KEY 'CUSTOMERS'))
   FOR SERVER xml_server
   OPTIONS( XPATH './/order');
```

Issue the following nickname statement to create the payments nickname.

```
CREATE NICKNAME payments
(
   number   INTEGER       OPTIONS(XPATH './number'),
   date     VARCHAR(10)   OPTIONS(XPATH './date'),
   cid      VARCHAR(16)   OPTIONS(FOREIGN_KEY 'CUSTOMERS'))
   FOR SERVER xml_server
   OPTIONS( XPATH './/payment');
```

Issue the following nickname statement to create the items nickname.

```
CREATE NICKNAME items
(
   name        VARCHAR(20)   OPTIONS(XPATH './name'),
   quantity    INTEGER       OPTIONS(XPATH './@quant'),
   oid         VARCHAR(16)   OPTIONS(FOREIGN_KEY 'ORDERS'))
   FOR SERVER xml_server
   OPTIONS( XPATH './/item');
```

Column option examples:

The column option examples show you how to create nicknames by using the DOCUMENT column options.

The following CREATE NICKNAME example shows the use of the DOCUMENT 'FILE' 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)    OPTIONS(PRIMARY_KEY 'YES'))
   FOR SERVER xml_server
   OPTIONS(XPATH '//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';
```

The following CREATE NICKNAME example shows the use of the DOCUMENT 'DIRECTORY' 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)    OPTIONS(PRIMARY_KEY 'YES'))
   FOR SERVER xml_server
   OPTIONS(XPATH '//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.

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)    OPTIONS(PRIMARY_KEY 'YES'))
   FOR SERVER xml_server
   OPTIONS(XPATH '//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';
```

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 '//employee');
```

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.

View examples:

The view examples show you how to create views for nonroot nicknames to describe XML source documents. In these examples, assume that the nicknames of the sample file shown in Figure 11 on page 248 were previously created as customers, orders, payments, and items.

```
<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 11. Sample XML file

The following example shows how to create a 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 view for the nonroot nickname payment:

```
CREATE VIEW payment_view AS
SELECT p.amount, p.date, c.cid
FROM customers c, payments p
WHERE c.cid = p.cid;
```

The following example shows how to create a view for the nonroot nickname item:

```
CREATE VIEW item_view AS
SELECT it.quantity, it.name, o.oid
FROM customers c, orders o, items i
WHERE c.cid = o.cid AND o.oid = i.oid;
```

Queries that are submitted to these views are processed correctly because the join path to the root directory is present.

For example, the following query pairs the amounts of customer's orders and payments from the same date:

```
SELECT o.amount, p.amount
FROM order_view o, payment_view p
WHERE p.date = o.date AND
   p.cid = o.cid;
```

Related tasks:

- "Registering nicknames for XML data sources" on page 243

Related reference:

- "CREATE NICKNAME statement syntax - XML wrapper" on page 351

Creating federated views for nonroot nicknames (XML wrapper)

Creating federated views for nonroot nicknames (XML wrapper) is part of the larger task of adding XML to a federated system.

You can define federated views over the hierarchy of nicknames that describe an XML document. Defining federated views ensures that the queries that join pieces of an XML nickname hierarchy (not including the root nickname and queries that join columns other than the special PRIMARY_KEY and FOREIGN_KEY columns) run properly.

Procedure:

To define federated views that include all required predicates and a full path to the root directory, follow these steps:

1. Define a view for each nonroot nickname as a join of all the nicknames on the path to the root.
2. In the WHERE clause, make the join predicates over the PRIMARY_KEY and FOREIGN_KEY columns.
3. 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.
4. In the SELECT list, include the column of the parent nickname designated with the PRIMARY_KEY option.

Related reference:

- "CREATE NICKNAME statement - Examples for XML wrapper" on page 244

XML data source - Example queries

This topic provides several sample queries that use the nicknames customers, orders, and items. These nicknames were previously registered by using CREATE NICKNAME statements.

The following query displays all customer names:

```
SELECT name FROM customers;
```

The following query displays all records in which the customer name is Chang:

```
SELECT * FROM customers where name='Chang';
```

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;
```

You must specify the join, c.cid=o.cid, to indicate the parent-child relationship between the customers nickname and the orders nickname.

The following query selects the customer addresses, order amounts, and item names for each order and item of each customer:

```
SELECT c.address, o.amount, i.name FROM customers c, orders o, items i
WHERE c.cid=o.cid AND o.oid=i.oid;
```

You must specify the two joins to maintain the parent-child relationships.

The following examples show how to write queries by using a nickname that specifies a DOCUMENT column option rather than a FILE_PATH nickname option. The corresponding CREATE NICKNAME statement that is used to create the customers nickname is shown here:

```
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 '//customer');
```

The following query selects all the data from the XML file Customers.xml with a file path of /home/db2user/Customers.xml:

```
SELECT * FROM customers WHERE doc='/home/db2user/Customers.xml';
```

The following query selects names of customers and dates of their orders from the Customers.xml file for each order with an amount over 1000:

```
SELECT c.name, o.date FROM customers c, orders o
WHERE c.doc='/home/db2user/Customers.xml' AND o.amount > 1000;
```

The file path of /home/db2user/Customers.xml specifies the location of the Customers.xml file.

Related reference:

- "CREATE NICKNAME statement syntax - XML wrapper" on page 351
- "CREATE NICKNAME statement - Examples for XML wrapper" on page 244

Messages for the XML wrapper

This topic describes messages that you might encounter when working with the wrapper for XML. For more information about messages, see the *DB2 Message Reference*.

Table 44. Messages issued by the wrapper for XML

Error Code	Message	Explanation
SQL0405N	The numeric literal ″<column_name>″ is not valid because its value is out of range.	The specified numeric literal is not within the acceptable range. Check the data type of the column in the CREATE NICKNAME statement.
SQL0408N	A value is not compatible with the data type of its assignment target. Target name is ″<column_name>.″	The data type of the value that is being assigned to the column is not compatible with the declared data type of the assignment target. Check the data type of the column in the CREATE NICKNAME statement.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Error creating wrapper object.″)	An error occurred when creating a new wrapper object. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″<xerces_xalan_error_message>.″)	An error occurred during a call to a Xerces or a Xalan function. Check the XML document. If the document is well structured, refer to the Xalan documentation for more information about the error message.

Table 44. Messages issued by the wrapper for XML (continued)

Error Code	Message	Explanation
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″XalanDOMException: exception code is <exception_code>.″)	A XalanDOMException exception occurred. Refer to the Xalan documentation for more information about the exception code.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″XMLException: <exception_error_message>.″)	An XMLException exception occurred. Refer to the Xalan documentation for more information about the exception code.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″XSLException: <exception_error_message>.″)	An XSLException exception occurred. Refer to the Xalan documentation for more information about the exception code.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″SAXParseException: <exception_error_message>.″)	A SAXParseException exception occurred. Refer to the Xalan documentation for more information about the exception code.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Error getting node value.″)	Xalan tried to access a node that is not valid. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Error parsing XML document.″)	An error occurred when parsing the XML document. Check the XML document.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Error getting root element of XML document.″)	After parsing the XML document, Xalan tried to retrieve the root element but failed. Check the XML document.

Table 44. Messages issued by the wrapper for XML (continued)

Error Code	Message	Explanation
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Unspecified exception while evaluating XPath expression.″)	Xalan generated an unspecified exception when evaluating an XPath expression. Check the XML document, and refer to the Xalan documentation.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Unspecified exception while getting node value.″)	Xalan generated an unspecified exception when retrieving a node value. Check the XML document, and refer to the Xalan documentation.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Unspecified exception while parsing input document.″)	Xalan generated an unspecified exception when parsing the XML document. Check the XML document, and refer to the Xalan documentation.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Error when evaluating cardinality.″)	Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″<SOAP_error_message>.″)	The SOAP library issued an error. If you cannot resolve the SQL statement error, contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Invalid URI.″)	The wrapper cannot access the specified URL. Verify that the URL is accessible.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Invalid XML document content.″)	The content of the XML document is not valid. Verify that the document is well structured.

Table 44. Messages issued by the wrapper for XML (continued)

Error Code	Message	Explanation
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Invalid SOAP envelope.″)	The SOAP envelope is not valid. Check its syntax and content.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Memory allocation error.″)	An error occurred when allocating memory.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″Incorrect DATE format.″	A date value in the XML document does not have the correct format. The valid format for date values is yyyy-mm-dd. Check the XML document.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″Column data type not supported.″	A nickname column has an unsupported data type. Check the CREATE NICKNAME statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″TYPE clause not supported.″	The CREATE SERVER statement contains a TYPE clause. This clause is not supported by the XML wrapper. Remove the clause.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″VERSION clause not supported.″	The CREATE SERVER statement contains a VERSION clause. This clause is not supported by the XML wrapper. Remove the clause.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″Invalid use of predicate with DOCUMENT column.″	The query contains a predicate with incorrect operands. Check the predicates in the query.

Table 44. Messages issued by the wrapper for XML (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″Invalid use of predicate with FOREIGN_KEY column.″	The query contains a predicate with incorrect operands. Check the predicates in the query.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″Invalid use of predicate with PRIMARY_KEY column.″	The query contains a predicate with incorrect operands. Check the predicates in the query.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″XPATH and DOCUMENT options not compatible.″	The CREATE NICKNAME statement is not correct as specified. Check the syntax of the statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″XPATH and FOREIGN_KEY options not compatible.″	The CREATE NICKNAME statement is not correct as specified. Check the syntax of the statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″XPATH and PRIMARY_KEY options not compatible.″	The CREATE NICKNAME statement is not correct as specified. Check the syntax of the statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″DOCUMENT and FOREIGN_KEY options not compatible.″	The CREATE NICKNAME statement is not correct as specified. Check the syntax of the statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″DOCUMENT and PRIMARY_KEY options not compatible.″	The CREATE NICKNAME statement is not correct as specified. Check the syntax of the statement.

Table 44. Messages issued by the wrapper for XML (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″FOREIGN_KEY and PRIMARY_KEY options not compatible.″	The CREATE NICKNAME statement is not correct as specified. Check the syntax of the statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″Column option missing.″	The CREATE NICKNAME statement is not correct as specified. Check the syntax of the statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″DOCUMENT column option not unique.″	The CREATE NICKNAME statement is not correct as specified. Check the syntax of the statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″FOREIGN_KEY column option not unique.″	The CREATE NICKNAME statement is not correct as specified. Check the syntax of the statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″PRIMARY_KEY column option not unique.″	The CREATE NICKNAME statement is not correct as specified. Check the syntax of the statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″Invalid DOCUMENT option value.″	The value of the DOCUMENT option that is specified in the CREATE NICKNAME statement is not valid. The value must be FILE. Check the CREATE NICKNAME statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″Invalid PRIMARY_KEY option value.″	The value of the PRIMARY_KEY option that is specified in the CREATE NICKNAME statement is not valid. The value must be YES. Check the CREATE NICKNAME statement.

Table 44. Messages issued by the wrapper for XML (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″Invalid FOREIGN_KEY option value.″	The value of the FOREIGN_KEY option that is specified in the CREATE NICKNAME statement is not valid. The value does not match any parent nickname. Check the CREATE NICKNAME statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″FILE_PATH and DOCUMENT options not compatible.″	The CREATE NICKNAME statement is not correct as specified. The FILE_PATH and DOCUMENT options cannot be specified at the same time. Check the syntax of the CREATE NICKNAME statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″FILE_PATH and SOAP options not compatible.″	The CREATE NICKNAME statement is not correct as specified. The FILE_PATH and SOAP options cannot be specified at the same time. Check the syntax of the CREATE NICKNAME statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″DIRECTORY_PATH and SOAP options not compatible.″	The CREATE NICKNAME statement is not correct as specified. The DIRECTORY_PATH and SOAP options cannot be specified at the same time. Check the syntax of the CREATE NICKNAME statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″FILE_PATH and DIRECTORY_PATH options not compatible.″	The CREATE NICKNAME statement is not correct as specified. The FILE_PATH and DIRECTORY_PATH options cannot be specified at the same time. Check the syntax of the CREATE NICKNAME statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″VALIDATE and STREAMING options not compatible.″	The CREATE NICKNAME statement is not correct as specified. The VALIDATE and STREAMING options cannot be specified at the same time. Check the syntax of the CREATE NICKNAME statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″FILE_PATH and FOREIGN_KEY options not compatible.″	The CREATE NICKNAME statement is not correct as specified. The FILE_PATH and FOREIGN_KEY options cannot be specified at the same time. Check the syntax of the CREATE NICKNAME statement.

Table 44. Messages issued by the wrapper for XML (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″DIRECTORY_PATH and FOREIGN_KEY options not compatible.″	The CREATE NICKNAME statement is not correct as specified. The DIRECTORY_PATH and FOREIGN_KEY options cannot be specified at the same time. Check the syntax of the CREATE NICKNAME statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″XPATH option value not valid with STREAMING enabled.″	The nickname XPATH expression is not valid when you enable the STREAMING feature. Check the XPATH option for values that are not valid such as /, ./, and //.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″Unable to read XML file.″	The file path that is specified in the CREATE NICKNAME statement or in the query is not valid. The specified file does not exist. Check the CREATE NICKNAME statement and the query.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″Unable to open directory.″	The directory path that is specified in the CREATE NICKNAME statement or in the query is not valid. The specified directory does not exist. Check the CREATE NICKNAME statement and the query.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″Reference to XML data missing.″	The CREATE NICKNAME statement must contain a reference to the XML data. Check the CREATE NICKNAME statement.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″DOCUMENT column option with value 'SOAP' missing.″	The CREATE NICKNAME statement is not correct as specified. Check the value of the DOCUMENT option. The value must be SOAP.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″SOAP option missing.″	The CREATE NICKNAME statement is not correct as specified. You must specify the SOAP option.

Table 44. Messages issued by the wrapper for XML (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″INSTANCE_PARSE_TIME only for root nicknames.″	The CREATE NICKNAME statement is not correct as specified. You can specify an INSTANCE_PARSE_TIME value only for root nicknames. Check the CREATE NICKNAME syntax.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″VALIDATE option only for root nicknames.″	The CREATE NICKNAME statement is not correct as specified. You can set the VALIDATE option to YES only if the specified nickname is a root nickname. Check the CREATE NICKNAME syntax.
SQL1822N	Unexpected error code ″<trace_point>″ received from data source ″XML wrapper.″ Associated text and tokens are ″STEAMING option only for root nicknames.″	The CREATE NICKNAME statement is not correct as specified. You can set the STREAMING option to YES only if the specified nickname is a root nickname. Check the CREATE NICKNAME syntax.
SQL1823N	No data type mapping exists for data type ″<data_type_name>″ from server ″<server_name>.″	The CREATE NICKNAME statement is not correct as specified. A column data type is not valid. Check the CREATE NICKNAME syntax.
SQL1881N	″<option_name>″ is not a valid ″<option_type>″ option for ″<object_name>.″	The specified option might not exist or might not be valid for this data source. Check the CREATE NICKNAME statement.
SQL1881N	″DIRECTORY_PATH″ is not a valid ″NICKNAME″ option for ″<object_name>.″	The value of the DIRECTORY_PATH option that is specified in the CREATE NICKNAME statement is not valid. The specified directory must be a root directory. Check the CREATE NICKNAME statement.
SQL1882N	The ″nickname″ option ″VALIDATE″ cannot be set to ″<option_value>″ for ″<object_name>.″	The value of the VALIDATE option that is specified in the CREATE NICKNAME statement is not valid. This value must be either YES or NO. Check the CREATE NICKNAME statement.

Table 44. Messages issued by the wrapper for XML (continued)

Error Code	Message	Explanation
SQL1882N	The ″nickname″ option ″STREAMING″ cannot be set to ″<option_value>″ for ″<object_name>.″	The value of the STREAMING option that is specified in the CREATE NICKNAME statement is not valid. This value must be either YES or NO. Check the CREATE NICKNAME statement.
SQL1883N	″<option_name>″ is a required ″<option_type>″ option for ″<object_name>.″	A required DB2 option was not specified. Check the CREATE NICKNAME statement.

Related reference:

- "Messages for the table-structured file wrapper" on page 152
- "Messages for the Documentum wrapper" on page 184
- "Messages for the Excel wrapper" on page 198
- "Messages for the BLAST wrapper" on page 228
- "CREATE NICKNAME statement syntax - XML wrapper" on page 351

Chapter 17. Configuring access to Entrez data sources

This chapter explains what Entrez is, how to add Entrez data sources to your federated system, and lists the error messages associated with the Entrez wrapper.

What is Entrez?

Entrez is a query and retrieval system developed by the National Center for Biotechnology Information (NCBI). You can use Entrez to access several linked databases hosted by the NCBI.

These databases include:

- PubMed (biomedical literature)
- Nucleotide (a sequence database also called GenBank)
- OMIM (Online Mendelian Inheritance in Man from John Hopkins University)
- Genome (complete genome assemblies)

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 into the DB2® environment. Although the Entrez interface supports many databases, the Entrez wrapper supports only PubMed and Nucleotide.

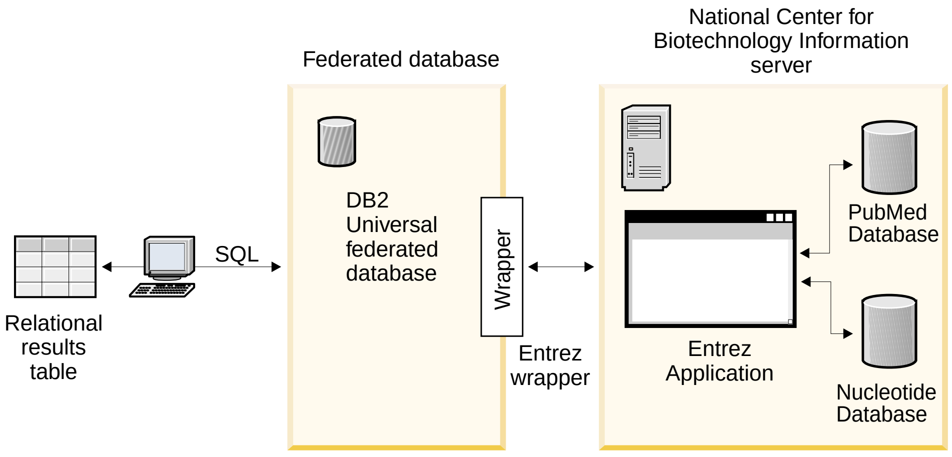


Figure 12. How the Entrez wrapper works

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

Related concepts:

- "What are table-structured files?" on page 143
- "What is Documentum?" on page 157
- "What is Excel?" on page 191
- "What is BLAST?" on page 205
- "What is XML?" on page 231

Related tasks:

- "Adding Entrez to a federated system" on page 262

Adding Entrez to a federated system

You can run statements from the DB2 Command Line Processor. After you add the Entrez wrapper to your federated system, you can run queries on an Entrez data source.

Procedure:

To add the Entrez data source to a federated server:

1. Register custom functions by issuing a CREATE FUNCTION statement.
2. Register the wrapper by issuing a CREATE WRAPPER statement.
3. Optional: Set the DB2_DJ_COMM environment variable to improve query performance.
4. Register the server by issuing a CREATE SERVER statement.
5. Register nicknames by issuing a CREATE NICKNAME statement.

Related tasks:

- "Registering custom functions for the Entrez wrapper" on page 263

Registering custom functions for the Entrez wrapper

Registering custom functions for the Entrez wrapper is part of the larger task of adding Entrez to a federated system. After the custom functions are registered, you must register the wrapper.

Restrictions:

- All of the custom functions for the Entrez wrapper must be registered with the schema name Entrez.
- You must register each custom function once for each DB2 database that has the Entrez wrapper installed.

Procedure:

To register custom functions, you must issue the CREATE FUNCTION statement with the AS TEMPLATE keyword.

The fully qualified name of each function is Entrez.<function–name>.

The following example registers one version of the CONTAINS function:

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

To assist you in registering custom functions, the sample file, `create_function_mappings.ddl`, is provided in the `samples/lifesci/entrez` directory. This file contains definitions for each custom function. You can run this DDL file to register the custom functions for each DB2 database that has the Entrez wrapper installed.

The next task in this sequence of tasks is registering the Entrez wrapper.

Related reference:

- “CREATE FUNCTION (Sourced or Template) statement” in the *SQL Reference, Volume 2*
- “Custom functions and Entrez queries” on page 267
- “Custom function table - Entrez wrapper” on page 273

Registering the Entrez wrapper

Registering the Entrez wrapper is part of the larger task of adding Entrez to a federated system. You must register the wrapper in order to access a data source. Wrappers are mechanisms that federated servers use to communicate with and retrieve data from data sources. You install wrappers on your system as library files.

Procedure:

To register the Entrez wrapper, issue a CREATE WRAPPER statement.

For example, to create an Entrez wrapper on AIX called `entrez_wrapper` from the default library file, `libdb2lsentrez.a`, submit the following statement:

```
CREATE WRAPPER entrez_wrapper LIBRARY 'libdb2lsentrez.a'
  OPTIONS(EMAIL 'jeff@someplace.com', DB2_FENCED 'N');
```

You must specify an e-mail address when you register a 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.

The next task in this sequence of tasks is setting the DB2_DJ_COMM environment variable for the Entrez wrapper.

Related tasks:

- "Setting the DB2_DJ_COMM DB2 profile variable for the Entrez wrapper" on page 264
- "After installing nonrelational wrappers" in the *DB2 Information Integrator Installation Guide*

Related reference:

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*

Setting the DB2_DJ_COMM DB2 profile variable for the Entrez wrapper

Setting the DB2_DJ_COMM DB2 profile variable for the Entrez wrapper is part of the larger task of adding Entrez to a federated system. To improve performance when Entrez data sources are accessed, set the DB2_DJ_COMM DB2 profile variable. This variable determines whether the federated server loads the wrapper upon initialization.

Procedure:

To set the DB2_DJ_COMM DB2 profile variable, issue the **db2set** command with the wrapper library that corresponds to the wrapper that you specified in the associated CREATE WRAPPER statement.

```
db2set DB2_DJ_COMM='libdb2lsentrez.a'
```

Ensure that there are no spaces on either side of the equal sign (=).

Processor usage increases when the federated server loads the wrapper libraries during database startup. To avoid excessive usage, specify only the libraries that you intend to access.

The next task in this sequence of tasks is registering the server for an Entrez data source.

Related tasks:

- "Registering the server for an Entrez data source" on page 265

Registering the server for an Entrez data source

Registering the server for an Entrez data source is part of the larger task of adding Entrez to a federated system. After you register the wrapper, you must register a corresponding server.

The database, PubMed or Nucleotide, that is represented by a particular data source is identified by the server type value, as expressed on the CREATE SERVER statement. This server type value controls the structure of any nicknames that are created.

Procedure:

To register the Entrez server to the federated system, issue a CREATE SERVER statement.

For example, to register a server named `pubmed_server1` for the `entrez_wrapper` wrapper, issue the following statement:

```
CREATE SERVER pubmed_server1
 TYPE PUBMED
   VERSION 1.0
   WRAPPER entrez_wrapper;
```

Additionally, to register a server named `nucleotid_server1` for the `entrez_wrapper` wrapper, issue the following statement:

```
CREATE SERVER nucleotid_server1
 TYPE NUCLEOTIDE
   VERSION 1.0
   WRAPPER entrez_wrapper;
```

The next task in this sequence of tasks is registering nicknames for Entrez data sources.

Related tasks:

- "Registering nicknames for Entrez data sources" on page 266

Related reference:

- "CREATE SERVER statement arguments - Entrez wrapper" on page 358

Registering nicknames for Entrez data sources

Registering nicknames for Entrez data sources is part of the larger task of adding Entrez to a federated system.

Restrictions:

The schema for each Entrez database is fixed by the wrapper and cannot be changed or amended. For each database, there is a fixed set of tables with a fixed list of columns for each table. The tables in a database have a hierarchical relationship. One table, which is the parent of all of the other tables in the database, is called the root table. All of the other tables in the database have a parent-child relationship that leads back to the root table.

Procedure:

To register nicknames for Entrez data sources, issue a CREATE NICKNAME statement.

Because the list of columns for the nicknames is fixed and is supplied by the wrapper, the basic syntax to create Nucleotide nicknames is simple. For example:

```
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;
```

Here is an example of the basic syntax to create PubMed 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;
CREATE NICKNAME PMURL FOR SERVER pubmed_server;
```

The name of the nickname is the name of the underlying table.

Use of this syntax limits you to one family of nicknames per DB2 schema. You can use other names by using the nickname options REMOTE_OBJECT and PARENT. For a root nickname, only REMOTE_OBJECT is required. For any other nickname, both REMOTE_OBJECT and PARENT must be provided.

The following example shows the same set of Nucleotide nicknames using the renaming capability:

```
CREATE NICKNAME NewSeq FOR SERVER nuc1 OPTIONS (REMOTE_OBJECT 'GBSEQ');
CREATE NICKNAME NewFeatures FOR SERVER nuc1
 OPTIONS (REMOTE_OBJECT 'GBFEATURES', PARENT 'NEWSEQ');
CREATE NICKNAME NewIntervals FOR SERVER nuc1
 OPTIONS (REMOTE_OBJECT 'GBINTERVALS', PARENT 'NEWFEATURES');
CREATE NICKNAME NewQualifiers FOR SERVER nuc1
 OPTIONS (REMOTE_OBJECT 'GBQUALIFIERS', PARENT 'NEWFEATURES');
CREATE NICKNAME NewReference FOR SERVER nuc1
 OPTIONS (REMOTE_OBJECT 'GBREFERENCE', PARENT 'NEWSEQ');
```

This example shows the same set of PubMed nicknames using the renaming capability:

```
CREATE NICKNAME newpmarticles FOR SERVER pubmed_server
 OPTIONS (REMOTE_OBJECT 'PMARTICLES');
CREATE NICKNAME NEWPMACCESSION FOR SERVER pubmed_server
 OPTIONS (REMOTE_OBJECT 'PMACCESSION', PARENT 'NEWPMARTICLES');
CREATE NICKNAME NEWPMCHEMICAL FOR SERVER pubmed_server
 OPTIONS (REMOTE_OBJECT 'PMCHEMICAL' , PARENT 'NEWPMARTICLES');
CREATE NICKNAME NEWPMMESH FOR SERVER pubmed_server
 OPTIONS (REMOTE_OBJECT 'PMMESH' , PARENT 'NEWPMARTICLES');
CREATE NICKNAME NEWPMCOMMENTS FOR SERVER pubmed_server
 OPTIONS (REMOTE_OBJECT 'PMCOMMENTS' , PARENT 'NEWPMARTICLES');
CREATE NICKNAME NEWPMARTICLEID FOR SERVER pubmed_server
 OPTIONS (REMOTE_OBJECT 'PMARTICLEID' , PARENT 'NEWPMARTICLES');
CREATE NICKNAME NEWPMURL FOR SERVER pubmed_server
 OPTIONS (REMOTE_OBJECT 'PMURL' , PARENT 'NEWPMARTICLES');
```

The next task in this sequence of tasks is to register custom functions for Entrez data sources.

Related reference:

- “CREATE NICKNAME statement” in the *SQL Reference, Volume 2*
- “PubMed and Nucleotide schema tables” on page 273

Custom functions and Entrez queries

The federated environment uses two query engines. For the Entrez wrapper, these query engines are DB2 and Entrez. With one exception, you specify all predicates for the Entrez engine through custom functions. For the DB2 engine, you specify all predicates through the relational operators.

The main custom function is ENTREZ.CONTAINS. The CONTAINS function requires a search term column argument and a query text argument. The following example shows an ENTREZ.CONTAINS statement:

```
ENTREZ.CONTAINS (<search term column>, <query text>)
```

A tag in the Q column of the schema tables identifies a search term. The query text must be in the modified Entrez query syntax. This syntax consists of search terms separated by Boolean operators (OR, AND, and NOT) and grouped by using parentheses. The syntax of the CONTAINS query text argument differs from the standard Entrez query syntax in that search term qualifiers, such as [pd], are not allowed.

The custom functions are registered in the Entrez schema, which must be used to refer to the functions. When the custom functions are used, their return value must be compared to the value 1 in an equality predicate.

In some situations, DB2 and Entrez predicates 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, 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 DB2 (the relational predicate on BaseCountA).

```
WHERE
 ENTREZ.CONTAINS (Organism, 'drosophila') = 1
 OR (BaseCountA > 10 AND ENTREZ.CONTAINS (Keywords, 'glop') = 1)
```

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. Pseudo-columns are provided for these search terms. If a pseudo-column is referenced in a select list, a value of NULL is returned.

You can run queries that might not otherwise be possible by issuing the ENTREZ.SEARCH_TERM master function. If you specify the ENTREZ.SEARCH_TERM master function, it must be the only custom function in a query. For each query, there can be only one ENTREZ.SEARCH_TERM master function per Entrez nickname. Also, SEARCH_TERM and CONTAINS functions cannot be mixed for the same nickname in the same query. The first argument, column specification, must be the primary key column for the parent nickname. The second argument, query text, 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 following example shows a query with a WHERE clause on a PubMed nickname:

```
WHERE
 ENTREZ.CONTAINS (authors, 'kaufmann OR ito AND NOT rakesh')
 AND
 (ENTREZ.CONTAINS (title, 'drosophila')
   OR
 ENTREZ.CONTAINS(alltext, 'drosophila OR "fruit fly"'))
```

In this example, the individual predicates are `authors`, `title`, and `all text`.

The individual predicates are modified so that the qualifier is added after each search term . Then, the terms are grouped with parentheses to enforce the DB2 Boolean operator precedence. Because of these modifications, the `authors` predicate becomes:

```
((kaufmann[auth] OR ito[auth]) AND (NOT (rakesh[auth])))
```

The `title` predicate becomes:

```
(drosophila[titl])
```

And the `all text` predicate becomes:

```
(drosophila[all] OR "fruit fly"[all])
```

When the individual predicates are combined, parentheses are used to maintain DB2 Boolean operator precedence. Excluding text transformations that are necessary to express the string as part of a URI, the final search term string submitted to Entrez is:

```
((kaufman[auth] OR ito[auth]) AND (NOT (rakesh[auth))) AND
((drosophila[titl]) OR (drosophila[all] OR "fruit fly"[all])
```

Related reference:

- “Custom function table - Entrez wrapper” on page 273

Relational predicates for the Entrez wrapper

The Entrez wrapper supports relational predicates, such as =, BETWEEN, LIKE, and <>, on nickname columns. However, the Entrez search engine processes only a few of these relational predicates. Relational predicates that are not processed by the Entrez search engine are processed by DB2. The Entrez search engine processes equality (=) and IN predicates on certain ID columns for each schema. These predicates allow the Entrez wrapper to bypass the search phase and execute the fetch phase directly. Examples of valid predicates are:

```
WHERE pmid = '1234567'
```

```
WHERE medlineid IN ('1234567', '9191919')
```

Columns that can be used in this kind of predicate are identified by the F column of the schema tables. The value of this option must be Y.

Related concepts:

- “Invalid WHERE clauses for the Entrez wrapper” on page 270

Related tasks:

- “Entrez data source — Example queries” on page 271
- “Registering custom functions for the Entrez wrapper” on page 263

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 an equality (or IN) predicate on the primary ID for the schema, or a custom function. Queries that do not meet these criteria are rejected with error code SQL0142N or SQL30090N.

Related concepts:

- “Relational predicates for the Entrez wrapper” on page 269

Related tasks:

- “Entrez data source — Example queries” on page 271
- “Registering custom functions for the Entrez wrapper” on page 263

Schema data element simplification

Several data elements are converted to a canonical form when they are presented through the SQL schema. These data elements include item lists, names, and dates.

Item lists

Unless otherwise noted, lists of items that are denormalized into a single column have individual items separated by a semicolon and a single space. For example, if an entry contains the keywords dnaA gene, dnaN gene, and orf187, the corresponding Keywords column will contain the value dnaA gene; dnaN gene; orf187.

Names

Names in the NCBI schemas consist of a required last name and one of several optional elements. Some of these optional elements can occur together and others are exclusive of each other. To create a canonical form of a name, assign a precedence to these elements. In order from highest to lowest, these elements are:

- Forename
- First or middle name
- Initials

You can present names with or without affiliations. Without an affiliation, a name is formatted as <last name>, <first>, where <first> is one of the optional elements. If the <first> element is not found, then the comma is not used. An affiliation can be added in the form (<affiliation>).

Separate names in denormalized lists with a semicolon and a space. An example of the correct way to separate names is:

```
Parker, M. J.; Ranjan, K. A.
```

Dates

Dates, especially publication dates, come in a wide variety of formats in the NCBI schemas. To accommodate these formats and allow for date comparisons and date arithmetic where possible, dates in the SQL schema are represented in two forms. First, a date can be a character string. Second, a date can be a column of type DATE.

If only a month is present in a date value without reference to a day, the first day of the month is the default day. If a season is present rather than a month, or a month and day, the first day of the season is used.

Entrez data source — Example queries

This topic provides some sample queries to run on Entrez data sources.

Procedure:

To run queries, use the following examples as a guide.

On PubMed nicknames:

The following shows a query with a single fetch key on a PubMed nickname:

```
select PMID, ArticleTitle FROM pmarticles WHERE pmid = '12345';
```

The following shows a query with mixed fetch keys on a PubMed nickname:

```
select PMID, ArticleTitle FROM pmarticles
 WHERE pmid = '12345' OR MedlineID = '12346';
```

The following shows a query with 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 shows a query that 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 shows a query with 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;
```

On Nucleotide nicknames:

The following shows a query with multiple fetch keys on a Nucleotide nickname:

```
select PrimaryAccession, LocusName, SeqLength  from gbseq
 WHERE PrimaryAccession in ('NM_000890', 'NC_003106');
```

The following shows a query that searches 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 shows a query that searches all of the free text on a Nucleotide nickname:

```
select * from gbseq WHERE entrez.contains(FreeText, 'abcde')=1;
```

The following shows a query that 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 shows a query that searches for a keyword on a Nucleotide nickname:

```
select PrimaryAccession, substr(KeywordList,1,200), Segment from gbseq
 WHERE entrez.contains(KeywordList, 'nkcc1 gene') = 1;
```

Related concepts:

- "Relational predicates for the Entrez wrapper" on page 269
- "Invalid WHERE clauses for the Entrez wrapper" on page 270

Related tasks:

- "Registering custom functions for the Entrez wrapper" on page 263

Custom function table - Entrez wrapper

Table 45. Custom functions for the Entrez wrapper

Function name	Description
CONTAINS (col VARCHAR(), term VARCHAR()), CONTAINS (col INTEGER, term VARCHAR()), CONTAINS (col SMALLINT, term VARCHAR()), CONTAINS (col REAL, term VARCHAR()), CONTAINS (col DOUBLE, term VARCHAR()), CONTAINS (col DATE, term VARCHAR()), CONTAINS (col TIME, term VARCHAR()), CONTAINS (col CHAR(), term VARCHAR()), CONTAINS (col TIMESTAMP(), term VARCHAR())	Searches a tagged column using the given expression. **col** Tagged column. **term** Search term.
SEARCH_TERM (col VARCHAR(), term VARCHAR())	Passes an Entrez search term directly to the Entrez search engine. **col** Tagged column. **term** Search term.

PubMed and Nucleotide schema tables

This topic provides tables for the PubMed and Nucleotide schemas.

PubMed schema

This schema defines the appearance of data from a PubMed type server. The schema consists of several related nicknames. In the following tables, the Q column is the field tag. For a list of valid searchable tags, see http://www.ncbi.nlm.nih.gov/ entrez/query/static/help/ pmhelp.html#SearchFieldDescriptionsandTags. The F column indicates whether the nickname column is a designated fetch key. Use of fetch keys might expedite processing in some cases.

Table 46. PubMed PMArticles nickname

Column name	Data type	Description	Q	F
PMID	VARCHAR(10) NOT NULL	PubMed ID PRIMARY_KEY Y	UID	Y
MedlineID	VARCHAR(10)	Medline ID	UID	Y

Table 46. PubMed PMArticles nickname (continued)

Column name	Data type	Description	Q	F
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.		
Status	VARCHAR(32) NOT NULL	Publication status as defined by NCBI. Values might include: In-Process, Completed, Out-of-scope, PubMed-not_MEDLINE		
DateCreated	DATE NOT NULL			
DateCompleted	DATE			
DateRevised	DATE			
ArticleTitle	VARCHAR(250) NOT NULL		TI	
Pagination	VARCHAR(32)			
Abstract	VARCHAR(32000)		TIAB	
Affiliation	VARCHAR(250)	Affiliation and address of first author	AD	
AuthorList	VARCHAR(3200)	List of authors; canonized	AU	
LanguageList	VARCHAR(250) NOT NULL	Semicolon-separated list	LA	
PublicationTypeList	VARCHAR(250) NOT NULL	Semicolon-separated list	PT	
VernacularTitle	VARCHAR(250)			
DateOfElectronicPublication	VARCHAR(32)	The NCBI schema specifies no structure for this column		

Table 46. PubMed PMArticles nickname (continued)

Column name	Data type	Description	Q	F
Country	VARCHAR(128)			
MedlineTA	VARCHAR(250) NOT NULL		TA	
NlmUniqueId	VARCHAR(32)	Contains MedlineCode if NlmUniqueID is not present		
GeneSymbolList	VARCHAR(250)	Semicolon-separated list; not used since 1996		
NumberOfReferences	INTEGER			
PersonalNameSubjectList	VARCHAR(250)	Canonized as semicolon-separated list of names	PS	
KeywordList	VARCHAR(3200)	Semicolon-separated list		
SpaceFlightMissionList	VARCHAR(250)	Semicolon-separated list		
InvestigatorList	VARCHAR(250)	Canonized as semicolon-separated list of names		
PublicationStatus	VARCHAR(32)			
ProviderID	VARCHAR(32)			
CitationSubsetList	VARCHAR(250)	Semicolon-separated list	SB	
AllFields	VARCHAR(1)	Pseudo-column; always returns NULL	ALL	
TextWords	VARCHAR(1)	Pseudo-column; always returns NULL	TW	
PubDate	DATE	Includes journal and book publication date + medline date	DP	

Table 46. PubMed PMArticles nickname (continued)

Column name	Data type	Description	Q	F
PubDateString	VARCHAR(32)	Includes journal and book publication date + medline date	DP	
Title	VARCHAR(250)	Book or journal title	TA	
Journal_ISSN	CHAR(9)		TA	
Journal_Volume	VARCHAR(10)		VI	
Journal_Issue	VARCHAR(10)		IP	
Journal_Coden	VARCHAR(32)			
Journal_ISOAbbreviation	VARCHAR(32)			
Book_Publisher	VARCHAR(128)			
Book_Authors	VARCHAR(250)	Canonized as other author lists		
Book_CollectionTitle	VARCHAR(128)			
Book_Volume	VARCHAR(10)			

Table 47. PubMed PMAccession nickname

Column name	Data type	Description	Q
PMID	VARCHAR(10) NOT NULL	FOREIGN_KEY PMARTICLES	
DataBankName	VARCHAR(250) NOT NULL		SI
Accession	VARCHAR(32) NOT NULL		SI

Table 48. PubMed PMChemical nickname

Column name	Data type	Description	Q
PMID	VARCHAR(10) NOT NULL	FOREIGN_KEY PMARTICLES	
NameOfSubstance	VARCHAR(128) NOT NULL		NM
RegistryNumber	VARCHAR(32) NOT NULL	Might be CAS or other registry number	RN
CASRegistry	CHAR	Y or N	

Table 49. PubMed PMMeSHHeading nickname

Column name	Data type	Description	Q
PMID	VARCHAR(10) NOT NULL	FOREIGN_KEY PMARTICLES	ID
DescriptorOrName	VARCHAR(128) NOT NULL		MH (If the predicate ″DescriptorIsMajor = Y″ is included in the query, then the search term is MAJR.)
DescriptorIsMajor	CHAR NOT NULL	Y if descriptor is major	
QualifierOrSubhead	VARCHAR(128)		SH
QSIsMajor	CHAR	Y if qualifier or subhead is major	

Table 50. PubMed PMComments nickname

Column name	Data type	Description	Q
PMID	VARCHAR(10) NOT NULL	FOREIGN_KEY PMARTICLES	
RefSource	VARCHAR(128) NOT NULL		
Type	VARCHAR(32) NOT NULL	CommentOn, CommentIn, ErratumIn, ErratumFor, RepublishedFrom, RepublishedIn, RetractionOf, RetractionIn, UpdateIn, UpdateOf, SummaryForPatents, OriginalReportIn	
Note	VARCHAR(3200)		

Table 51. PubMed PMArticleID nickname

Column name	Data type	Description	Q
PMID	VARCHAR(10) NOT NULL	FOREIGN_KEY PMARTICLES	

Table 51. PubMed PMArticleID nickname (continued)

Column name	Data type	Description	Q
ArticleID	VARCHAR(32) NOT NULL		
IdType	VARCHAR(8) NOT NULL	doi, pii, pmcpid, pmpid, sici, pubmed, medline, pmcid	

Table 52. PubMed PMURL nickname

Column name	Data type	Description	Q
PMID	VARCHAR(10) NOT NULL	FOREIGN_KEY PMARTICLES	
URL	VARCHAR(250) NOT NULL		
Language	CHAR(2)	ISO Language code	
Type	CHAR(1)	F for FullText, S for Summary	

Nucleotide schema

See http://www.ncbi.nlm.nih.gov/entrez/query/static/help/Summary_Matrices.html#Search_Fields_and_Qualifiers.

Table 53. Nucleotide GBSeq nickname

Column name	Data type	Description	Q	F
PrimaryAccession	VARCHAR(16) NOT NULL	Primary accession number	PACC	Y
SequenceKey	VARCHAR(32) NOT NULL	PRIMARY_KEY Y		
LocusName	VARCHAR(16) NOT NULL		ACCN	
SeqLength	INTEGER NOT NULL		SLEN	
Strandedness	VARCHAR(32)	not-set, single-stranded, double-stranded, mixed-stranded		

Table 53. Nucleotide GBSeq nickname (continued)

Column name	Data type	Description	Q	F
MoleculeType	VARCHAR(16)	nucleic-acid, dna, rna, trna, rrna, mrna, urna, snrna, snorna, peptide	PROP	
Topology	VARCHAR(16)	linear, circular		
Division	CHAR(3) NOT NULL		PROP	
UpdateDate	DATE NOT NULL		MDAT	
CreateDate	DATE NOT NULL			
Definition	VARCHAR(7000) NOT NULL		TITL	
Version	INTEGER			
GI	VARCHAR(16)	FETCH_KEY Y	UID	
KeywordList	VARCHAR(7000)	Semicolon separated list	KYWD	
Segment	VARCHAR(250)			
Source	VARCHAR(200) NOT NULL		ORGN	
Organism	VARCHAR(7000) NOT NULL		ORGN	
Taxonomy	VARCHAR(7000) NOT NULL			
Comment	VARCHAR(7000)			
Primary	VARCHAR(7000)			
SourceDB	VARCHAR(250)			
Sequence	CLOB			
AllText	VARCHAR(1)	Pseudo-column, always returns NULL	ALL	

Table 53. Nucleotide GBSeq nickname (continued)

Column name	Data type	Description	Q	F
FreeText	VARCHAR(1)	Pseudo-column, always returns NULL	WORD	

Table 54. GBReference nickname

Column name	Data type	Description	Q
SequenceKey	VARCHAR(32) NOT NULL	FOREIGN_KEY Y	
ReferenceNum	INTEGER NOT NULL	Parsed from GBReference_reference	
RangeLow	INTEGER NOT NULL	Low base for reference (parsed from GBReference_reference)	
RangeHigh	INTEGER NOT NULL	High base for reference (parsed from GBReference_reference)	
Authors	VARCHAR(3200)	Semicolon-separated list of names in GenBank form	AUTH
Consortium	VARCHAR(250)		
Title	VARCHAR(250)		WORD
Journal_Title	VARCHAR(250) NOT NULL		JOUR
MedlineID	INTEGER		
PubMedID	INTEGER		
Remarks	VARCHAR(3200)		

Table 55. Nucleotide GBFeatures nickname

Column name	Data type	Description	Q
SequenceKey	VARCHAR(32) NOT NULL	FOREIGN_KEY GBSEQ	
FeatureJoinKey	VARCHAR(32) NOT NULL	PRIMARY_KEY Y	

Table 55. Nucleotide GBFeatures nickname (continued)

Column name	Data type	Description	Q
FeatureKey	VARCHAR(20) NOT NULL		FKEY
FeatureLocation	VARCHAR(200) NOT NULL		

Table 56. Nucleotide GBIntervals nickname

Column Name	Data type	Description	Q
FeatureJoinKey	VARCHAR(32) NOT NULL	FOREIGN_KEY GBFEATURES	
IntervalFrom	INTEGER		
IntervalTo	INTEGER		
IntervalPoint	INTEGER		
IntervalAccession	VARCHAR(32) NOT NULL		

Table 57. Nucleotide GBQualifiers nickname

Column name	Data type	Description	Q
FeatureJoinKey	VARCHAR(32) NOT NULL	FOREIGN_KEY GBFEATURES	
QualifierName	VARCHAR(50)		
QualifierValue	VARCHAR(32000)		

Messages for the Entrez wrapper

This topic describes messages that you might encounter when working with the wrapper for Entrez. For messages that are not documented in this table, the *Message Reference: Volume 1*, or the *Message Reference: Volume 2*, contact IBM Software support.

Table 58. Messages issued by the wrapper for Entrez

Error Code	Message	Explanation
SQL0142N	The SQL statement is not supported.	An invalid query type was passed to the wrapper. Check to see if the issued SQL statement is supported by this wrapper.

Table 58. Messages issued by the wrapper for Entrez (continued)

SQL0204N	″<name>″ is an undefined name.	The specified name is not valid. Check the CREATE NICKNAME statement.
SQL0405N	The numeric literal ″<literal>″ is not valid because its value is out of range.	A column in the retrieved XML data or a predicate in an SQL statement contains a value that is out of the possible range for that data type. Check the data type for this column and the column in the data source, or redefine the column to a more appropriate type.
SQL0408N	A value is not compatible with the data type of its assignment target. Target name is ″<target_name>″.	A column in the XML data contains characters that are not valid for that data type. Check the data type for this column and the column in the data source, or redefine the column to a more appropriate type.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Cannot find database prototype.″)	This is an internal error. Contact IBM Software support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″No data to unpack.″)	This is an internal error. Contact IBM Software support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Error creating wrapper object.″)	This is an internal error. Contact IBM Software support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Bad expression type.″)	This is an internal error. Contact IBM Software support.

Table 58. Messages issued by the wrapper for Entrez (continued)

SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Cannot find nickname.″)	This is an internal error. Contact IBM Software support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Memory allocation error.″)	There is not sufficient memory to process the allocation request inside of the wrapper.
SQL1816N	Wrapper ″<wrapper_name>″ cannot be used to access the ″version″ of data source (″<server_type>″, ″<server_version>″) that you are trying to define to the federated server.	A value in the VERSION clause of the CREATE SERVER statement is not valid.
SQL1816N	Wrapper ″<wrapper_name>″ cannot be used to access the ″type″ of data source (″<server_type>″, ″<server_version>″) that you are trying to define to the federated server.	A value in the TYPE clause of the CREATE SERVER statement is not valid.
SQL1817N	The CREATE SERVER statement does not identify the ″type″ of the data source that you want to define to the federated database.	The TYPE clause of the CREATE SERVER statement is required but was not specified.
SQL1822N	Unexpected error code ″900″ received from data source ″Entrez Wrapper.″ Associated text and tokens are ″Parent nickname not defined.″	This is an internal error. Contact IBM Software support.
SQL1823N	No data type mapping exists for data type ″<data_type>″ from server ″<server_name>.″	This is an internal error. Contact IBM Software support.

Table 58. Messages issued by the wrapper for Entrez (continued)

SQL1881N	″<option_name>″ is not a valid ″<option_type>″ for ″<option_name>.″	The specified option is not a valid option. Check the CREATE NICKNAME statement.
SQL1882N	The ″<option_type>″ option ″<option_name>″ cannot be set to ″<option_value>″ for ″<option_name>.″	The specified value is not valid for this option. Check the CREATE NICKNAME statement.
SQL1883N	″<option_name>″ is a required ″<option_type>″ option for ″<option_name>.″	The specified option is required for the object but was not specified. Check the CREATE NICKNAME statement.
SQL1884N	You specified ″FOREIGN_KEY″ (a ″COLUMN″ option) more than once.	This is an internal error. Contact IBM Software support.
SQL1884N	You specified ″PRIMARY_KEY″ (a ″COLUMN″ option) more than once.	This is an internal error. Contact IBM Software support.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Cannot change server version″.	The version of a server cannot be changed by issuing the ALTER SERVER statement. A new server must be created with the new version.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Invalid PARENT nickname″.	The referenced nickname in a PARENT nickname option is not valid for the current nickname.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Invalid column name″.	A specified column name in the CREATE NICKNAME statement does not match any of the possible columns for the nickname.

Table 58. Messages issued by the wrapper for Entrez (continued)

SQL30090N	Operation invalid for application execution environment. Reason code = ″Cannot AND fetch keys″.	Multiple references to a fetch key, such as the PMID column of the PMArticles nickname, were made in a conjunction. For example, ″PMID = 12346 AND PMID = 12348″. Fetch key predicates can be associated only using OR.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Mixed SEARCH_TERM and CONTAINS functions″.	The SEARCH_TERM and CONTAINS functions cannot be mixed in a query. Only one SEARCH_TERM function is allowed per query.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Invalid first argument in function″.	The first argument to a SEARCH_TERM or CONTAINS function was not valid. This argument must be a reference to a column.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Invalid second argument in function″.	The second argument to a SEARCH_TERM or CONTAINS function was not valid. This argument must be a string literal, a host variable, or a column reference.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Untagged column in CONTAINS function″.	The first argument to the CONTAINS function was not valid. This argument must be a reference to a tagged column.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Invalid function″.	This is an internal error. Contact IBM Software support.

Chapter 18. Configuring access to Extended Search data sources

This chapter explains what IBM Lotus Extended Search is, how to add Extended Search data sources to your federated system, and how to use SQL to search Extended Search data sources. It also lists error messages associated with the Extended Search wrapper.

What is Extended Search?

The Extended Search product is a multi-tiered client/server system that provides extensive search and retrieval capabilities. With Extended Search, you can enter a single request and search potentially thousands of data repositories and the Internet at the same time. These repositories, which can be of varied content and structure, might be geographically dispersed throughout the world.

Extended Search supports distributed, heterogeneous searching of structured and unstructured data through a single point of access. It leverages your current data management investment and completely handles the logistics required to access many diverse sources simultaneously.

Extended Search uses its generalized query language (GQL) as a common search syntax and internally translates each search request into the native search languages of the data sources that you want to search. It also uses methods that are native to those sources to find and retrieve information without regard for where a source is located.

See the Extended Search product documentation for information about installing an Extended Search server, configuring the search domain, and using GQL. The following documents are available on the Resources page of the IBM® Lotus® Extended Search Web site:

`http://www.lotus.com/products/des.nsf/wdocuments/resources`

Extended Search General Information
: Describes the components in an Extended Search system and how they interact with each other and the backend data systems.

Extended Search Installation
: Defines the system prerequisites and provides instructions for installing the product and verifying the installation process.

Extended Search Administration
Provides instructions for adding data sources to the search domain, configuring searchable fields, and using sample search applications to query Extended Search sources.

Extended Search Programming
Discusses the application development tools that you can use to extend search support to data sources that are not supported in the default configuration of the product. Includes a description of the Extended Search generalized query language.

Extended Search data sources

With Extended Search, you can search the following types of data sources:

- Many popular Web search sites and news sites. If you need to search your intranet's search site, or other internal or external search sites, you can easily add support for doing so.
- Mail systems, such as those that you manage with Lotus Notes® and Microsoft® Exchange Server.
- Document management systems, such as DB2® Information Integrator for Content databases.
- Relational databases, such as IBM DB2, Oracle, Microsoft SQL Server, Microsoft Access, and other databases that comply with Open Database Connectivity (ODBC) standards.
- Full text indexes, such as those that you create with IBM WebSphere® Portal, Domino™ Domain Index, Microsoft Index Server, and Microsoft Site Server.
- Lotus repositories, including Notes databases, Domino.Doc libraries and cabinets, Lotus QuickPlace™ places, and Lotus Discovery Server knowledge maps (K-maps).
- Instant messaging systems, such as Lotus Sametime. This feature enables you to direct queries to knowledgeable persons, not just searchable data repositories.
- Lightweight Directory Access Protocol (LDAP) directories, such as those that you manage with IBM SecureWay, Domino LDAP Server, and Exchange LDAP Server.
- File systems. You can search text files that are stored locally or on network drives. You cannot search compressed or encrypted files.

With the Extended Search C++ and Java™ application programming interfaces (APIs), you can extend support to other types of sources, such as proprietary databases that are not mentioned here.

How the Extended Search wrapper works

In a structured relational database model, columns are named and represented in a consistent format. This feature allows you to perform precise computational operations and join data from different tables by comparing specific column values. You can also do other types of analysis, such as listing objects in one table that are missing from another table.

In contrast, unstructured data is often stored in a free text form. Typically, there is little or no metadata that enables you to query for information by column name. A search of unstructured data depends more on finding data that matches user-specified keywords than on computational criteria.

The Extended Search wrapper combines these two search techniques. With the wrapper, you can use structured query language to search unstructured content in an Extended Search domain. You can then perform analytical or relational operations on the search results.

You issue queries by entering SQL statements that refer to a special purpose DB2 table (a nickname table). Extended Search performs the search according to the SQL criteria and populates the nickname table with the result data. Because the search results persist in a table, the data is available for operations with other database tables, including other nickname tables.

When you submit a search request with the wrapper, you can retrieve data from any Extended Search source that is mapped to a nickname table. You can integrate this data with other data sources in your federated system without moving the data out of the native data source. Search results appear as a single result set regardless of how many sources provide responses to the query.

The following figure shows how the Extended Search wrapper connects the diverse data sources in an Extended Search domain to a federated database system. The wrapper accesses and retrieves data from one or more remote Extended Search servers. If the wrapper contacts an Extended Search server that is connected to other Extended Search servers, search results can be returned from multiple servers.

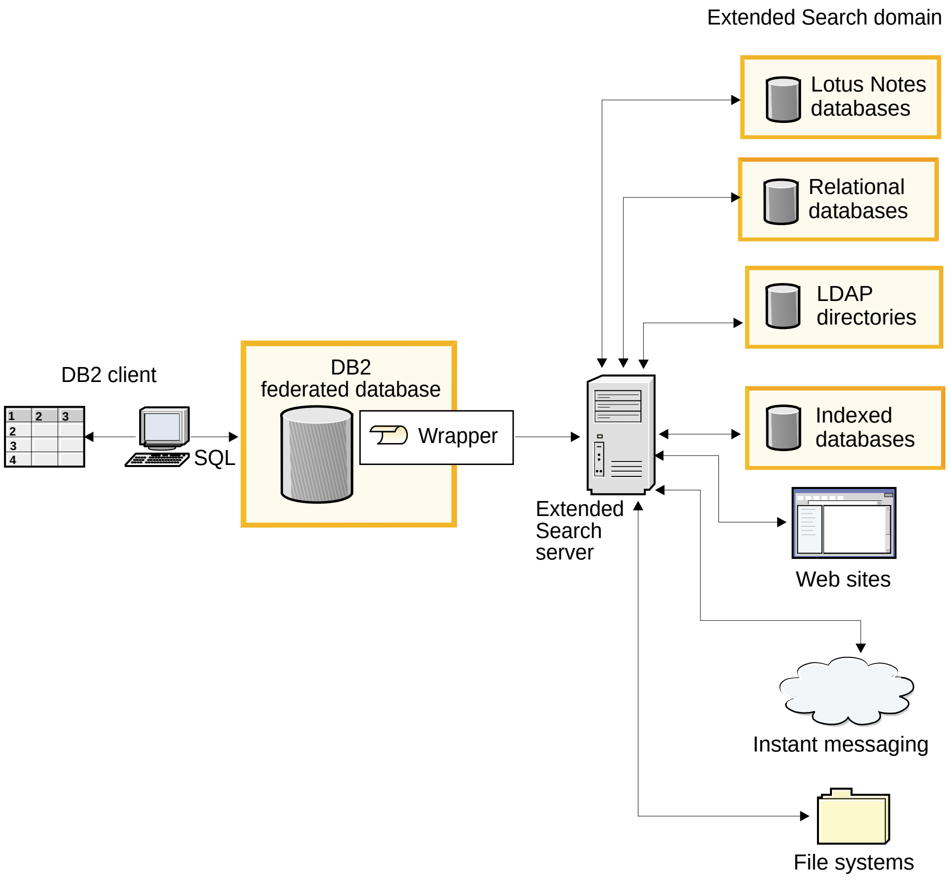


Figure 13. How the Extended Search wrapper works

Related tasks:

- "Adding Extended Search data sources to a federated server" on page 295

Extended Search nicknames

In the Extended Search data model, one or more fields constitute a document. A collection of documents constitutes a data source. You can combine any number of data sources into a category, which enables you to search them and administer them as a group.

To ensure that users access only the data sources for which they have a need, a category must belong to at least one application. Think of applications as a way of grouping users for purposes of controlling access and search capabilities. For example, a personnel application might include the same data sources as a financial application, but the users of each application would not necessarily need access to the same fields in those data sources.

When you register nicknames, you identify the applications, categories, data sources, and data source fields that you want to search. These entities must exist in the Extended Search configuration database. To search an Extended Search data source with the Extended Search wrapper, you must create a nickname for the source.

The contents of the nickname table reflect the state of the Extended Search configuration database at the time that you register the nickname. If an Extended Search administrator updates the configuration (for example, by adding or deleting sources or fields), those changes are not reflected in the nickname table. If a nickname table refers to changed data, and you want to stay current with the Extended Search configuration database, you must alter the nickname or drop it and create a new nickname.

If you do not alter or recreate the nickname, you might receive errors and reports of zero results when you attempt to search items that no longer exist in the Extended Search domain.

Although a single nickname table can contain information about all the sources that are configured in Extended Search, creating several nickname tables might be more useful. To use the full power of DB2, create a separate nickname for each type of data source that you plan to search with the Extended Search wrapper.

For example, you might have one nickname for Web sources, one for Notes databases, one for file systems, and so on. By having separate nickname tables, you are better able to perform joins on the data that is returned to the wrapper, relate diverse sources based on field values, and integrate the result data with other data in your federated system.

Related concepts:

- "Extended Search vertical tables" on page 292

Related tasks:

- "Registering nicknames for Extended Search data sources" on page 297

Related reference:

- "Extended Search wrapper - Example queries" on page 303
- "CREATE NICKNAME statement syntax - Extended Search wrapper" on page 343

Extended Search vertical tables

An Extended Search application can consist of many categories which, in turn, can contain many data sources. Because each data source uses its own conventions for field names, an intersection of fields might result in an empty set. When you map data source fields to user-defined columns in nickname tables, and present search results as a horizontal table, the table might contain an unmanageable number of columns. If many rows contain only a few columns with data, the table will appear sparsely populated. For example:

Column_1	**Column_2**	**Column_3**
Value_11		
	Value_22	
Value_31		Value_33

Within Extended Search, you can control the presentation of results by defining mapped fields. Mapped fields provide a way for you to combine content that has a common purpose but that is named differently in different sources. For example, you might create a mapped field named EmployeeNumber to represent result data from fields that are named EmpNum, EmpNo, and EmpID in various sources. Without this mapping feature, you would need to define a nickname column for each unique field name as opposed to a single column for the mapped field.

Mapping fields is useful when you know the names of the fields that you need to relate. Some applications, however, need to relate a large number of fields from many data sources. The relationships between the fields, particularly for unstructured data, might not be known ahead of time. Thus, it becomes difficult to define and structure meaningful nickname tables. To support this type of application, the Extended Search wrapper allows you to create a vertical nickname table.

When you create a nickname table for Extended Search, you can enable the VERTICAL_TABLE option. This option returns all the fields that are configured to be returnable in a data source, as defined in the Extended Search configuration database. Use this option when you are not sure which columns will be relevant in your search or which columns will be relevant when you perform post-processing queries or joins on the result sets.

Each row in the vertical table contains information about a field that was returned in the result set. For each row, Extended Search returns the name of the source that the field came from, the field name, its value, and its data type (date, integer, and so on). Unlike results that are scattered across columns in a horizontal table, the vertical table is densely populated and contains many rows of data. For example:

Field_Name	Field_Value	Field_Datatype
Column_1	Value_11	VARCHAR
Column_2	Value_22	DATE
Column_1	Value_31	VARCHAR
Column_3	Value_33	VARCHAR

You can perform SQL operations on this data when you query the table, and you can query all column labels. For example:

```
Field_Value LIKE '%IBM%'
```

Because the VERTICAL_TABLE option returns information about all returnable fields in a data source, you might not need to query specific user-defined columns. If you enable this option and then issue a SELECT statement to search user-defined columns, you might receive duplicate information in the search results. However, if you define user-defined columns, you can use those columns in joins with other tables in your federated system.

The following table summarizes the system-provided columns that Extended Search returns for each row in a vertical nickname table.

Column Name	Data Type	Description
The wrapper always returns the following three fixed columns for each nickname.		
DOC_ID	VARCHAR(512)	The document identifier, unique to each item in a set of search results.
DOC_RANK	INTEGER	The relevance ranking of the document.
CLIENT_LOCALE	VARCHAR(5)	The client locale of the search request. If the SQL query does not provide the client locale, the query will use enUS as the default client locale.
The wrapper creates the following fixed columns only if the VERTICAL_TABLE option is enabled.		
DATASOURCE_NAME	VARCHAR(128)	The name of the data source that produced the search result.
FIELD_NAME	VARCHAR(128)	The name of a field that was returned in the search result.

Column Name	Data Type	Description
FIELD_VALUE	VARCHAR(4096)	The value of a field that was returned in a result set. If the field value is longer than the maximum length of the nickname column (the VARCHAR value), the field value is truncated. The token ES_TRUNCATE at the end of the column indicates that the value is incomplete.
FIELD_DATATYPE	SMALLINT	An integer value that represents the actual data type of the field value: 384 DATE 448 VARCHAR 484 DECIMAL 496 INTEGER

A vertical table, which stores result data as VARCHAR values, can be difficult to query. For more precise searching, create mapped fields in the Extended Search configuration database and then define them in the nickname table. With mapped fields, you can create a concise horizontal table of search results. You also optimize your ability to perform relational operations on the results and combine them in queries that involve other tables in your federated database system.

For information about defining mapped fields in Extended Search, see *Extended Search Administration*, which is available on the Resources page of the IBM® Lotus® Extended Search Web site:

`http://www.lotus.com/products/des.nsf/wdocuments/resources`

Related concepts:

- "Extended Search nicknames" on page 290

Related tasks:

- "Registering nicknames for Extended Search data sources" on page 297

Related reference:

- "Extended Search wrapper - Example queries" on page 303
- "CREATE NICKNAME statement syntax - Extended Search wrapper" on page 343

Adding Extended Search data sources to a federated server

You can install the Extended Search wrapper on Microsoft Windows NT, Microsoft Windows 2000, and IBM AIX operating systems. You can use the wrapper to search Extended Search servers that exist on Windows, AIX, Sun Solaris, and Red Hat Linux for Intel operating systems.

Prerequisites:

Before you use the Extended Search wrapper, ensure that the sources that you plan to search are configured in the Extended Search configuration database. Submit a few queries through the Extended Search client to verify your ability to search the sources before you attempt to search them with the Extended Search wrapper.

Procedure:

To add Extended Search data sources to a federated system:

1. Register the Extended Search wrapper.
2. Register the Extended Search server that you want to use with the wrapper.
3. Register nicknames for the Extended Search sources that you want to search.
4. Register user mappings for DB2 users who need to search Extended Search sources.
5. Register the Extended Search custom function for specifying search options.

Related tasks:

- "Registering the Extended Search wrapper" on page 295
- "Registering the server for Extended Search data sources" on page 296
- "Registering nicknames for Extended Search data sources" on page 297
- "Registering user mappings for Extended Search data sources" on page 298
- "Registering the Extended Search custom function" on page 299

Registering the Extended Search wrapper

This task is part of the main task for adding Extended Search data sources to a federated system. To search Extended Search data sources, you must first register the Extended Search wrapper.

Procedure:

To register an Extended Search wrapper, issue a CREATE WRAPPER statement from the DB2 Command Line Processor.

For example, to register a wrapper named NotesDBwrapper on a Windows system, issue the following statement:

```
CREATE WRAPPER NotesDBwrapper LIBRARY 'db2uies.dll'
```

To register a wrapper named myESwrapper on an AIX system, issue the following statement:

```
CREATE WRAPPER myESwrapper LIBRARY 'libdb2uies.a'
```

The next task in this sequence of tasks is registering the server for Extended Search data sources.

Related tasks:

- "Registering the server for Extended Search data sources" on page 296

Related reference:

- "CREATE WRAPPER statement syntax - Extended Search wrapper" on page 362

Registering the server for Extended Search data sources

This task is part of the main task for adding Extended Search data sources to a federated system. After you register a wrapper, you must create a corresponding server definition to identify the remote Extended Search server that you are integrating with your federated system. This definition enables the wrapper to connect to the Extended Search server.

Procedure:

To register the Extended Search server, issue a CREATE SERVER statement from the DB2 Command Line Processor.

For example, to register a server named es1 for a wrapper named myESwrapper, issue the following statement. The Extended Search server uses the default port value.

```
CREATE SERVER es1 WRAPPER myESwrapper OPTIONS (ES_HOST 'my.server.com')
```

To create this same server, enable tracing for all message levels (critical, noncritical, warning, and information), and write the trace messages to a file named es1wrapper.log in the wrapper directory, issue the following statement:

```
CREATE SERVER es1 WRAPPER myESwrapper OPTIONS (ES_HOST 'my.server.com',
  ES_TRACING 'ON', ES_TRACELEVEL 'CNWI',
  ESTRACEFILENAME '/wrapper/es1wrapper.log')
```

The next task in this sequence of tasks is registering nicknames for Extended Search data sources.

Related tasks:

- "Registering nicknames for Extended Search data sources" on page 297

Related reference:

- "CREATE SERVER statement syntax - Extended Search wrapper" on page 359

Registering nicknames for Extended Search data sources

This task is part of the main task for adding Extended Search data sources to a federated system. After you register a server, you must register at least one nickname. A nickname table is a virtual DB2 table that identifies one or more searchable sources in an Extended Search domain. When you submit a query, you specify the nickname for the sources that you want to search.

Prerequisites:

Make sure that the Extended Search server for which you are creating nicknames is running. When you create a nickname, the system verifies that information about the sources and fields that you plan to search exists in the Extended Search configuration database.

Procedure:

To register an Extended Search nickname, issue a CREATE NICKNAME statement from the DB2 Command Line Processor.

For example, issue the following statement to create a nickname table for searching all data sources that belong to the Web category in the Demo application that is hosted by the es1 Extended Search server. Return the WebTitle and WebDescription fields and use the default search processing options.

```
CREATE NICKNAME allweb (WebTitle VARCHAR(255), WebDescription VARCHAR(1000))
  FOR SERVER es1 OPTIONS(APPLICATIONID 'Demo', CATEGORY 'Web')
```

Issue the following statement to create a nickname table for searching several data sources in the Science application. Present the search results as a vertical list of column names, set the timeout value to 60 seconds, allow each source to return up to 100 result documents, expand the size of the result set to 1000 entries, and sort the results by author name.

```
CREATE NICKNAME stars (Title VARCHAR(80), Author VARCHAR(40),
        Abstract VARCHAR(200))
  FOR SERVER es1 OPTIONS (APPLICATIONID 'Science',
  DATASOURCES 'Astronomy;NASA Library;Astrophysics', VERTICAL_TABLE 'yes',
  TIMEOUT '60', MAXHITS '100', TOTALMAXHITS '1000', SORTFIELD 'Author')
```

The next task in this sequence of tasks is registering user mappings for the Extended Search wrapper.

Related concepts:

- “Extended Search nicknames” on page 290
- “Extended Search vertical tables” on page 292

Related tasks:

- “Registering user mappings for Extended Search data sources” on page 298

Related reference:

- “Extended Search wrapper - Example queries” on page 303
- “CREATE NICKNAME statement syntax - Extended Search wrapper” on page 343

Registering user mappings for Extended Search data sources

This task is an optional step in the main task for adding Extended Search data sources to a federated system.

User mappings provide a way to authenticate the access of users who query an Extended Search source with the Extended Search wrapper. If a user submits an SQL query to a registered Extended Search nickname, and no user mappings are defined for that user, the Extended Search wrapper will use a default user ID and password in an attempt to retrieve data from the remote Extended Search server. If a data source that is being queried requires authentication, an empty result set might be returned.

To ensure that the correct user ID and password get passed to the Extended Search server, create user mappings in your federated system for users who are authorized to search Extended Search sources. When you create a user mapping, the password is stored in an encrypted format in a DB2 catalog table. The password remains in a secure format as it is passed from DB2 through Extended Search to the sources that are being searched.

Security settings in the Extended Search configuration database determine whether the user ID and password are authorized to access the sources that are being searched and whether any additional mapping of the user ID will be performed.

Procedure:

To register Extended Search user mappings, issue a CREATE USER MAPPING statement from the DB2 Command Line Processor.

The statement must identify the DB2 user ID that needs to be mapped, the Extended Search server that hosts the target data sources, and the user ID and password that enable the user to access those data sources.

For example, the following statement registers the user1 user ID so that it can use the es1 Extended Search server to search remote databases.

```
CREATE USER MAPPING FOR user1 SERVER es1 OPTIONS
  (REMOTE_AUTHID 'ESUserId', REMOTE_PASSWORD 'abc123def')
```

The next task in this sequence of tasks is registering the Extended Search custom function template.

Related tasks:

- “Registering the Extended Search custom function” on page 299

Related reference:

- “CREATE USER MAPPING statement syntax - Extended Search wrapper” on page 361

Registering the Extended Search custom function

This task is an optional step in the main task for adding Extended Search data sources to a federated system.

Custom functions contain no executable code. After you register a function, you can refer to it in queries to alter default search behavior. The custom function for the Extended Search wrapper, ES_SEARCH, enables you to specify precise search expressions and search content that is not defined as a column in the nickname table.

Restrictions:

- You can call the ES_SEARCH function only with a WHERE clause.

- The WHERE clause must contain at least one predicate that serves as a search predicate, either the ES_SEARCH function or a predicate of type ″column-name operator constant.″
- The ES_SEARCH function is a scalar function template. It must use the EQUAL (=) operator and the comparison value must be one (1).
- The first parameter in the ES_SEARCH function serves as an anchor value for identifying the nickname to which the function should be applied, such as the document's rank (DOC_RANK) in the search results. You must specify an INTEGER field for this parameter. This parameter, which does not get evaluated, is particularly important if the SQL query contains more than one nickname or a combination of nicknames and tables. For example:

```
SELECT * FROM ES_N1, ES_N2
WHERE ESWRAPPER.ES_SEARCH(ES_N1.DOC_RANK, '"IBM"') = 1 AND
      ESWRAPPER.ES_SEARCH(ES_N2.DOC_RANK, '"IBM"') = 1
```

Procedure:

To register the Extended Search custom function, issue the following CREATE FUNCTION statement from the DB2 Command Line Processor:

```
CREATE FUNCTION ESWRAPPER.ES_SEARCH(INTEGER, VARCHAR(1024))
  RETURNS INTEGER AS TEMPLATE
```

Related reference:

- "Extended Search wrapper - Example queries" on page 303
- "Extended Search wrapper - Generalized query language" on page 305
- "CREATE FUNCTION statement syntax - Extended Search wrapper" on page 335
- "Extended Search wrapper - Query guidelines" on page 300

Extended Search wrapper - Query guidelines

The Extended Search wrapper expects queries to be in a specific format and does not support queries that do not meet precise language criteria. This topic provides guidelines for creating queries and gives examples of correct and incorrect query syntax.

Querying Web sources in multiple languages

The third-party software that Extended Search uses to link to Web sources supports languages that use the ISO–8859–1 code page (such as English, French, German, Portuguese, and Swedish). Therefore, when you search Web sources, you cannot search double-byte character set languages such as

Korean, bi-directional languages such as Hebrew, or other non-ISO–8859–1 languages. The parser that processes search results fails when it detects what it regards as illegal character codes.

Specifying the CLIENT_LOCALE value

If you include the CLIENT_LOCALE column in a WHERE clause to set the value of the client locale, you must use an AND predicate to specify the search criteria. You cannot use an OR predicate with the CLIENT_LOCALE column.

Examples — correct syntax

The following examples show the correct way to include the CLIENT_LOCALE column in a WHERE clause:

```
WHERE CLIENT_LOCALE = 'enUS' AND
ESWRAPPER.ES_SEARCH(DOC_RANK, '"IBM"')=1

WHERE ESWRAPPER.ES_SEARCH(DOC_RANK, '"IBM"')=1
AND CLIENT_LOCALE = 'enUS'
```

Examples — incorrect syntax

The following examples are incorrect because they attempt to use an OR predicate with the CLIENT_LOCALE column:

```
WHERE CLIENT_LOCALE = 'enUS' OR
ESWRAPPER.ES_SEARCH(DOC_RANK, '"IBM"')=1

WHERE ESWRAPPER.ES_SEARCH(DOC_RANK, '"IBM"')=1
OR CLIENT_LOCALE = 'enUS'
```

Specifying predicates on Extended Search fixed columns

An SQL statement that contains an Extended Search nickname must specify a predicate for the nickname in the WHERE clause. However, a predicate on an Extended Search fixed column does not count as a predicate.

Examples — incorrect syntax

The following example shows a query that is incorrect because it does not contain a predicate:

```
SELECT * FROM ES_NICKNAME
```

The following example shows a query that is incorrect because the only predicate is on a fixed column:

```
SELECT * FROM ES_NICKNAME WHERE DOC_RANK < 20
```

Specifying unbound predicates

A predicate on a user-defined column will be handled by the Extended Search wrapper only if the predicate value is a constant. If the predicate value is unbound, the predicate will be handled by the DB2 engine. If an unbound predicate is the only predicate in an SQL statement, an error will result. An Extended Search nickname requires a predicate that can be handled by the Extended Search wrapper.

Examples — correct syntax

The WHERE statement in the following example shows a predicate that will be handled by the Extended Search wrapper:

```
SELECT *
FROM   ES_NICKNAME
WHERE  Author = 'Ernest Hemingway'
```

Examples — incorrect syntax

The WHERE statement in the following example shows a predicate that will be handled by DB2:

```
SELECT *
FROM   ES_NICKNAME_1, ES_NICKNAME_2
WHERE  ES_NICKNAME_1.Author = ES_NICKNAME_2.Author
```

Joining queries with an OR predicate

The Extended Search wrapper cannot search different nickname tables, or nickname tables and database tables, that are joined by a simple OR predicate. You can use an OR predicate only within the same nickname.

Examples — incorrect syntax

```
SELECT *
FROM   ES_Nickname as N1, TABLE as T1
WHERE  N1.Column1 = 'abc' OR T1.Column1 = 'abc'

SELECT *
FROM  ES_Nickname_1 as N1, ES_Nickname_2 as N2
WHERE N1.USerdefCol = 'abc' OR N2.USerdefCol = 'cdf'

SELECT *
FROM  ES_Nickname_1 as N1, ES_Nickname_2 as N2
WHERE ESWRAPPER.ES_SEARCH(N1.DOC_RANK, '"IBM"')=1 OR
      ESWRAPPER.ES_SEARCH(N2.DOC_RANK, '"LOTUS"')=1
```

Related tasks:

- “Registering the Extended Search custom function” on page 299

Related reference:

- "Extended Search wrapper - Example queries" on page 303
- "Extended Search wrapper - Generalized query language" on page 305

Extended Search wrapper - Example queries

To run queries with the Extended Search wrapper, you specify a registered nickname and nickname columns in your SQL statements the same way that you specify a typical DB2 table name and table columns.

In this sample search scenario, a hospital team needs to search and compare the latest medical research. To search a wide variety of sources, the hospital uses an Extended Search server. The Extended Search domain includes an application named MedResearch and several categories that are configured to search document-based databases, mail servers, and the Web.

In addition to searching, the team needs to compare the results from various searches. For example, they need to identify people who published articles within a certain time frame, recently purchased herbs and vitamins, discussed alternative medicine with colleagues through e-mail, and applied to renew a medical license. The Extended Search wrapper, with its ability to integrate unstructured Extended Search data into DB2 for structured retrieval, provides the solution.

The hospital team decides to create the following three nicknames, one for searching document repositories, one for searching e-mail systems, and one for searching specific Web sources. The Owner and Date fields are defined as mapped fields in the Extended Search configuration database, which enables you to use them in joins regardless of how the fields are named in the native data sources.

Document nickname:

```
CREATE NICKNAME MedDocs ( Owner     VARCHAR(80),
                          Date      DATE,
                          Title     VARCHAR(80),
                          Abstract  VARCHAR(200) )
FOR SERVER esServer OPTIONS ( APPLICATIONID 'MedResearch',
                          CATEGORY 'AMA Library;Medical Records;Pharmacy',
                          VERTICAL_TABLE 'YES',
                          TIMEOUT '60', MAXHITS '100',
               TOTALMAXHITS '1000' )
```

E-mail nickname:

```
CREATE NICKNAME MedMail ( Owner    VARCHAR(80),
                          To       VARCHAR(80),
                          Date     DATE,
                          Subject  VARCHAR(80) )
FOR SERVER esServer OPTIONS ( APPLICATIONID 'MedResearch',
                              CATEGORY 'Exchange Server;Lotus Notes',
                              VERTICAL_TABLE 'YES', )
                              TIMEOUT '60', MAXHITS '100',
                              TOTALMAXHITS '1000' )
```

Web nickname:

```
CREATE NICKNAME MedWeb ( WebTitle VARCHAR(255),
                         WebDescription VARCHAR(1000) )
FOR SERVER esServer OPTIONS ( APPLICATIONID 'MedResearch',
                              DATASOURCES 'Google!;Alta Vista;CNN',
                              TOTALMAXHITS '500' )
```

The following query searches for documents that contain the phrase Artificial Liver in the title and the abbreviation MARS in the document content. The result set should exclude any documents that were published before the year 2001.

```
SELECT OWNER, DOC_CONTENT
FROM   MedDocs
WHERE  ESWRAPPER.ES_Search(DOC_RANK, '( ( TOKEN:EXACT "MARS") AND
                          ( ("TITLE" IN "Artificial Liver") AND
                          ("DATE" >= "01/01/2001") ) ) ') = 1
```

The following query searches for e-mail that was written during the past few months that discussed alternative medicine:

```
SELECT *
FROM   MedMail
WHERE  ESWRAPPER.ES_Search(DOC_RANK, '(
                          ("SUBJECT" IN "alternative medicine") AND
                          ("DATE" BETWEENI "03/01/2002" AND
                           "09/30/2002") ) ') = 1
```

The following query searches Web sources that refer to complementary and alternative medicine (CAM) therapy and its acceptance by the American public:

```
SELECT WebTitle, WebDescription
FROM   MedWeb
WHERE  ESWRAPPER.ES_Search(DOC_RANK, '(
                           TOKEN:EXACT "CAM therapy" ) AND
                           ( TOKEN:FUZZY "United States" ) ' ) = 1
```

The following query searches for recently licensed doctors who purchased large quantities of herbs or vitamins from the hospital pharmacy. The query then matches up the names of those doctors with persons who wrote e-mail about alternative medicine.

```
SELECT  N2.OWNER, N2.DATE
FROM    MedDocs as N1,
MedMail as N2
```

```
WHERE   ESWRAPPER.ES_SEARCH(N1.DOC_RANK, ' (
                           ("LICENSE_DATE" >= "01/01/2002")  AND
                  ( ( ( "PRODUCT" = "HERB") OR ("PRODUCT" = "VITAMIN") ) AND
                  ("QUANTITY" > "1000") ) ) ' ) = 1
AND ESWRAPPER.ES_SEARCH(N2.DOC_RANK, ' ("SUBJECT" IN
                           "alternative medicine") ') = 1
AND N1.OWNER = N2.OWNER
```

Related concepts:

- "Extended Search nicknames" on page 290
- "Extended Search vertical tables" on page 292

Related tasks:

- "Registering nicknames for Extended Search data sources" on page 297
- "Registering the Extended Search custom function" on page 299

Related reference:

- "Extended Search wrapper - Generalized query language" on page 305
- "CREATE FUNCTION statement syntax - Extended Search wrapper" on page 335
- "CREATE NICKNAME statement syntax - Extended Search wrapper" on page 343
- "Extended Search wrapper - Query guidelines" on page 300

Extended Search wrapper - Generalized query language

Queries that you pass to an Extended Search server through the Extended Search wrapper can contain search expressions in generalized query language (GQL), the query language of Extended Search.

For example, assume a user wants to find all employees whose names start with JO in a relational database that contains a table with employee information. You might issue the following query in GQL:

```
(LIKE "EMPLOYEE_NAME" "JO")
```

You might issue the same query in SQL as follows:

```
SELECT * FROM EMP.TABLE WHERE EMPLOYEE_NAME LIKE JO%
```

Like SQL, the wrapper supports infix notation, a syntax that requires operators to be between the field name and a comparison value. The native Extended Search GQL grammar uses prefix notation, a syntax that requires operators to precede the fields and values that you want to evaluate.

Compare the following query expressions that search for documents that contain the word IBM in the TITLE field:

Infix GQL

```
("TITLE" IN "IBM")
```

Prefix GQL

```
(IN "TITLE" "IBM")
```

When you submit a query with the Extended Search wrapper, the API converts the infix SQL statements to prefix GQL for processing by Extended Search.

The following syntax description shows the Backus-Naur Form specification for the Extended Search grammar that you can use in queries.

```
expr:           pattern_expr
              | bool_expr
              | field_expr
              | prox_expr

pattern_expr:   STRING
              | token_expr

token_expr:     ( TOKEN [:CASE] [:STEM] [:EXACT] [:WEIGHT "x"]
                        [:WILD] [:FUZZY] STRING )

bool_expr:      (expr_list bool_operator [:WEIGHT "x"] expr )

bool_text_expr: (text_expr_list  bool_operator [:WEIGHT "x"] text_expr )

text_expr:      pattern_expr
              | bool_text_expr
              | prox_expr

text_expr_list: text_expr
              | text_expr_list text_expr

expr_list:      expr
              | expr_list expr

field_expr:    ( field_name operator_1 [:WEIGHT "x"] text_expr )
              | ( field_name operator_2 [:WEIGHT "x"] value )
              | ( field_name operator_3 [:WEIGHT "x"] value_1 AND value_2 )
              | ( field_name operator_4 value )

prox_expr:     ( prox_op [:COUNT "x"][:ORDER][:MATH "y"][:WEIGHT "x"]
                expr_list expr )

prox_op:        DOCUMENT
              | PARAGRAPH
              | SENTENCE
              | WORD
```

```
               | CHARACTER

operator1:       START
               | END
               | IN
               | =

operator_2:      =
               | >
               | >=
               | <
               | <=
               | EQ
               | GT
               | GTE
               | LT
               | LTE

operator_3:      BETWEENI
               | BETWEENE
               | LIKE

bool_operator:   AND
               | OR
               | NOT
```

For complete information about the GQL grammar, see *Extended Search Programming*, which is available on the Resources page of the IBM Lotus Extended Search Web site:

`http://www.lotus.com/products/des.nsf/wdocuments/resources`

Related tasks:

- "Registering the Extended Search custom function" on page 299

Related reference:

- "Extended Search wrapper - Example queries" on page 303
- "CREATE FUNCTION statement syntax - Extended Search wrapper" on page 335
- "Extended Search wrapper - Query guidelines" on page 300

Messages for the Extended Search wrapper

This topic describes messages that you might encounter while you work with the Extended Search wrapper.

Table 59. Messages issued by the wrapper for Extended Search

Error Code	Message	Explanation
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason: INTERNAL Extended Search WRAPPER ERROR - RC: *xxx*.)	Record the reason code (a number from 901 to 999) and contact IBM Software Support.
SQL0973N	Not enough storage is available in the Application heap to process the statement.	The Extended Search wrapper was not able to allocate memory in the Application heap. To resolve the problem, increase the Application heap size and try the statement again. For example: `db2 update db cfg` `for` *`db-name`* `using` *`heap-name heap-size`* If the error continues after you increase this value, contact IBM Software Support.
SQL1822N	Unexpected error code ″<error_code>″ received from data source ″Extended Search wrapper″. Associated text and tokens are ″<tokens>″.	The remote Extended Search server returned an error while processing a search request. The error also returned a token that indicates what caused the error on the remote server. If tracing is enabled for the Extended Search server, review the trace log file for diagnostic help.
SQL1823N	No data type mapping exists for data type ″<data_type>″ from server ″<server_name>″.	A column in a CREATE NICKNAME statement or ALTER NICKNAME statement uses a data type that is not supported by the Extended Search system. This error can also occur during query processing. To solve the problem if it occurs while the query is being processed, drop the nickname table and create a new nickname.
SQL1825N	This SQL statement cannot be handled in a federated environment.	The current SQL statement cannot be handled by the Extended Search wrapper. To solve the problem, see the Extended Search wrapper documentation, change the SQL statement as needed, and submit the request again.

Table 59. Messages issued by the wrapper for Extended Search (continued)

Error Code	Message	Explanation
SQL1833N	Connection to remote Extended Search server ″<host_name>″ on port ″<port_number>″ could not be established or was terminated.	The Extended Search wrapper tried to connect to the remote Extended Search server at the specified port but the connection could not be established or was terminated by the remote server. Verify the host name and port number of the remote Extended Search server, make sure that the Extended Search server is running, and try again.
SQL1834N	User-defined column ″<column_name>″ is identical to a fixed column for wrapper ″<wrapper_name>″ but uses a different data type.	A CREATE NICKNAME statement or ALTER NICKNAME statement contains a user-defined column that has the same name as a fixed column for the specified Extended Search wrapper but uses a different data type. You do not need to specify fixed columns in the column definition of a CREATE NICKNAME statement. If you do, make sure that the fixed column name, data type, and data type length match the fixed column definition. You cannot ALTER a fixed column name or data type.
SQL1835N	Extended Search object ″<object_name>″ of type ″<object_type>″ could not be found on the remote Extended Search server ″<host_name>″.	The specified Extended Search object could not be found on the specified remote Extended Search server. Verify that the object name is defined on this Extended Search server and that it is of the specified object type. Also verify that the spelling of this object is correct.
SQL1836N	No column mapping exists between user-defined column ″<column_name>″ and a field name on the remote Extended Search server ″<host_name>″.	None of the data sources that are included in a DATASOURCE or CATEGORY option contain a field name the matches the specified user-defined column name. Verify that the column name is a field in at least one of the data sources in the DATASOURCE option, or in at least one of the data sources that belongs to a category in CATEGORY option, and submit the statement again.

Table 59. Messages issued by the wrapper for Extended Search (continued)

Error Code	Message	Explanation
SQL1837N	The required option ″<option_name>″ of type ″<object_type>″ on wrapper ″<wrapper_name>″ cannot be dropped.	You cannot drop a required option. Change the ALTER statement to use SET instead of DROP. Correct the search statement and submit the request again. Consult the DB2 SQL Reference for information about creating valid SQL search statements. If the search statement includes the ES_SEARCH function, consult the Extended Search wrapper documentation for information about using Extended Search generalized query language (GQL).
SQL1838N	The search statement ″<option_name>″ is not a valid Extended Search query.	The Extended Search wrapper tried to process the specified search statement but the query failed because the statement does not use proper query syntax. Consult the *DB2 SQL Reference* for information about creating valid SQL search statements. If the search statement includes the ES_SEARCH function, consult the Extended Search wrapper documentation for information about using Extended Search generalized query language (GQL).
SQL1839N	One or more search parameters are not valid.	The Extended Search wrapper tried to use the specified search parameters, but they are not valid for Extended Search. Consult the Extended Search wrapper documentation, correct the invalid parameters, and submit the request again.
SQL1881N	″<option_name>″ is not a valid ″<option_type>″ option for ″<object_name>″.	The specified option is not valid for the specified object (wrapper, server, nickname, column, or user mapping). See the Extended Search wrapper documentation, remove or change the invalid option, and submit the statement again.
SQL1882N	The ″<option_type>″ option ″<option_name>″ cannot be set to ″<option_value>″ for ″<object_name>″.	The specified option value is not valid for the specified object (wrapper, server, nickname, column, or user mapping). See the Extended Search wrapper documentation, change the invalid option value, and submit the statement again.

Table 59. Messages issued by the wrapper for Extended Search (continued)

Error Code	Message	Explanation
SQL1883N	″<option_name>″ is a required ″<option_type>″ option for ″<object_name>″.	A required option for the Extended Search wrapper was missing from the statement to create, alter, or initialize the specified object (wrapper, server, nickname or user mapping). See the Extended Search wrapper documentation, add the required option, and submit the statement again.

For more information about messages, see the *DB2 Message Reference*. You might also want to consult the Extended Search product messages in *Extended Search Administration*. If you receive errors about improper GQL query syntax, see *Extended Search Programming*. The Extended Search documents are available on the Resources page of the IBM Lotus Extended Search Web site:

`http://www.lotus.com/products/des.nsf/wdocuments/resources`

Related reference:

- “sql0900” in the *Message Reference: Volume 2*
- “sql1800” in the *Message Reference: Volume 2*

Chapter 19. Configuring access to HMMER data sources

This chapter explains what HMMER is, how to add HMMER data sources to your federated system, and lists the error messages associated with the HMMER wrapper.

What is HMMER?

HMMER is a tool that you can use to search gene sequence databases that use statistical models or profile Hidden Markov Models (HMMs). You can download HMMER at no charge from http://hmmer.wustl.edu/. HMMER was first developed to improve BLAST search capabilities. This HMMER wrapper version uses a gene sequence to search a database of models and determine to which family the test gene sequence might belong.

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 software package is tightly tied to the construction and use of the PFAM database. The HMMER wrapper supports the use of the **hmmpfam** program, which searches a profile HMM database, such as PFAM, with a particular gene sequence.

The HMMER wrapper starts the hmmpfam utility, which uses profile HMMs to model the primary structure consensus of a family of protein or nucleic acid sequences, as described in Table 60.

Table 60. HMMER utilities

HMMER utility	Description
hmmpfam	Calculates how well each model matches a specified sequence and a database of models. The match is expressed in terms of statistical significance.
hmmalign	Aligns multiple gene sequences to a profile HMM.

Table 60. HMMER utilities (continued)

HMMER utility	Description
hmmbuild	Builds a profile HMM from a multiple gene sequence alignment.
hmmcalibrate	Determines appropriate statistical significance parameters for a profile HMM before a database search is performed.
hmmconvert	Converts HMMER profile HMMs to other formats such as Genetics Computer Group (GCG) profiles.
hmmemit	Generates gene sequences that use probability models from a profile HMM.
hmmfetch	Retrieves an HMM from an HMM database.
hmmindex	Creates a binary server-side include (server-side includes (SSI)) index for an HMM database.
hmmsearch	Searches a gene sequence database with a profile HMM and finds additional homologues of a modeled family.

From a client, users or applications submit SQL statements with HMMER-specific predicates that map to hmmpfam command-line options. These SQL statements and predicates are sent to your federated database server, which includes the HMMER wrapper.

The HMMER wrapper transforms the query into a format that the HMMER application can interpret and starts the hmmpfam utility to run the query. The server that runs hmmpfam can be a separate system from the system with the federated database server. A special daemon program runs on your HMMER server. This daemon, which uses information from a daemon configuration file, receives the query request from the federated database server and sends it to the HMMER application. The HMMER application then runs on a profile database.

Figure 14 on page 315 shows how HMMER works with your federated system.

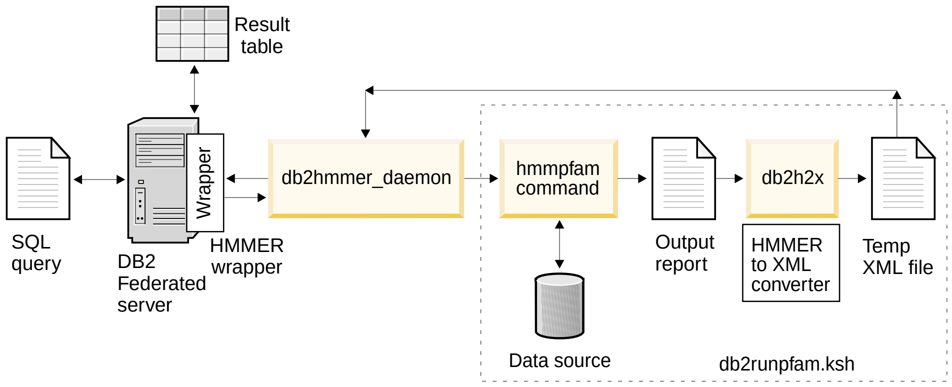


Figure 14. How the HMMER wrapper works

The daemon returns HMMER 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 utilities, 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.

Figure 15 on page 316 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 15. Sample query run on 7LES_DROME data

The HMMER wrapper transforms the query results into the relational table shown in Table 61.

Table 61. HMMER returns results in a relational table when the HMMER wrapper is integrated with your federated system

Model	ModelScore	DomainNumber	DomainScore
pkinase	+3.04100000000000E+002	1	+3.04100000000000E+002

Table 61. HMMER returns results in a relational table when the HMMER wrapper is integrated with your federated system (continued)

Model	ModelScore	DomainNumber	DomainScore
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 fully relational form and can be joined with data from other data sources.

Related concepts:
- "What are table-structured files?" on page 143
- "What is Documentum?" on page 157
- "What is Excel?" on page 191
- "What is BLAST?" on page 205
- "What is XML?" on page 231
- "What is Entrez?" on page 261
- "What is Extended Search?" on page 287

Adding HMMER to a federated system

Procedure:

To add the HMMER data source to a federated server:
1. Verify that you installed the correct version of the hmmpfam executable.
2. Configure the HMMER daemon.Configure the HMMER daemon.
3. Start the HMMER daemon.Start the HMMER daemon.
4. Register the wrapper by issuing the CREATE WRAPPER statement.Register the wrapper by issuing the CREATE WRAPPER statement.
5. Optional: Set the DB2_DJ_COMM environment variable to improve query performance.
6. Register the server by issuing the CREATE SERVER statement.Register the server by issuing the CREATE SERVER statement.

7. Register nicknames by issuing the CREATE NICKNAME statement.Register nicknames by issuing the CREATE NICKNAME statement.

You can run the statements from the DB2 command-line processor. After you add the HMMER wrapper to your federated system, you can run queries on the HMMER data source.

Related tasks:

- "Verifying that the correct version of the hmmpfam executable is installed" on page 318

Verifying that the correct version of the hmmpfam executable is installed

Verifying that the correct version of the hmmpfam executable is installed is part of the larger task of adding HMMER to a federated system. Verify that you have the latest version of the hmmpfam executable installed on your HMMER server with the following procedure.

Procedure:

To check the version level of your hmmpfam executable:

1. Run the following from the command line and note the version number located in the output file:

   ```
   hmmpfam -h
   ```
2. If you do not have the latest version of the hmmpfam executable (HMMER 2.2 or above), download the files from http://hmmer.wustl.edu/.

The next task in this sequence of tasks is configuring the HMMER daemon.

Related tasks:

- "Configuring the HMMER daemon" on page 318

Configuring the HMMER daemon

Configuring the HMMER daemon is part of the larger task of adding HMMER to a federated system. The HMMER wrapper requires that a HMMER daemon runs on your AIX-based machine. You must also be able to access the HMMER daemon through Transmission Control Protocol/Internet Protocol (TCP/IP) from your federated server. The daemon runs separately from the wrapper and DB2 Universal Database and listens for HMMER job requests from the wrapper. The daemon-executable file, `db2hmmer_daemon`, can reside in any directory on the HMMER server.

During DB2 Universal Database installation, the daemon executable is placed on the same computer as the federated server. On the AIX platform, the directory in which the daemon executable is placed is `/usr/opt/db2_08_01/bin`. If you did not install HMMER and the federated server on the same computer, you must copy the daemon executable to a location of your choice on the computer where you installed HMMER. You must also copy the configuration file HMMER_DAEMON.config, the supplied conversion utility named db2h2x, and the shell script named db2runpfam.ksh.

Programs must be executable. If not, run the AIX command `chmod a+x db2hmmer_daemon db2h2x db2runpfam.ksh` on the target system to make programs executable.

The HMMER daemon must have:

- Execute access to the hmmpfam-executable file 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.

The HMMER daemon requires a configuration file. A sample daemon configuration file, named `HMMER_DAEMON.config`, is placed in the directory `DB2PATH/samples/lifesci`, where `DB2PATH` is the directory in which DB2 Universal Database is installed. `HMMER_DAEMON.config` is the default name for the file.

Copy the configuration file to any location accessible to the daemon, rename it if you want, and edit it to work with your data sources. By default, the HMMER daemon looks for its configuration information in the working directory from which it was started.

Procedure:

To configure the daemon, 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 will listen for HMMER job requests submitted by the wrapper.

MAX_PENDING_REQUESTS
: This is the maximum number of HMMER job requests that can block 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 will create its log file. This file will contain 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 will be 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 will create the temporary file to store the HMMER output data. Data will be read from this file and passed back to the wrapper via the network connection, at which point the daemon cleans up the temporary file.

RUNPFAM_PATH
: This is the fully-qualified name of the db2runpfam.ksh shell script provided.

HMMPFAM_PATH
: This is the fully-qualified name of the hmmpfam executable file on the machine that runs the daemon.

H2X_PATH
: This is the fully-qualified name of the db2h2x (HMMER to XML) conversion program provided with the daemon.

database specification entry
: Specifies the location of a profile database. When you create a nickname for the data source with the CREATE NICKNAME statement, make note of the entry name that you use in the DATASOURCE option of the configuration file. You must specify this entry name for the daemon to function properly.

The configuration file must contain at least one database specification entry in the following format:

entry_name = path to profile_database

For example, to specify the MYHMMS profile database, you would add the following line to the daemon configuration file:

```
myhmms=/home/user_ID/myhmms
```

The configuration file must end with a new line character.

Example:

The following example shows the contents of a sample configuration file, with the required options and profile database specification for PFAM.

```
=
DAEMON_PORT=4098
MAX_PENDING_REQUESTS=10
DAEMON_LOGFILE_DIR=./
Q_SEQ_DIR_PATH=./
HMMER_OUT_DIR_PATH=./
RUNPFAM_PATH=./db2runpfam.ksh
HMMPFAM_PATH=/home/user_id/hmmer/bin/hmmpfam
H2X_PATH=/home/user_id/sqllib/bin/db2h2x
myhmms=/home/user_id/hmmer/tutorial/myhmms
pfamls=/home/user_id/hmmer/pfam/Pfam_ls
```

1. Make sure that you start the first line with an equal sign or the daemon will not start. You will get an error message unless you specify the DAEMON_PORT.
2. Make sure that you end the last line in the configuration file with a new line. Otherwise, you will get an error message when you run a HMMER query that uses the data source listed on the last line.

The next task in this sequence of tasks is starting the HMMER daemon.

Related tasks:

- "Starting the HMMER daemon" on page 321

Starting the HMMER daemon

Starting the HMMER daemon is part of the larger task of adding HMMER to a federated system. Before you can access HMMER data sources, you must start the HMMER daemon.

Prerequisites:

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.

Procedure:

If the following conditions are true:

- You are in the daemon installation directory.
- You did not change the name of the daemon configuration file.
- Put the configuration file in the same directory as the daemon executable file.

Type the following at the command line to start the HMMER daemon:

```
db2hmmer_daemon
```

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

If you changed the configuration file name or directory location, use the -c option on the wrapper daemon command to point the daemon executable to the new name or location.

For example, the following command causes the wrapper daemon to look for its configuration information in a file called `HMMER_D.config` in the subdirectory `cfg`.

```
db2hmmer_daemon -c cfg/HMMER_D.config
```

The next task in this sequence of tasks is registering the HMMER wrapper.

Related tasks:

- "Registering the HMMER wrapper" on page 322

Registering the HMMER wrapper

Registering the HMMER wrapper is part of the larger task of adding HMMER to a federated system. You must register the wrapper in order to access a data source. Wrappers are mechanisms that federated servers use to communicate with and retrieve data from data sources. Your computer installs wrappers as library files.

Procedure:

To register the HMMER wrapper, issue the CREATE WRAPPER statement.

For example, to create a HMMER wrapper on AIX called `my_hmmer` from the default library file, `libdb2lshmmer.a`, submit the following statement:

```
CREATE WRAPPER my_hmmer LIBRARY 'libdb2lshmmer.a'
  OPTIONS(DB2_FENCED 'N');
```

For Windows, use `db2lshmmer.dll` instead of `libdb2lshmmer.a`.

The next task in this sequence of tasks is setting the DB2_DJ_COMM environment variable for the HMMER wrapper.

Related tasks:

- "Setting the DB2_DJ_COMM DB2 profile variable for the HMMER wrapper" on page 323

Related reference:

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*

Setting the DB2_DJ_COMM DB2 profile variable for the HMMER wrapper

Setting the DB2_DJ_COMM DB2 profile variable for the HMMER wrapper is an optional task of the larger task of adding HMMER to a federated system. To improve performance when you access HMMER data sources, set the DB2_DJ_COMM DB2 profile variable. This variable determines whether the federated server loads the wrapper upon initialization.

Procedure:

To set the DB2_DJ_COMM DB2 profile variable, submit the `db2set` command with the wrapper library that you specified in the associated CREATE WRAPPER statement.

For example:

```
db2set DB2_DJ_COMM=libdb2lshmmer.a,libdb2lshmmerF.a,libdb2lshmmerU.a
```

Where `libdb2lshmmer.a`, `libdb2lshmmerF.a`, and `libdb2lshmmerU.a` refer to main, fenced, and unfenced library names on an AIX platform.

Ensure that there are no spaces on either side of the equal sign (=).

To avoid unnecessary overhead when you load the wrapper libraries at database startup, specify only libraries that you intend to access.

The next task in this sequence of tasks is registering the server for a HMMER data source.

Related tasks:
- "Registering the server for a HMMER data source" on page 323

Registering the server for a HMMER data source

Registering the server for a HMMER data source is part of the larger task of adding HMMER to a federated system. After you register the wrapper, you must register a corresponding server.

Procedure:

To register the HMMER server to the federated system, use the CREATE SERVER statement.

You must register each server on which a HMMER search that uses the hmmpfam executable and daemon instance runs.

For example, for a wrapper called my_hmmer that uses the CREATE WRAPPER statement for hmmpfam searches, you would register the hmmer_server1 server with the following statement:

```
CREATE SERVER hmmer_server1
 TYPE pfam
   VERSION 2.2
   WRAPPER my_hmmer
   OPTIONS (NODE 'someserver.someschool.edu', DAEMON_PORT '4098')
```

Arguments

TYPE Required: Determines the type of search performed that uses the specified server. In this release, the value must be set to PFAM.

VERSION
Required: Specifies the server version, which should match the version of the hmmpfam executable version that you are running (HMMER 2.2 or above).

WRAPPER
Required: Specifies the name of the wrapper that you registered when you issued the CREATE WRAPPER statement.

Options

You must enclose server option values in single quotation marks.

NODE
Required: Specifies the host name of the server on which the HMMER daemon process runs.

DAEMON_PORT
Optional: Specifies the port number on which the daemon listens for HMMER job requests. The port number must be the same number specified in the DAEMON_PORT option of the daemon configuration file. The default is 4098.

PROCESSORS
Optional: Specifies the number of processors that the HMMER program uses. This option is equivalent to the --cpu option of the `hmmpfam` command. Example: `PROCESSORS '2'`.

HMMPFAM_OPTIONS
Optional: 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. Example: `HMMPFAM_OPTIONS '--xnu --pvm'`. In this example, instead of using hmmpfam options, you would use the appropriate column name in the WHERE clause of the SQL query.

The next task in this sequence of tasks is registering nicknames for HMMER data sources.

Related tasks:

- "Registering nicknames for HMMER data sources" on page 325

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*

Registering nicknames for HMMER data sources

Registering nicknames for HMMER data sources is part of the larger task of adding HMMER to a federated system. After you register a server, you must register a corresponding nickname. When you refer to a HMMER data source in a query, you use nicknames.

Procedure:

To register a HMMER nickname, use the CREATE NICKNAME statement. You must define a separate nickname for each profile database that you want to query.

The CREATE NICKNAME statement syntax for HMMER is:

```
►►──CREATE NICKNAME──nickname──FOR SERVER──server-name──OPTIONS──(──────────────────►

►─DATASOURCE──'data_source_name' ──┬──────────────────────────────────┬────────────►
                                   └─,──TIMEOUT──'timeout_duration' ──┘

►─┬──────────────────────────────┬─────────────────────────────────────────────────►◄
  └─,──HMMTYPE──'hmmer_type' ────┘


►►──)──────────────────────────────────────────────────────────────────────────────►◄
```

Example:

```
CREATE NICKNAME myhmms
FOR SERVER hmmer_server
OPTIONS(DATASOURCE 'myhmms',TIMEOUT '60')
```

Nickname options

You must enclose nickname option values in single quotation marks.

DATASOURCE

Required: The name of the data source on which you run the HMMER search. The exact string that is used here must be present in the configuration file of the HMMER daemon.

TIMEOUT

Optional: The maximum time, in minutes, that the HMMER wrapper will wait for results from the daemon. The default value is 60.

HMMTYPE

Optional: The alphabet that is used in both models and gene sequences. The value may be either NUCLEIC or PROTEIN and is not case sensitive. The default value is PROTEIN.

Fixed columns

The CREATE NICKNAME statement automatically creates fixed columns. You can reference these fixed columns in SQL queries as part of the nickname definition. Fixed columns do not appear in the CREATE NICKNAME statement. There are two types of fixed columns, input and output.

Input fixed columns

You use input-fixed columns as parameter-passing predicates in SQL queries. They pass standard hmmpfam options to HMMER. HMMER then runs on the specified data source that uses these hmmpfam options. You can also reference input-fixed columns in the query-select list, which are returned as part of the results table. Table 62 lists input-fixed columns.

Table 62. Input fixed columns

Name	Data type	Description	Allowed Operators	hmmpfam Option	Returned Value
HmmQSeq	varchar(32000)	Input gene sequence that is used to search	=		Same as input; this column is required.
ModelEValue	double	Estimated e-value	<	-E *n*	See output.
ModelScore	double	Raw score	>	-T *n*	See output.
DBSize	integer	Calculate e-values as if the database had ’n’ gene sequences	=	-Z *n*	Same as input; uses hmmpfam default if not specified.
CutMode	char(2)	Cutoff mode; may be ga, tc or nc (case sensitive)	=	--cut_ga --cut_tc --cut_nc	Same as input; NULL if not specified.

Table 62. Input fixed columns (continued)

Name	Data type	Description	Allowed Operators	hmmpfam Option	Returned Value
DomainScore	double	Domain score	>	--domT *n*	See output.
DomainEValue	double	Domain e-value	<	--domE *n*	See output.
ForwardAlgorithm	char	Use Forward algorithm rather than Viterbi; value may be ’Y’ or ’N’	=	--forward	Same as input; ’N’ is the default.

Output fixed columns

You can use the output fixed columns that are returned in the query results table as predicates. Table 63 lists output-fixed columns.

Table 63. Output fixed columns

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.

Table 63. Output fixed columns (continued)

Name	Data type	Description
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.

There are no further tasks in this sequence of tasks.

Related reference:

- "HMMER data source – complete example" on page 328
- "CREATE NICKNAME statement - Example for HMMER wrapper" on page 328

CREATE NICKNAME statement - Example for HMMER wrapper

The following CREATE NICKNAME statement defines the nickname myhmms:

```
CREATE NICKNAME myhmms
FOR SERVER hmmer_server
OPTIONS(DATASOURCE 'myhmms',TIMEOUT '60')
```

After you issue the CREATE NICKNAME statement, you can use the nickname myhmms to query your federated system. You can also join the myhmms nickname with other nicknames and tables in your federated system.

There are no further tasks in this sequence of tasks.

HMMER data source – complete example

SQL statements for HMMER data sources must contain special input predicates that are used to pass standard HMMER options to the hmmpfam-executable file.

Restrictions:

To be valid, every query passed to the HMMER wrapper must contain at least the `HmmQSeq` input predicate. All other predicates are optional.

Procedure:

To construct a HMMER query, use the input predicates in the WHERE clause of your SQL statement.

The following complete example shows all the statements you need to create and run a query that uses HmmQSeq as a search sequence:

```
CREATE WRAPPER hmmer_wrapper
LIBRARY 'libdb2lshmmer.a'
OPTIONS (DB2_FENCED 'N');

CREATE SERVER hmmer_serv
TYPE pfam VERSION 2.2
WRAPPER hmmer_wrapper
OPTIONS(NODE 'HMMERserv.MyCompany.com');

CREATE NICKNAME myhmms
FOR SERVER hmmer_serv
OPTIONS(DATASOURCE 'myhmms', TIMEOUT '1');

-- Run the 7LES_DROME gene sequence on the myhmms nickname
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 myhmms
WHERE HmmQSeq =
     'MTMFWQQNVDHQSDEQDKQAKGAAPTKRLNISFNVKIAVNVNTKMTTTHINQQAPGTSS...';
```

Related tasks:

- "Registering nicknames for HMMER data sources" on page 325
- "Construct new HMMER queries with samples" on page 330

Construct new HMMER queries with samples

The following sample HMMER queries illustrate how to construct queries for HMMER data sources.

Procedure:

To run queries, use the following examples as a guide.

In these queries, the name that is used for each nickname describes the type of HMMER search and data source. With descriptive names, one does not need to list registration statements with each sample query. Also, some examples illustrate HMMER wrapper behavior when joined with other data sources.

Query 1.

```
SELECT Model, ModelScore, ModelEValue, DomainNumber, DomainScore, DomainEvalue
FROM myhmms
WHERE HmmQSeq = 'MTMFWQQNVDHQSDEQDKQAKGAAPTKRLNISFNVKIAVNVNTKMTTTHINQ...'
```

When this SQL statement runs, the wrapper will perform an hmmpfam search of myhmms that uses the indicated gene sequence. The wrapper will return all of the available columns, including both the input parameter columns and the HMMER result columns.

Query 2.

```
SELECT Model, ModelScore, ModelEValue
FROM myhmms
WHERE HmmQSeq = 'MTMFWQQNVDHQSDEQDKQAKGAAPTKRLNISFNVKIAVNVNTKMTTTHINQ...'
AND ModelScore > 0
```

When this SQL statement runs, the wrapper will perform an hmmpfam search of myhmms that uses the indicated gene sequence. In addition, the wrapper passes the -T 0 option (see Table 62 on page 326) to the hmmpfam command. The wrapper will return the three columns listed after SELECT.

Query 3.

```
SELECT Model, DomainNumber, DomainScore, DomainEValue
FROM myhmms
WHERE HmmQSeq = 'MTMFWQQNVDHQSDEQDKQAKGAAPTKRLNISFNVKIAVNVNTKMTTTHINQ...'
AND ModelEValue < 1
ORDER BY DomainScore DESC
```

When this SQL statement runs, the wrapper will perform an hmmpfam search of myhmms that uses the indicated gene sequence. In addition, the wrapper passes the -E 1 option (see Table 62 on page 326) to the hmmpfam command.

The wrapper will return the four columns listed after SELECT and sort the result by DomainScore from highest to lowest.

Related tasks:

- “Entrez data source — Example queries” on page 271

Related reference:

- “Documentum data source – Example queries” on page 179
- “Excel data source – Example queries” on page 195
- “Extended Search wrapper - Example queries” on page 303

Optimization tips for the HMMER wrapper

Running both the wrapper and the daemon on the same server can eliminate potential network communication bottlenecks.

Related tasks:

- “Setting the DB2_DJ_COMM DB2 profile variable for the HMMER wrapper” on page 323

Related reference:

- “Optimization tips and considerations for the table-structured file wrapper” on page 151
- “Optimization tips for the BLAST wrapper” on page 228

Messages for the HMMER wrapper

For the HMMER wrapper to work, you must specify a query that contains a predicate on the HmmQSeq column. When you query a fragment that lacks a predicate on the HmmQSeq column, you get an error.

This section lists and describes messages that you might encounter when you work with the HMMER wrapper.

Table 64. HMMER wrapper messages

Error Code	Message	Explanation
SQL0142N	The SQL statement is not supported.	The SQL query submitted to DB2 could not be processed by the wrapper. Add the required predicate and resubmit. Verify that the operator used in a predicate is valid for that column (see Table 62 on page 326).

Table 64. HMMER wrapper messages (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″Unspecified Error″ received from data source ″Hmmer wrapper″. Associated text and tokens are ″Unable to resolve NODE host name″.	The TCP/IP NODE name specified in CREATE SERVER is invalid.
SQL1822N	Unexpected error code ″Unspecified Error″ received from data source ″Hmmer wrapper″. Associated text and tokens are ″Unable to connect to daemon″.	Either the hmmer_daemon program is not currently running on the target node, or the DAEMON_PORT specified in the CREATE SERVER command does not match the DAEMON_PORT value specified in daemon configuration file HMMER_DAEMON.config.
SQL1822N	Unexpected error code ″Unspecified Error″ received from data source ″Hmmer wrapper″. Associated text and tokens are ″Unknown error from the hmmer daemon″.	The DATASOURCE name specifed in the CREATE NICKNAME statement may not match any of the profile database names listed in the daemon configuration file HMMER_DAEMON.config.
SQL1822N	Unexpected error code ″Unspecified Error″ received from data source ″Hmmer wrapper″. Associated text and tokens are ″FATAL: No such option ″--cut_TC″.	The CutMode predicate must be specified in lower case. Example: WHERE CutMode = ’tc’

Related reference:

- “Messages for the table-structured file wrapper” on page 152
- “Messages for the Documentum wrapper” on page 184
- “Messages for the Excel wrapper” on page 198
- “Messages for the BLAST wrapper” on page 228
- “Messages for the XML wrapper” on page 251
- “Messages for the Entrez wrapper” on page 281
- “Messages for the Extended Search wrapper” on page 307

Chapter 20. Altering nicknames

This chapter explains how to use the ALTER NICKNAME statement to alter previously registered nicknames.

Altering nicknames

You can use the ALTER NICKNAME statement to modify the federated database's representation of a data source or view.

Restrictions:

The ALTER NICKNAME statement can not be used to alter column names for the BLAST, Documentum, or EXCEL wrappers. The ALTER NICKNAME statement can be used to alter column names for the table-structured file and XML wrappers.

Procedure:

To alter nickname column values, you must use the ALTER NICKNAME statement to:

- Change the local data types of these columns
- Add, change, or delete options for these columns

Related tasks:

- "Changing the data type" on page 333
- "Changing the nickname option" on page 334

Changing the data type

You can use the ALTER NICKNAME statement to change the data type of a column.

Procedure:

To change the data type of a column, use the ALTER NICKNAME statement.

For example, the following ALTER NICKNAME statement changes the local data type of the DRUG column to CHAR(30). The DRUG column was originally defined as a CHAR(20) using a CREATE NICKNAME statement. The nickname DRUGDATA1 refers to a local table-structured file called drugdata1.txt.

```
ALTER NICKNAME DRUGDATA1
   ALTER COLUMN DRUG
   LOCAL TYPE CHAR(30)
```

Related tasks:

- "Altering nicknames" on page 333
- "Changing the nickname option" on page 334

Changing the nickname option

You can use the ALTER NICKNAME statement to change a nickname option.

Procedure:

To change a nickname option, use the ALTER NICKNAME statement.

For example, the following ALTER NICKNAME statement changes the fully qualified path for the table-structured file, drugdata1.txt. The path was originally defined as '/user/pat/drugdata1.txt' using a CREATE NICKNAME statement. The nickname DRUGDATA1 refers to a local table-structured file called drugdata1.txt.

```
ALTER NICKNAME DRUGDATA1
   OPTIONS (SET FILE_PATH '/usr/kelly/data/drugdata1.txt')
```

Related tasks:

- "Altering nicknames" on page 333
- "Changing the data type" on page 333

Chapter 21. DDL command reference

This chapter provides details of the syntax statements, arguments, and options for the wrapper DDL commands covered in this book. The statements are ordered alphabetically by statement, then wrapper.

CREATE FUNCTION statement syntax - Extended Search wrapper

```
►►──CREATE FUNCTION ESWRAPPER.ES_SEARCH──(──INTEGER──,──VARCHAR(1024)──)──────────────►

►──RETURNS INTEGER AS TEMPLATE──────────────────────────────────────────────────────►◄
```

INTEGER
Defines the query reference parameter. In a query, this parameter must specify the name of an INTEGER column that is defined in the nickname table for which this custom function is being called. The value must be a bind column of the nickname, not a constant (for example, DOC_RANK).

The reference parameter identifies the nickname to which the ES_SEARCH function should be applied. The parameter itself is not evaluated.

If a SELECT statement contains more than one table in the FROM clause, and the WHERE clause contains an ES_SEARCH statement, the reference parameter allows you to tell DB2 which table a particular search statement belongs to. For example:

```
SELECT *
FROM ES_Nickname_1 as N1, ES_Nickname_2 as N2
WHERE  ESWRAPPER.ES_SEARCH(N1.DOC_RANK, 'IBM')=1 AND
       ESWRAPPER.ES_SEARCH(N2.DOC_RANK, 'LOTUS')=1
```

VARCHAR(1024)
Defines the query expression. In a query, this parameter must specify a string that uses Extended Search generalized query language.

Related tasks:

- “Registering the Extended Search custom function” on page 299

Related reference:

- “CREATE FUNCTION (Sourced or Template) statement” in the *SQL Reference, Volume 2*
- “Extended Search wrapper - Example queries” on page 303
- “Extended Search wrapper - Generalized query language” on page 305

CREATE NICKNAME statement syntax - BLAST wrapper

```
                                  ┌─,──────────────────────────────────┐
                                  ▼                                    │
►►─CREATE NICKNAME─nickname─(────column-name──┤ column-information ├──┴─)──────────►

►─FOR SERVER─server-name─OPTIONS─(─DATASOURCE─'data_source_name' ─────────────────►

►─┬────────────────────────────────────┬─┬──────────────────────────────────┬─)──────►◄
  └─,─PROCESSORS─'processor_number' ─┘ └─,─TIMEOUT─'timeout_duration' ─┘
```

column-information:

```
├──┤ data-type ├──┤ column-option ├──┤ nickname-column-options ├────────────────┤
```

data-type:

```
├──┬─┬─INTEGER─┬──────────────────┬─────────────────────────────────────────┤
   │ └─INT─────┘                  │
   ├─┬─FLOAT─┬─────────────────┬─┤
   │ │       └─(─integer─)─┘   │ │
   │ │        ┌─PRECISION─┐    │ │
   │ └─DOUBLE─┴───────────┴────┘ │
   ├─┬─CHARACTER─┬─┬─────────────────┬─┤
   │ └─CHAR──────┘ └─(─integer─)─┘ │
   └─VARCHAR─(─integer─)────────────┘
```

column-option:

```
├──┬──────────┬──────────────────────────────────────────────────────────┤
   └─NOT NULL─┘
```

nickname-column-options:

```
├──OPTIONS─(─INDEX─'index_number' ─,─DELIMITER─'delimiter' ──────────────────────►

►─┬───────────────────────────────────┬─)──────────────────────────────────────┤
  └─DEFAULT─'new_default_value' ─┘
```

Nickname column options:

Nickname column option values must be enclosed in single quotation marks.

INDEX

The ordinal number of the column on which this option appears in the group of definition line columns. This option is required.

DELIMITER

The delimiter characters that should 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 will signal the end of this field's information. The default is end of line. This option is required, except for the last column specified if you want that column to contain the remainder of the definition line.

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 pre-set default values. The new default value must be of the same type as the value indicated for a given column. This option is optional.

Nickname options:

Nickname option values must be enclosed in single quotation marks.

DATASOURCE

The name of the data source on which the BLAST search will be run. The exact string used here must be present in the configuration file of the BLAST daemon. This option is required.

PROCESSORS

Specifies the number of processors to be used when evaluating a BLAST query. It corresponds to BLAST's `blastall -a` option. This option is optional. The default value is 1.

TIMEOUT
The maximum time, in minutes, that the BLAST wrapper will wait for results from the daemon. The default is 60. This option is optional.

Related tasks:
- "Configuring the BLAST daemon" on page 211
- "Registering nicknames for BLAST data sources" on page 217

Related reference:
- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*
- "CREATE NICKNAME statement - Examples for BLAST wrapper" on page 223

CREATE NICKNAME statement syntax - Documentum wrapper

The syntax for the CREATE NICKNAME statement for Documentum is:

```
                                      ┌─,──────────────────────────────────────────┐
                                      ▼                                            │
►►──CREATE NICKNAME──nickname──(──────column-name──┤ column-information ├──────────┴──)──────────►

►──FOR SERVER──server-name──OPTIONS──(──┬──────────────────────────────┬──────────────────────────►
                                        └─ALL_VERSIONS──┬─'Y'─┬──,─────┘
                                                        └─'N'─┘

►──┬─────────────────────────────┬──┬──────────────────────────────┬──────────────────────────────►
   └─FOLDERS──'folder_string'──,─┘  └─IS_REG_TABLE──┬─'Y'─┬──,─────┘
                                                    └─'N'─┘

►──REMOTE_OBJECT──'remote_object_type'──)────────────────────────────────────────────────────────►◄
```

column-information:

```
├──┤ data-type ├──┤ column-option ├──┬────────────────────────────────────┬──────────────────────┤
                                     └──┤ nickname-column-options ├───────┘
```

data-type:

```
├──┬─SMALLINT──────────────────────────┬──────────────────────────────────────────────────┤
   ├─┬─INTEGER─┬───────────────────────┤
   │ └─INT─────┘                       │
   ├─DOUBLE─┬───────────┬──────────────┤
   │        └─PRECISION─┘              │
   ├─┬─CHARACTER─┬──┬──────────────┬───┤
   │ └─CHAR──────┘  └─(─integer─)──┘   │
   ├─VARCHAR─(─integer─)───────────────┤
   ├─DATE──────────────────────────────┤
   └─TIMESTAMP─────────────────────────┘
```

column-option:

```
├──┬──────────┬───────────────────────────────────────────────────────────────────────┤
   └─NOT NULL─┘
```

nickname-column-options:

```
├──OPTIONS─(─┬──────────────────────────────────────┬──┬───────────────────────────────┬──────►
             └─REMOTE_NAME─'attribute_name' ─,──────┘  └─DELIMITER─'delimiter' ─,──────┘

►─┬──────────────────────────────┬──┬──────────────────────────┬─────────────────────────┤
  └─IS_REPEATING─┬─'Y' ─┬─,──────┘  └─ALL_VALUES─┬─'Y' ─┬──────┘
                 └─'N' ─┘                        └─'N' ─┘
```

Column options associated with the CREATE NICKNAME statement for Documentum are:

NOT NULL
: All single-valued columns except those defined as TIMESTAMP and DATE must be defined as NOT NULL. Repeating attributes must not be defined as NOT NULL in nicknames.

Nickname column options associated with the CREATE NICKNAME statement for Documentum are:

Nickname column option values must be enclosed in single quotation marks.

ALL_VALUES
: Specifies that all values of a repeating attribute will be returned, separated by the specified delimiter. If this option is missing or is 'N', then only the last value of a repeating attribute will be returned. As noted under DELIMITER, ALL_VALUES may only be specified for VARCHAR columns for which the IS_REPEATING option is 'Y' (and is invalid when IS_REG_TABLE = 'Y').

DELIMITER
: Specifies the delimiter string to be used when concatenating multiple values of a repeating attribute. The delimiter can be one or more

characters. The default delimiter is a comma. This option is only valid for attributes of objects with data type VARCHAR where the IS_REPEATING option is set to ’Y’. This option is optional.

IS_REPEATING
Indicates if the column is multi-valued. Valid values are ’Y’ and ’N’. The default is ’N’. This option is optional.

Only the last value is returned for

- non-VARCHAR repeating attributes
- VARCHAR columns when ALL_VALUES ’N’ is specified

To overcome this limitation, you can create a dual definition for the repeating attribute column.

REMOTE_NAME
Specifies the name of the corresponding Documentum attribute or column. This option maps remote attribute or column names to local DB2 column names. It defaults to the DB2 column name. This option is optional.

Nickname column options associated with the CREATE NICKNAME statement for Documentum are:

Nickname option values must be enclosed in single quotation marks.

ALL_VERSIONS
Specifies whether all object versions will be searched. The valid values are ’y’, ’Y’, ’n’, and ’N’. The default value of ’N’ means that only the current object versions are included in query processing. This option is invalid when IS_REG_TABLE = ’Y’. This option is optional.

FOLDERS
Specifies a string that contains one or more logically-combined and syntactically-correct Documentum FOLDER predicates. Specifying FOLDER predicates restricts the set of documents represented by this nickname to those in the designated folders.

When you specify this option, enclose the entire value of the FOLDERS option in single quotes and use double quotes in place of the single quotes within the string.

For example, if you want to insert:

```
FOLDER('/Tools',DESCEND) OR FOLDER('/Cars')
```

Specify the following FOLDERS option:

```
FOLDERS 'FOLDER("/Tools",DESCEND) OR FOLDER("/Cars")'
```

This option is invalid when IS_REG_TABLE = ’Y’. This option is optional.

IS_REG_TABLE
Specifies whether the object specified by the REMOTE_OBJECT option is a Documentum registered table. The valid values are ’y’, ’Y’, ’n’, and ’N’. The default value is ’N’. This option is optional.

You cannot change a nickname from a Documentum object to a registered table (or back) by changing this option with the ALTER NICKNAME statement. Instead, you must DROP and re-CREATE the nickname.

REMOTE_OBJECT
Specifies the name of the Documentum object type associated with the nickname. The name can be any Documentum object type or registered table. In the case of a registered table, it should be prefixed by the table owner’s name. If the registered table belongs to the Docbase owner, dm_dbo can be used for the owner name. This option is required.

Using ALTER NICKNAME to change the value of the REMOTE_OBJECT option will result in errors if the structure of the new object is not similar to that of the original object.

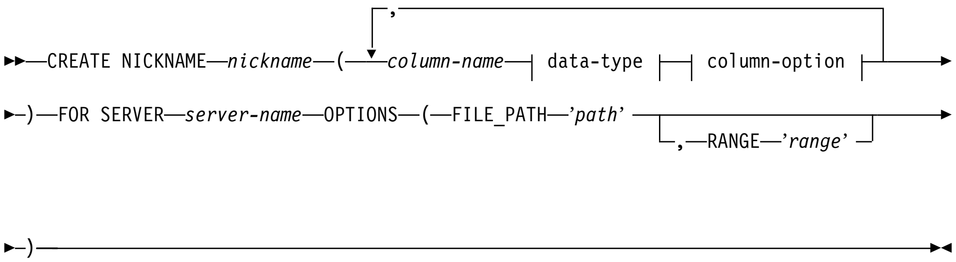
Related tasks:

- “Registering nicknames for Documentum data sources” on page 165

Related reference:

- “CREATE NICKNAME statement” in the *SQL Reference, Volume 2*
- “CREATE NICKNAME statement - Example for Documentum wrapper” on page 170

CREATE NICKNAME statement syntax - Excel wrapper



data-type:

```
├──┬─┬─INTEGER─┬─────────────┬──────────────────────────────────────────────────┤
   │ └─INT─────┘             │
   ├─FLOAT─────────────┬─────┤
   │                   └─(─integer─)─┘
   ├─VARCHAR─(─integer─)─────┤
   └─DATE────────────────────┘
```

column-option:

```
├──┬──────────┬────────────────────────────────────────────────────────────────┤
   └─NOT NULL─┘
```

Where:

FOR SERVER
: Identifies the server that you registered in the associated CREATE SERVER statement. This server is used to access the Excel spreadsheet. Specify the server name.

The following list describes the CREATE NICKNAME options for Excel:

FILE_PATH
: Specifies the fully qualified directory path and file name of the Excel spreadsheet that you want to access.

  Data types must be consistent within each column and the column data types must be described correctly during the register nickname process.

  The Excel wrappers can only access the primary spreadsheet within an Excel workbook.

  Blank cells in the spreadsheet are interpreted as NULL.

  Up to 10 consecutive blank rows can exist in the spreadsheet and 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 spreadsheet. However, these columns must be registered and described as valid fields even if they will not be used.

  The database codepage must match the file's character set; otherwise, you could get unexpected results.

RANGE
: Specifies a range of cells to be used in the data source. This option is not required.

Any syntax or semantic error in the range option value results in an SQL1882E message. Errors might include:

- The top left and bottom right indicators are not oriented correctly. An incorrect orientation is one in which the top-left cell indicator is either below or to the right of the bottom-right cell indicator.
- 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.

Here is an example of the RANGE nickname option:

```
CREATE NICKNAME excel2
(c1  VARCHAR (10),
c2 VARCHAR (10),
c3  VARCHAR (10),
c4  VARCHAR (10)
)  FOR SERVER excel_server
OPTIONS (FILE_PATH 'C:\My Documents\test2.xls',
 RANGE 'B2:E5');
```

In this example, **B2** represents the top left of a cell range, and **E5** represents the bottom right of the cell range. The letter *B* in the B2 designation is the column designation. The number *2* in the B2 representation is the row number.

The bottom right designation can be omitted from the range. In this case, the bottom right valid row is used. If the top left value is omitted, then the value is taken as *A1*. If the range specifies more rows than are actually in the spreadsheet, then the actual number of rows is used.

Related tasks:

- "Registering nicknames for Excel data sources" on page 194

Related reference:

- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*

CREATE NICKNAME statement syntax - Extended Search wrapper

```
                                  ┌─,───────────────────────────────┐
                                  ▼                                 │
►►──CREATE NICKNAME──nickname──(──┴─column-name──┤ data-type ├────┴──)──────────►
```

```
►─FOR SERVER──server-name──OPTIONS──(──APPLICATIONID──'application-id' ──,─────────────►

►──┬──────────────────────────────────────────────┬──────────────────────────────────────►
   │                ┌─;────────────────┐          │
   └─CATEGORY──' ───▼─category-name────┴──' ──,───┘

►──┬──────────────────────────────────────────────┬──┬──────────────────────────────────────┬──►
   │                ┌─;──────────────┐            │  │                 ┌─'NO'──┐            │
   └─DATASOURCE──' ─▼─source-name────┴──' ──,─────┘  └─VERTICAL_TABLE──┼───────┼──,─────────┘
                                                                       └─'YES'─┘

►──┬────────────────────────────────────┬──┬────────────────────────────────────┬──────────►
   │          ┌─'30'──────┐             │  │          ┌─'50'──────┐             │
   └─TIMEOUT──┼───────────┼──,──────────┘  └─MAXHITS──┼───────────┼──,──────────┘
              └─'timeout'─┘                           └─'results'─┘

►──┬──────────────────────────────────────────────┬──┬────────────────────────────┬───────►
   │               ┌─'50'────────────┐            │  │            ┌─'A'─┐         │
   └─TOTALMAXHITS──┼─────────────────┼──,─────────┘  └─SORTORDER──┼─────┼──,──────┘
                   └─'total-results'─┘                            └─'D'─┘

►──┬──────────────────────────────────┬──)────────────────────────────────────────────────►◄
   │            ┌─'DOC_RANK'───┐      │
   └─SORTFIELD──┼──────────────┼──────┘
                └─'field-name'─┘
```

data-type:

```
├──┬─SMALLINT──────────────────────┬────────────────────────────────────────────────────┤
   ├─┬─INTEGER─┬───────────────────┤
   │ └─INT─────┘                   │
   ├─DOUBLE──┬───────────┬─────────┤
   │         └─PRECISION─┘         │
   ├─VARCHAR──(──integer──)────────┤
   ├─DECIMAL──(──integer──)────────┤
   └─DATE──────────────────────────┘
```

NICKNAME
Specifies a unique name for this Extended Search nickname table. This

name must be distinct from all other nicknames in the schema for which it is being defined. This parameter is required.

column-name
Specifies one or more user-defined column names. The column name must match the name of a native or mapped field that is defined in the Extended Search configuration database. This parameter is optional.

data-type
Specifies the SQL data type of the named column. This data type must correspond to the data type that is defined for this field in the Extended Search configuration database. For example, to search a field in an Extended Search data source that has a String data type, define a VARCHAR column for this field in the nickname table. If you specify a *column-name*, this parameter is required.

FOR SERVER
Specifies the name of a previously registered server definition that was created for the Extended Search server that you want to search. This parameter is required.

APPLICATIONID
Specifies the name of the Extended Search application that you want to search. This name must exist in the Extended Search configuration database. This parameter is required.

CATEGORY
Specifies one or more Extended Search categories that you want to search. If you omit this option, you must specify at least one data source name. To specify multiple categories, delimit the category names with a semicolon. For example:

```
CATEGORY 'LotusNotes;MSAccess;LDAP'
```

DATASOURCE
Specifies one or more Extended Search data sources that you want to search. If you omit this option, you must specify at least one category name. To specify multiple data sources, delimit the data source names with a semicolon. For example:

```
DATASOURCE 'AltaVista;Google!;CNN'
```

VERTICAL_TABLE
Specifies the presentation format for search results. If you specify YES, Extended Search returns all fields that are configured as returnable, not just the user-defined columns. The wrapper stores the results in the nickname table as a vertical list of column names. The default value is NO.

TIMEOUT
An INTEGER that specifies the number of seconds to wait for a response from a server before the request times out. This option is optional. The default value is 30.

MAXHITS
: An INTEGER that specifies the maximum number of results that can be returned from each source that is being searched. This option is optional. The default value is 50.

TOTALMAXHITS
: An INTEGER that specifies the maximum number of results that can be returned from all the sources that are being searched. The wrapper combines these results into a single result set. This option is optional. The default value is 50.

SORTORDER
: Specifies a sort order for the return of search results, either ascending (A) or descending (D). The default value is A.

SORTFIELD
: Specifies the name of a field on which search results should be sorted. The default value, DOC_RANK, is a field that Extended Search uses to weigh the relevancy of a result document. If you specify a different field name, be sure that name exists in the sources that you search.

Related concepts:

- "Extended Search nicknames" on page 290
- "Extended Search vertical tables" on page 292

Related tasks:

- "Registering nicknames for Extended Search data sources" on page 297

Related reference:

- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*
- "Extended Search wrapper - Example queries" on page 303

CREATE NICKNAME statement options - Entrez wrapper

The following list describes the CREATE NICKNAME options for Entrez:

REMOTE_OBJECT
: Specifies the name of the Entrez object type 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.

PARENT
: Specified only for a child nickname whose parent has been 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.

Related tasks:

- "Registering nicknames for Entrez data sources" on page 266

Related reference:

- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*

CREATE NICKNAME statement syntax - Table-structured file wrapper

The syntax for the CREATE NICKNAME statement is:

```
                                        ┌─,──────────────────────────────────┐
                                        ▼                                    │
►►──CREATE NICKNAME──nickname──(────column-name──┤ column–information ├──┴──)──────────►

►─FOR SERVER──server-name──OPTIONS──(──┬────────────────────┬──────────────────────────►
                                       └─FILE_PATH──'path'─┘


►──┬──────────────────────────────┬──┬─────────────────────┬──────────────────────────►
   └─,──COLUMN_DELIMITER──'delimiter'─┘  └─,──SORTED──┬─'Y'─┬─┘
                                                      └─'N'─┘


►──┬─────────────────────────────────────────┬────────────────────────────────────────►
   │   (1)                                   │
   └─────────,──KEY_COLUMN──'key-column-name'─┘


►──┬────────────────────────────────────────┬──)───────────────────────────────────────►◄
   │   (1)                                  │
   └─────────,──VALIDATE_DATA_FILE──┬─'Y'─┬─┘
                                    └─'N'─┘
```

column–information:

```
├──┤ data-type ├──┤ column-option ├──┤ nickname–column–options ├──────────────────────┤
```

data-type:

```
├──┬─SMALLINT──────────────────────────────────────────┬────────────────────────────────┤
   ├─┬─INTEGER─┬───────────────────────────────────────┤
   │ └─INT─────┘                                       │
   ├─FLOAT──┬─────────────────┬────────────────────────┤
   │        └─(──integer──)──┘                         │
   ├─REAL──────────────────────────────────────────────┤
   ├─DOUBLE──┬─────────────┬───────────────────────────┤
   │         └─PRECISION──┘                            │
   ├─┬─DECIMAL─┬─┬──────────────────────────────────┬──┤
   │ ├─DEC─────┤ └─(──integer──┬────────────┬──)──┘   │
   │ ├─NUMERIC─┤               └─,──integer─┘          │
   │ └─NUM─────┘                                       │
   ├─┬─CHARACTER─┬─┬─────────────────┬─────────────────┤
   │ └─CHAR──────┘ └─(──integer──)──┘                  │
   └─VARCHAR──(──integer──)────────────────────────────┘
```

column-option:

```
├──┬──────────┬───────────────────────────────────────────────────────────────┤
   └─NOT NULL─┘
```

nickname–column–options:

```
├──OPTIONS──(──DOCUMENT──'FILE' ──)──────────────────────────────────────────┤
```

Notes:

1 Not allowed for unsorted files. Optional for sorted files.

nickname
: A unique nickname for the table-structured file to be accessed. It must be distinct from all other nicknames, tables, and views in the schema in which it is being registered.

column-name
: A unique name given to each field in the table-structured file. Follow each column name with its data type. Only columns of type CHAR, VARCHAR, SMALLINT, INTEGER, FLOAT, DOUBLE, REAL, and DECIMAL are supported.

SMALLINT
: For a small integer.

INTEGER or INT
: For a large integer.

FLOAT(*integer*)
: For a single or double precision floating-point number, depending on the value of *integer*. The value of *integer* must be in the range 1 through 53. The values 1 through 24 indicate single precision and the values 25 through 53 indicate double precision.

REAL For single precision floating-point.

DOUBLE or DOUBLE PRECISION
For double precision floating-point.

FLOAT
For double precision floating-point.

DECIMAL(*precision-integer, scale-integer***) or DEC(***precision-integer, scale-integer***)**
For a decimal number.

The first integer is the precision of the number; that is, the total number of digits. This value can range from 1 to 31.

The second integer is the scale of the number; that is, the number of digits to the right of the decimal point. This value can range from 0 to the precision of the number.

If precision and scale are not specified, the default values of 5,0 are used.

The words **NUMERIC** and **NUM** can be used as synonyms for **DECIMAL** and **DEC**.

CHARACTER(*integer***) or CHAR(***integer***) or CHARACTER or CHAR**
For a fixed-length character string of length *integer*, which can range from 1 to 254. If the length specification is omitted, a length of 1 character is assumed.

VARCHAR(*integer***)**
For a varying-length character string of maximum length *integer*, which can range from 1 to 32672.

NOT NULL
Prevents the column from containing null values.

The wrapper does not enforce the NOT NULL constraint, but DB2 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, DB2 will issue a SQL0407N error stating that you can't 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.

FOR SERVER
Identifies the server you registered using the CREATE SERVER statement. This server will be used to access the table-structured file.

FILE_PATH
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 should be specified. If the FILE_PATH nickname option is specified then no DOCUMENT nickname column option can be specified.

SORTED

Specifies whether the data source file is sorted or unsorted. This option accepts either 'Y', 'y', 'n', or 'N'. It has a default value of '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, it is recommended you set VALIDATE_DATA_FILE to 'Y'.

COLUMN_DELIMITER

The delimiter used to separate columns of the table-structured file, enclosed in single quotation marks. Only single character delimiters are allowed. If no column delimiter is defined, the column delimiter defaults to the 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.

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 defined in the CREATE NICKNAME statement. 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. It is recommended that the key column be designated not nullable by adding the NOT NULL option to its definition in the nickname statement.

This option is case-sensitive. However, DB2 folds column names to upper case unless the column is defined with double quotes.

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', enclosed in single quotation marks. The check is done once at registration time. If this option is not specified, then no validation takes place. This option is not allowed if the DOCUMENT nickname column option is used for the file path.

DOCUMENT

Specifies the kind of table-structured file. Currently, this wrapper only supports FILE for this option. Only one column can be specified with the DOCUMENT option per nickname. The column associated with the DOCUMENT option has to be of data type VARCHAR or CHAR.

Using the DOCUMENT nickname column option, instead of the FILE_PATH nickname option, implies that the file corresponding to this nickname will be supplied during query execution. If the DOCUMENT option has the ″FILE″ value, it means that what will be supplied during query execution is the full path of the file whose schema matches the nickname definition for this nickname. The following CREATE NICKNAME example illustrates the use of 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 file_server
```

The following query, specifying the location of the table-structured file in the WHERE clause, can now be run against the customers nickname:

```
SELECT name, address, id FROM customers
WHERE doc='/home/db2user/Customers.txt'
```

Related tasks:

- “Registering the server for table-structured files” on page 148

Related reference:

- “CREATE NICKNAME statement” in the *SQL Reference, Volume 2*
- “CREATE NICKNAME statement - Examples for table-structured file wrapper” on page 150

CREATE NICKNAME statement syntax - XML wrapper

```
►►──CREATE NICKNAME──nickname──(──column-name───┤ Column structure ├───)──────────►

►─FOR SERVER──server-name──OPTIONS──(──┬─────────────────────────────┬──────────────►
                                        ├─FILE_PATH──'path' ──,───────┤
                                        └─DIRECTORY_PATH──'path' ──,──┘

►─XPATH──'xpath_expression' ──┤ Nickname parameters ├──)──────────────────────────►◄
```

Column structure:

```
├──┤ Data type options ├──┬──────────┬──┤ Nickname column options ├──────────────┤
                          └─NOT NULL─┘
```

Data type options:

```
├──┬─SMALLINT─────────────────────────────────────────┬────────────────────────┤
   ├─┬─INTEGER─┬──────────────────────────────────────┤
   │ └─INT─────┘                                      │
   ├─REAL─────────────────────────────────────────────┤
   ├─DOUBLE──┬───────────┬────────────────────────────┤
   │         └─PRECISION─┘                            │
   ├─┬─DECIMAL─┬──┬────────────────────────────────┬──┤
   │ ├─DEC─────┤  └─(──integer──┬─────────────┬──)─┘  │
   │ ├─NUMERIC─┤                └─,──integer──┘       │
   │ └─NUM─────┘                                      │
   ├─┬─CHARACTER─┬──┬────────────────┬────────────────┤
   │ └─CHAR──────┘  └─(──integer──)──┘                │
   ├─VARCHAR──(──integer──)───────────────────────────┤
   └─DATE─────────────────────────────────────────────┘
```

Nickname column options:

```
├──OPTIONS──(──┬─DOCUMENT──┬─'FILE' ──────┬──┬──)──────────────────────────────┤
               │           ├─'DIRECTORY' ─┤  │
               │           ├─'URI' ───────┤  │
               │           └─'COLUMN' ────┘  │
               ├─XPATH──'xpath_expression' ──┤
               ├─PRIMARY_KEY──'YES' ─────────┤
               └─FOREIGN_KEY──'parent_nickname' ─┘
```

Nickname parameters:

```
├──┬───────────────────────────────┬──┬──────────────────────────────┬──────────────►
   │              ┌─'NO' ──┐       │  │             ┌─'NO' ──┐       │
   └─STREAMING────┴─'YES' ─┴──,────┘  └─VALIDATE────┴─'YES' ─┴──,────┘

►──┬────────────────────────────────────┬──┬──────────────────────────────┬────────►
   └─INSTANCE_PARSE_TIME──'value' ──,───┘  └─XPATH_EVAL_TIME──'value' ──,─┘
```

```
►─┬──────────────────┬─────────────────────────────────────────────────┤
  └─NEXT_TIME─'value'─┘
```

Nickname parameters and options:

FILE_PATH
: Specifies the file path of the XML document. If you specify this nickname option, do not specify a DOCUMENT column. This FILE_PATH option is accepted only for the root nickname (the nickname that identifies the elements at the top level of the XML document).

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 those files with a .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 DIRECTORY_PATH option is accepted only for the root nickname (the nickname that identifies the elements at the top level of the XML document).

XPATH
: Specifies an XPath expression that identifies the XML elements that represent 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.

  Do not specify a namespace prefix in an XPath expression. The XML wrapper does not support namespaces.

Nickname column options:

DOCUMENT
: 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 that contains an XML document. 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 located in XML files that reside under the specified directory path. The XML wrapper uses only those files with a .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.

COLUMN
Specifies that the XML document is stored in a relational column.

XPATH
Specifies the XPath expression in the XML document that contains the data that corresponds to this column. The XML 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.

Do not specify a namespace prefix in an XPath expression. The XML wrapper does not support namespaces.

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 column's value cannot be retrieved in a SELECT query, and the XPATH option must not be specified for this column. The column can be used only to join parent nicknames and child nicknames.

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 column that is designated with this option holds a key that is generated by the wrapper. The column's value cannot be retrieved in a SELECT query, and 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 to be lowercase or mixed case by enclosing it in quotation marks under the corresponding CREATE NICKNAME statement, then when you refer to this nickname in the FOREIGN_KEY clause, you must specify the nickname in uppercase.

Nickname parameters:

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 XML wrapper then parses and processes the XML source data fragment by fragment, reducing total memory use. 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). The default streaming value is NO.

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.

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 default value is NO.

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.

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. The default value is 7 milliseconds.

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. The default value is 1 millisecond.

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 nonroot nicknames. The default value is 1 millisecond.

Usage notes:

If you use the DATE data type option, the dates in your XML source document must have the following format: CCYY-MM-DD. For example, if the date is 17 November 2002, the date must be specified as 2002-11-17 in the XML source document. If a date has any other format, you will receive an error message.

Do not set both the STREAMING parameter and the VALIDATE parameter to YES. The XML wrapper validates an entire XML source document and does not validate source document fragments. If you set both parameters to YES, you will receive an error message.

Related tasks:

- "Registering nicknames for XML data sources" on page 243

Related reference:

- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*
- "CREATE NICKNAME statement - Examples for XML wrapper" on page 244

CREATE SERVER statement arguments - BLAST wrapper

Arguments:

TYPE
Determines the type of BLAST search performed using the given server. This argument is required. It must be set to one of the following values: blastn, blastp, blastx, tblastn, tblastx.

VERSION
Specifies the version of the server that you are using. It should be set to the version of blastall that you are running. This argument is required.

WRAPPER
Specifies the name of the wrapper that you registered using the CREATE WRAPPER statement. This argument is required.

Options:

NODE
Specifies the host name of the system on which the BLAST daemon process is running. This option is required.

DAEMON_PORT
Specifies the port number on which the daemon will listen for BLAST job requests. The port number must be the same number specified in the daemon_port option of the daemon configuration file. The default is 4007. This option is optional.

Related tasks:

- "Registering the server for a BLAST data source" on page 217

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*

CREATE SERVER statement arguments and options - Documentum wrapper

Arguments associated with the CREATE SERVER statement for Documentum are:

TYPE
Specifies the type of the data source. For Documentum, the type is DCTM. This argument is required.

VERSION
Specifies the version of the data source. For EDMS98, the value is ’3’. For 4i, the value is ’4’. This argument is required.

WRAPPER
Specifies the name of the wrapper associated with this server. This argument is required.

Options associated with the CREATE SERVER statement for Documentum are:

CONTENT_DIR
Specifies the name of the locally-accessible root directory for storing content files retrieved by the GET_FILE, GET_FILE_DEL, GET_RENDITION, and GET_RENDITION_DEL pseudo columns. It must be writable by all users who can use these pseudo columns. Its default value is /tmp. This option is optional.

NODE
Specifies the actual name of the Documentum Docbase. This option is required.

OS_TYPE
Specifies the Docbase server’s operating system. Valid values are AIX, SOLARIS, and WINDOWS. This option is required.

RDBMS_TYPE
Specifies the RDBMS used by the Docbase. Valid values are DB2, INFORMIX, ORACLE, SQLSERVER or SYBASE. This option is required.

TRANSACTIONS
Specifies the server transaction mode. The valid values are:

- NONE — no transactions are enabled.
- QUERY — transactions are enabled only for Dctm_Query methods.
- ALL — transactions are enabled for the Dctm_Query method. ALL has the same function as QUERY in this release.

The default is QUERY. This option is optional.

Related tasks:

- “Registering the server for Documentum data sources” on page 164

Related reference:

- “CREATE SERVER statement” in the *SQL Reference, Volume 2*

CREATE SERVER statement arguments - Entrez wrapper

Arguments for the CREATE SERVER statement for Entrez are:

TYPE Specifies the type of the data source. The acceptable values for server type are `PubMed` and `Nucleotide`. These are case-insensitive.

VERSION
Specifies the version of the NCBI XML schema that you are using. This argument is optional. If the version of the server is not specified, the default is `1.0`.

WRAPPER
Specifies the name of the wrapper that you registered by using the CREATE WRAPPER statement.

Related tasks:

- "Registering the server for an Entrez data source" on page 265

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*

CREATE SERVER statement arguments - Excel wrapper

Arguments associated with the CREATE SERVER statement for Excel are:

WRAPPER
Specifies the name of the wrapper that you registered in the associated CREATE WRAPPER statement. This argument is required.

Related tasks:

- "Registering the server for an Excel data source" on page 194

Related reference:

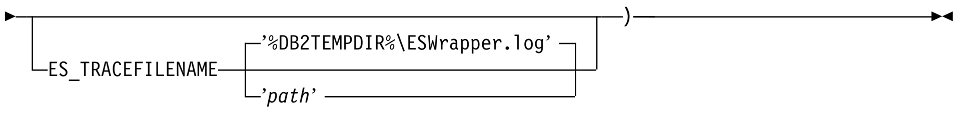
- "CREATE SERVER statement" in the *SQL Reference, Volume 2*

CREATE SERVER statement syntax - Extended Search wrapper

```
►►──CREATE SERVER──server-name──WRAPPER──wrapper-name──OPTIONS──(──────────────►

►─ES_HOST──'host-name' ──,──┬────────────────────────────┬──────────────────────►
                            └─ES_PORT──'port-number' ──,─┘

►──┬────────────────────────────────┬──┬──────────────────────────────────────┬──►
   │              ┌─'OFF' ─┐        │  │                   ┌──────┐           │
   └─ES_TRACING───┴─'ON' ──┴──,─────┘  └─ES_TRACELEVEL──' ─▼─┬─C─┬─┴─' ──,────┘
                                                             ├─N─┤
                                                             ├─W─┤
                                                             └─I─┘
```



SERVER
Specifies a unique name for this server definition. This parameter is required.

WRAPPER
Specifies the name of a previously registered Extended Search wrapper that you want to use with this server definition. This parameter is required.

ES_HOST
Specifies the fully qualified host name or IP address of the Extended Search server that you want to search. This option is required.

ES_PORT
Specifies the port number where this Extended Search server listens for requests. If you omit this option, the default value is `6001`.

ES_TRACING
Specifies whether tracing should be enabled for error messages, warning messages, and informational messages that are produced by the remote Extended Search server. The default value, `OFF`, means that no trace messages will be logged.

ES_TRACELEVEL
If tracing is enabled, this option specifies the types of messages that will be written to the log file. The default value, `C`, logs only critical messages. You can enable and disable the following trace levels independently:

`C` – Critical error messages
`N` – Noncritical messages
`W` – Warning messages
`I` – Informational messages

For example:

```
ES_TRACELEVEL 'W'
ES_TRACELEVEL 'CN'
ES_TRACELEVEL 'CNWI'
```

ES_TRACEFILENAME
If tracing is enabled, this option specifies the name of a directory and file where messages will be written. If you omit this option, the default value is the ESWrapper.log file in your DB2 temp directory (`%DB2TEMPDIR%\ESWrapper.log` or `%DB2TEMPDIR%/ESWrapper.log`).

Related tasks:

- "Registering the server for Extended Search data sources" on page 296

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*

CREATE USER MAPPING statement options - Documentum wrapper

Option definitions:

REMOTE_AUTHID
: Authorization identifier for you at the remote server.

REMOTE_PASSWORD
: Password for you at the remote server.

REMOTE_DOMAIN
: Windows networking domain for you at the remote server. Valid only for Windows platforms.

Related tasks:

- "Mapping users (Documentum wrapper)" on page 164

Related reference:

- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*

CREATE USER MAPPING statement syntax - Extended Search wrapper

```
►►──CREATE USER MAPPING FOR──authorization-name──SERVER──server-name──OPTIONS──────────►

►─(──REMOTE_AUTHID──'user-id' ──,──REMOTE_PASSWORD──'password' ──)──────────────────►◄
```

FOR
: Specifies the user ID of a DB2 user that you want to authorize to access Extended Search data sources. This parameter is required.

SERVER
: Specifies the name of a previously registered server definition that was created for the Extended Search server that the user wants to search. This parameter is required.

REMOTE_AUTHID
: Specifies a user ID that allows this DB2 user to access Extended Search data sources. This remote ID must be in the format that is expected by the data source that is being searched. This option is required.

REMOTE_PASSWORD
: Specifies the password for this remote ID. This option is required.

Related tasks:

- “Registering user mappings for Extended Search data sources” on page 298

Related reference:

- “CREATE USER MAPPING statement” in the *SQL Reference, Volume 2*

CREATE WRAPPER statement syntax - Extended Search wrapper

```
►►──CREATE WRAPPER──wrapper-name──LIBRARY──'library-name' ──────────────────►◄
```

WRAPPER
: Specifies a unique name for this Extended Search wrapper.

LIBRARY
: Specifies one of the following platform-dependent library names:
  - Windows: `db2uies.dll`
  - AIX: `libdb2uies.a`

Related tasks:

- “Registering the Extended Search wrapper” on page 295

Related reference:

- “CREATE WRAPPER statement” in the *SQL Reference, Volume 2*

Appendix A. 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 for Linux, UNIX, and Windows database. There are several unique views which contain information pertinent to a federated system, such as the SYSCAT.WRAPPERS view.

As noted in the DB2 for Linux, UNIX, and Windows Version 6 and Version 7 SQL Reference manuals, the DB2 Version 8 SYSCAT views are now 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 65. Catalog views typically used with a federated system

Catalog views	**Description**
SYSCAT.COLUMNS	Contains column information about the data source objects (tables and views) that you have created nicknames for.
SYSCAT.COLOPTIONS	Contains information about column option values that you have set for a nickname.
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.ROUTINES	Contains local DB2 user-defined functions, or function templates. Function templates are used to map to a data source function.
SYSCAT.INDEXES	Contains index specifications for data source objects.

Table 65. Catalog views typically used with a federated system (continued)

Catalog views	Description
SYSCAT.REVTYPEMAPPINGS	Contains reverse data type mappings. The mapping is from local DB2 data types to data source data types. These mappings are only used with remote (transparent) tables.
SYSCAT.SERVEROPTIONS	Contains information about server option values that you set with a server definition.
SYSCAT.SERVERS	Contains server definitions that you create for data source servers.
SYSCAT.TABLES	Contains information about each local DB2 table, federated view, and nickname that you create.
SYSCAT.TYPEMAPPINGS	Contains forward data type mappings. The mapping is to local DB2 data types from data source data types. These mappings are used when you query a data source using a DB2 SQL statement.
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 66. 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.

Table 66. Federated updatable global catalog views (continued)

Catalog views	**Description**
SYSSTAT.FUNCTIONS	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.INDEXES	Contains statistical information about each index specification for data source objects.
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.

Appendix B. Server options for federated systems

Server options are used with the CREATE SERVER statement 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.

Nonrelational wrappers have additional, very specific server options that are documented in the data source configuration information.

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 67. Available server options

Data Source	COLLATING_SEQUENCE	COMM_RATE	CONNECTSTRING	CPU_RATIO	DATEFORMAT	DB2_MAXIMAL_PUSHDOWN	DBNAME	FOLD_ID	FOLD_PW	IFILE	IGNORE_UDT	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 for iSeries	X	X		X		X	X	X	X				X	X				X		X				X
DB2 for z/OS and OS/390	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
DB2 for Linux, UNIX, and Windows	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				
Microsoft SQL Server	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	
OLE DB	X		X			X																		
Oracle	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			
Teradata	X	X		X		X							X	X		X				X				

The following table describes each server option and lists the valid and default settings.

Table 68. Server options and their settings

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 1x1023. Values may be expressed in any valid double notation, for example 123E10, 123, or 1.21E4. Values may be expressed as whole numbers or floating point numbers.	’2’
CONNECTSTRING	Specifies initialization properties needed to connect to an OLE DB provider.	None
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 1x1023. Values may be expressed in any valid double notation, for example 123E10, 123, or 1.21E4. Values may be expressed as whole numbers or floating point numbers. 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 .5 indicates that the DB2 federated CPU speed is 50% slower than the data source CPUO speed. A setting of 2 indicates that the DB2 federated CPU speed is twice as fast as the data source CPU speed.	’1.0’

Table 68. Server options and their settings (continued)

Option	Description and valid settings	Default setting
DATEFORMAT (See note 5 at the end of this table)	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
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 is might be used. Queries that result in a Cartesian product will be processed by the federated database, and will not be pushed down. ’N’ The query optimizer chooses an access plan based on cost.	’N’
DBNAME	Name of the data source database that you want the federated server to access. For DB2, this value corresponds to a specific database within an instance or, with DB2 for z/OS or OS/390, the database LOCATION value. Does not apply to Oracle data sources because Oracle instances contain only one database.	None.

Table 68. Server options and their settings (continued)

Option	Description and valid settings	Default setting
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 none of these settings are used, the federated server tries to send the user ID to the data source in uppercase. If the user ID fails, the server tries sending it 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 none of these settings are used, the federated server tries to send the password to the data source in uppercase. If the password fails, the server tries sending it in lowercase.	None.
IFILE	Specifies the path and name of the Sybase Open Client interfaces file. On Windows NT federated servers, the default is %DB2PATH%\interfaces. On UNIX federated servers, the default path and name value is $DB2INSTANCE/sqllib/interfaces.	None.
IGNORE_UDT	Specifies whether user-defined types (UDTs) on data sources accessed using the CTLIB or DBLIB wrapper should be used by the federated server. Valid values are: 'Y' Ignore user-defined specifications of UDTs. 'N' Do not ignore user-defined specifications of UDTs.	'N'

Table 68. 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.	’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 may be expressed in any valid double notation, for example 123E10, 123, or 1.21E4. 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’
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 will roll back insert, update, or delete transactions if an error occurs and the data source does not enforce application savepoint statements. SQL error code (SQL1476) 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. The default values are the same as for TIMEOUT.	’0’

Table 68. Server options and their settings (continued)

Option	Description and valid settings	Default setting
NODE	Name by which a data source is defined as an instance to its RDBMS.	None.
PACKET_SIZE	Specifies the packet size of the Sybase interfaces file in bytes. If the data source does not support the specified packet size, the connection will fail. Increasing the packet size when each record is very large (for example, when inserting rows into large tables) significantly increases performance. The byte size is a numeric value.	
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’
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.	’Y’
TIMEFORMAT (See note 5 at the end of this table)	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
TIMESTAMPFORMAT (See note 5 at the end of this table)	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

Table 68. Server options and their settings (continued)

Option	Description and valid settings	Default setting
TIMEOUT	Specifies the number of seconds the DB2 federated server will wait for a response from Sybase Open Client for any SQL statement. The value of *seconds* is a positive whole number in DB2 Universal Database's integer range. The timeout value that you specify depends on which wrapper you are using. The default behavior of the TIMEOUT option for the Sybase wrappers is 0, which causes DB2 to wait indefinitely for a response.	'0'
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. 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. The only valid setting for DB2 family data sources is 'Y'. 'Y' Yes, trailing blanks are absent from these VARCHAR columns. This data source has blank-padded character comparison semantics that are different than the federated server. Character comparison operations will be processed on the federated server and will not pushed down to the data source. 'N' No, trailing blanks are present in these VARCHAR columns. This data source has blank-padded character comparison semantics that are similar to the semantics on the federated server. Character comparison operations can be pushed down to the data source for processing.	'N'

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 may 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 four times before successfully passing data source authentication).
5. This option is used only when the value of SERVER_TYPE is GENERIC. This option is ignored for all other values of SERVER_TYPE

Related concepts:

- "Server characteristics affecting pushdown opportunities" in the *Federated Systems Guide*
- "Server characteristics affecting global optimization" in the *Federated Systems Guide*

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*

Appendix C. User mapping options for federated systems

User mapping options provide authorization and accounting string information for user mappings between the federated server and a data source. These options can be used with any data source that supports user ID and password authorization.

These options are used with the CREATE USER MAPPING statement.

Table 69. User mapping options and their settings

Option	Valid settings	Default setting
ACCOUNTING_STRING	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
REMOTE_AUTHID	Indicates the authorization ID used at the data source. Valid settings include any string of length 255 or less.	The ID used to connect to the federated database
REMOTE_DOMAIN	Indicates the Windows NT domain used to authenticate users connecting to a Documentum data source. Valid settings include any valid Windows NT domain name. If this option is not specified, the Documentum data source will authenticate using the default authentication domain for that database.	None
REMOTE_PASSWORD	Indicates the authorization password used at the data source. Valid settings include any string of length 32 or less. If this option is not specified, then no password is used to connect to the data source server. If the server requires a password to connect, then the connection will fail.	None

Related concepts:

- "DB2 Connect and DRDA" in the *DB2 Connect User's Guide*
- "DRDA and data access" in the *DB2 Connect User's Guide*

Appendix D. Column options for federated systems

You can specify column information in the CREATE NICKNAME or ALTER NICKNAME statements using parameters called *column options*.

The primary purpose of column options is to provide information about nickname columns to the SQL Compiler. Setting column options for one or more columns to 'Y' allows the SQL Compiler to consider additional pushdown possibilities for predicates that perform evaluation operation. This assists the Compiler in reaching global optimization. You can specify any of these values in either uppercase or lowercase characters.

Attention: Nonrelational wrappers allow additional column options.

Table 70. Column options and their settings

Option	Valid settings		Default setting
NUMERIC_STRING	'Y'	Yes, this column contains strings of numeric characters '0', '1', '2', ... '9'. It does not contain blanks. IMPORTANT: If this column contains only numeric strings followed by trailing blanks, it is inadvisable to specify 'Y'.	'N'
	'N'	No, this column is either not a numeric string column or is a numeric string column that contains blanks.	
	By setting 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. This option is helpful when the collating sequence of a data source is different from DB2. Columns marked with this option will not be excluded from remote evaluation because of a different collating sequence.		

Table 70. Column options and their settings (continued)

Option	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. 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’ Yes, trailing blanks are absent from this VARCHAR column. This data source has blank-padded character comparison semantics that are different than the federated server. Character comparison operations will be processed on the federated server and will not pushed down to the data source. ‘N’ No, trailing blanks are present in this VARCHAR column. This data source has blank-padded character comparison semantics that are similar to the semantics on the federated server. Character comparison operations can be pushed down to the data source for processing.	‘N‘

Related concepts:

- “Pushdown analysis” in the *Federated Systems Guide*

Related tasks:

- “Global optimization” in the *Federated Systems Guide*

Appendix E. Function mapping options for federated systems

DB2 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.

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. 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 71. 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'

Table 71. Function mapping options and their settings (continued)

Option	Valid settings	Default setting
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

Appendix F. Valid server types in SQL statements

Server types indicate what kind of data source the server will represent. Server types vary by vendor, purpose, and operating system. Supported values depend on the wrapper being used.

You need to specify a valid server type in the CREATE SERVER statement.

CTLIB wrapper

Sybase data sources supported by Sybase CTLIB client software.

Server Type	Data Source
SYBASE	Sybase

DBLIB wrapper

Sybase or Microsoft SQL Server data sources supported by DBLIB client software.

Server Type	Data Source
SYBASE	Sybase

DJXMSSQL3 wrapper

Microsoft SQL Server data sources supported by ODBC 3.0 (or higher) driver.

Server Type	Data Source
MSSQLSERVER	Microsoft SQL Server

DRDA wrapper

DB2 Family

Table 72. DB2 for Linux, UNIX, and Windows

Server Type	Data Source
DB2/UDB	IBM DB2 Universal Database
DB2/6000	IBM DB2 for AIX

Table 72. DB2 for Linux, UNIX, and Windows (continued)

Server Type	Data Source
DB2/AIX	IBM DB2 for AIX
DB2/HPUX	IBM DB2 for HP-UX V1.2
DB2/HP	IBM DB2 for HP-UX
DB2/NT	IBM DB2 for Windows NT
DB2/EEE	IBM DB2 Enterprise-Extended Edition
DB2/CS	IBM DB2 for Common Server
DB2/SUN	IBM DB2 for Solaris V1 and V1.2
DB2/PE	IBM DB2 for Personal Edition
DB2/2	IBM DB2 for OS/2
DB2/LINUX	IBM DB2 for Linux
DB2/PTX	IBM DB2 for NUMA-Q
DB2/SCO	IBM DB2 for SCO Unixware

Table 73. DB2 for iSeries (and AS/400)

Server Type	Data Source
DB2/400	IBM DB2 for iSeries and AS/400

Table 74. DB2 for z/OS and OS/390

Server Type	Data Source
DB2/ZOS	IBM DB2 for z/OS
DB2/390	IBM DB2 for OS/390
DB2/MVS	IBM DB2 for MVS

Table 75. DB2 Server for VM and VSE

Server Type	Data Source
DB2/VM	IBM DB2 for VM
DB2/VSE	IBM DB2 for VSE
SQL/DS	IBM SQL/DS

Informix wrapper

Informix data sources supported by Informix Client SDK software.

Server Type	Data Source
INFORMIX	Informix

MSSQLODBC3 wrapper

Microsoft SQL Server data sources supported by DataDirect Connect ODBC 3.6 driver.

Server Type	Data Source
MSSQLSERVER	Microsoft SQL Server

NET8 wrapper

Oracle data sources supported by Oracle NET8 client software.

Server Type	Data Source
ORACLE	Oracle Version 8.0. or later

ODBC wrapper

ODBC data sources supported by the ODBC 3.x driver.

Server Type	Data Source
ODBC	ODBC

OLE DB wrapper

OLE DB providers compliant with Microsoft OLE DB 2.0 or later.

Server Type	Data Source
none required	Any OLE DB provider

SQLNET wrapper

Oracle data sources supported by Oracle SQL*Net V1 or V2 client software.

Server Type	**Data Source**
ORACLE	Oracle V7.3. or later

Teradata wrapper

Teradata data sources supported by the Teradata V2R3 and V2R4 client software.

Server Type	**Data Source**
TERADATA	Teradata

Appendix G. Default forward data type mappings

When a nickname is created for a data source object, DB2 for Linux, UNIX, and Windows populates the global catalog with information about the table.

This information includes the *remote* data type for each column, and the corresponding DB2 for Linux, UNIX, and Windows data type. The DB2 for Linux, UNIX, and Windows data type is referred to as the *local* data type.

The federated database uses data type mappings to determine which DB2 for Linux, UNIX, and Windows data type should be defined for the column of a data source object.

The data types at the data source must map to corresponding DB2 for Linux, UNIX, and Windows data types so that the federated server can retrieve data from data sources. For most data sources, the default type mappings are in the wrappers. 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.

DB2 for Linux, UNIX, and Windows federated servers do not support mappings for these local data types:
- DATALINK
- user-defined types

There are two kinds of mappings between data source data types and federated database data types: 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.

The following tables show the default forward mappings between DB2 for Linux, UNIX, and Windows data types and data source data types.

These mappings are valid with all the supported versions, unless otherwise noted.

Important: For all default forward data types mapping from a data source to DB2 for Linux, UNIX, and Windows, the DB2 federated schema is SYSIBM.

DB2 for z/OS and OS/390 data sources

Table 76. DB2 for z/OS and OS/390 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
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 iSeries data sources

Table 77. 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 Server for VM and VSE data sources

Table 78. 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 Linux, UNIX, and Windows data sources

Table 79. DB2 for Linux, UNIX, and Windows 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
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
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	0	Y

Table 79. DB2 for Linux, UNIX, and Windows 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
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	0	N

Informix data sources

Table 80. 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	32	-	-	-	-	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	-	-	-

Table 80. 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
REAL	-	-	-	-	-	-	REAL	4	-	-
SERIAL	-	-	-	-	-	-	INTEGER	4	-	-
SERIAL8	-	-	-	-	-	-	BIGINT	-	-	-
SMALLFLOAT	-	-	-	-	-	-	REAL	4	-	-
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 qualifer 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 contstants 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

Table 81. Microsoft SQL Server 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
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	-	-	-
DUMMY65 [1]	1	38	-84	127	-	-	DOUBLE	-	-	-
DUMMY2000 [3]	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

Table 81. Microsoft SQL Server 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
nchar	128	4000	-	-	-	-	VARCHAR	-	-	N
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 [2]	-	-	-	-	-	-	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_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	-	-

Table 81. Microsoft SQL Server 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
SQL_GUID [2]	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
sysname	-	-	-	-	-	-	VARCHAR	30		Y
sysname	1	254	-	-	-	-	CHAR	-	-	N
text	-	-	-	-	-	-	CLOB	-	-	N
timestamp	-	-	-	-	-	-	VARCHAR	8		Y
tinyint	-	-	-	-	-	-	SMALLINT	2	-	-
uniqueidentifier [2]	1	4000	-	-	Y	-	VARCHAR	16	-	Y
varbinary	1	8000	-	-	-	-	VARCHAR	-	-	Y
varchar	1	8000	-	-	-	-	VARCHAR	-	-	N

Notes:

1. This type mapping is valid only with Microsoft SQL Server Version 6.5.
2. This type mapping is valid only with Microsoft SQL Server Version 7 and Version 2000.
3. This type mapping is valid only with Windows 2000 operating systems.

ODBC data sources

Table 82. 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

Table 82. ODBC 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
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

Table 83. Oracle NET8 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	0	0	0	0	-	\0	BLOB	2147483647	0	Y
CHAR	1	254	0	0	-	\0	CHAR	0	0	N
CHAR	255	32672	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	63	0	0	-	\0	REAL	0	0	N
FLOAT	64	126	0	0	-	\0	DOUBLE	0	0	N
MLSLABEL	0	0	0	0	-	\0	VARCHAR	255	0	N
NUMBER	1	38	-84	127	-	\0	DOUBLE	0	0	N
NUMBER	1	31	0	31	-	>=	DECIMAL	0	0	N
NUMBER	1	5	0	0	-	\0	SMALLINT	0	0	N
NUMBER	6	10	0	0	-	\0	INTEGER	0	0	N
RAW	1	254	0	0	-	\0	CHAR	0	0	Y
RAW	255	32672	0	0	-	\0	VARCHAR	0	0	Y
ROWID	0	0	0	NULL	-	\0	CHAR	18	0	N
VARCHAR2	1	32672	0	0	-	\0	VARCHAR	0	0	N

Oracle SQLNET data sources

Table 84. Oracle SQLNET 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
CHAR	1	254	0	0	-	\0	CHAR	0	0	N
CHAR	255	32672	0	0	-	\0	VARCHAR	0	0	N
DATE	0	0	0	0	-	\0	TIMESTAMP	0	0	N
FLOAT	1	63	0	0	-	\0	REAL	0	0	N
FLOAT	64	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
MLSLABEL	0	0	0	0	-	\0	VARCHAR	255	0	N
NUMBER	1	38	-84	127	-	\0	DOUBLE	0	0	N
NUMBER	1	31	0	31	-	>=	DECIMAL	0	0	N
NUMBER	1	5	0	0	-	\0	SMALLINT	0	0	N
NUMBER	6	10	0	0	-	\0	INTEGER	0	0	N
RAW	1	254	0	0	-	\0	CHAR	0	0	Y
RAW	255	32672	0	0	-	\0	VARCHAR	0	0	Y
ROWID	0	0	0	NULL	-	\0	CHAR	18	0	N
VARCHAR2	1	32672	0	0	-	\0	VARCHAR	0	0	N

Sybase data sources

Table 85. Sybase CTLIB and DBLIB 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
binary	1	254	-	-	-	-	CHAR	-	-	Y
binary	255	16384	-	-	-	-	VARCHAR	-	-	Y
bit	-	-	-	-	-	-	SMALLINT	-	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	16384	-	-	-	-	VARCHAR	-	-	N
datetime	-	-	-	-	-	-	TIMESTAMP	-	-	-
datetimn	-	-	-	-	-	-	TIMESTAMP	-	-	-
decimal	1	31	0	31	-	-	DECIMAL	-	-	-
decimal	32	38	0	38	-	-	DOUBLE	-	-	-
decimaln	1	31	0	31	-	-	DECIMAL	-	-	-
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
numeric	1	31	0	31	-	-	DECIMAL	-	-	-

Table 85. Sybase CTLIB and DBLIB 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
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	-	-	-
varbinary	1	16384	-	-	-	-	VARCHAR	-	-	Y
varchar	1	16384	-	-	-	-	VARCHAR	-	-	N

Teradata data sources

Table 86. 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	-	-	-

Table 86. Teradata 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
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
VARCHAR	32673	64000	-	-	-	-	CLOB	-	-	-
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	-	-
VARGRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-

Related concepts:

- "Forward and reverse data type mappings" in the *Federated Systems Guide*

Appendix H. Default reverse data type mappings

There are two kinds of mappings between data source data types and federated database data types: 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.

For most data sources, the default type mappings are in the wrappers. 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 DB2 federated database, the definition includes a reverse type mapping. The mapping is from a *local* DB2 for Linux, UNIX, and Windows 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.

DB2 for Linux, UNIX, and Windows federated servers do not support mappings for these local data types: 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 7 and to INT8 in Informix Version 8.

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 DB2 for Linux, UNIX, and Windows local data types and remote data source data types.

These mappings are valid with all the supported versions, unless otherwise noted.

DB2 for z/OS and OS/390 data sources

Table 87. DB2 for z/OS and OS/390 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

DB2 for iSeries data sources

Table 88. 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_OPERATORS	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

Table 89. 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	-	-	-
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 Linux, UNIX, and Windows data sources

Table 90. DB2 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
VARGRAPHIC	-	-		-	-	-	-	VARGRAPHIC	-		-	-

Informix data sources

Table 91. Informix 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]	-	-	-	-	-	-	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	-	-	-
LONG VARCHAR	-	32700	-	-	N	-	TEXT	-	-	-
LONG VARCHAR	-	32700	-	-	Y	-	BYTE	-	-	-
REAL	-	4	-	-	-	-	SMALLFLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	INTEGER	-	-	-
TIME	-	3	-	-	-	-	DATETIME	6	10	-
TIMESTAMP	-	10	-	-	-	-	DATETIME	0	15	-
VARCHAR	1	254	-	-	N	-	VARCHAR	-	-	-
VARCHAR	255	32672	-	-	N	-	TEXT	-	-	-
VARCHAR	-	-	-	-	Y	-	BYTE	-	-	-
VARCHAR [2]	255	2048	-	-	N	-	LVARCHAR	-	-	-
VARCHAR [2]	2049	32672	-	-	N	-	TEXT	-	-	-

Table 91. Informix 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

Notes:

1. This type mapping is valid only with Informix server Version 7 (or lower).
2. This type mapping is valid only with Informix server Version 8 (or higher).

For the Informix DATETIME data type, the DB2 UNIX and Windows federated server uses the Informix high-level qualifer 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 contstants 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

Table 92. 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
BLOB	-	-	-	-	-	-	image	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
DATE	-	4	-	-	-	-	datetime	-	-	-
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	-	-	-

Oracle SQLNET data sources

Table 93. Oracle SQLNET 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	0	2147483647	0	0	Y	\0	LONG RAW	0	0	Y
CHARACTER	1	254	0	0	N	\0	CHAR	0	0	N
CHARACTER	0	0	0	0	Y	\0	RAW	0	0	Y
CLOB	0	2147483647	0	0	N	\0	LONG	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
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	0	10	0	0	N	\0	DATE	0	0	N
VARCHAR	1	2000	0	0	Y	\0	RAW	0	0	Y
VARCHAR	1	4000	0	0	N	\0	VARCHAR2	0	0	N

Note: The DB2 for Linux, UNIX, and Windows BIGINT data type is not available for transparent DDL. You cannot specify the BIGINT data type in a CREATE TABLE statement when creating a remote Oracle table.

Oracle NET8 data sources

Table 94. Oracle NET8 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	0	2147483647	0	0	Y	\0	BLOB	0	0	Y
CHARACTER	1	254	0	0	N	\0	CHAR	0	0	N
CHARACTER	0	0	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
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	0	10	0	0	N	\0	DATE	0	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: The DB2 for Linux, UNIX, and Windows BIGINT data type is not available for transparent DDL. You cannot specify the BIGINT data type in a CREATE TABLE statement when creating a remote Oracle table.

Sybase data sources

Table 95. Sybase CTLIB ans DBLIB 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	-	-	-	-	-	-	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	255	-	-	N	-	varchar	-	-	-
VARCHAR	256	32672	-	-	N	-	text	-	-	-
VARCHAR	1	255	-	-	Y	-	varbinary	-	-	-
VARCHAR	256	32672	-	-	Y	-	image	-	-	-

Teradata data sources

Table 96. 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	FEDERATED_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	-	-	-
DOUBLE	-	-	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	FLOAT	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARBYTE	-	-	-
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

Table 96. 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	FEDERATED_BIT_DATA

Notes:

1. The Teradata VARBYTE data type can contain only the specified length (1 to 64000) of a DB2 BLOB data type.
2. The Teradata VARCHAR data type can contain only the specified length (1 to 64000) of a DB2 CLOB data type.
3. The Teradata VARGRAPHIC data type can contain only the specified length (1 to 32000) of a DB2 DBCLOB data type.

Related concepts:

- "Forward and reverse data type mappings" in the *Federated Systems Guide*

DB2 Information Integrator technical documentation

The following topics describe how to:

- Access books and release information, including printing and ordering books
- Access topics by using the DB2 Information Integrator Information Center or the DB2 HTML Documentation CD

Accessing books and release information

DB2 Information Integrator technical information is available in the following formats:

- Books (PDF and printed). A description of each book in the DB2 Information Integrator library is available from the IBM Publications Center at www.ibm.com/shop/publications/order.
- An Information Center (HTML format).
- Help for DB2 Tools (HTML format).

DB2 Information Integrator books

The DB2 Information Integrator PDF Documentation CD contains PDF files of the books in the DB2 Information Integrator library and the DB2 Universal Database library. The structure of the DB2 Information Integrator PDF Documentation CD is:

- On Windows operating systems: *x*:\doc\*%L*
- On UNIX operating systems: /*cdrom*/doc/*%L*/

where:

- *x* represents the Windows CD-ROM drive letter
- *cdrom* refers to the UNIX mount point of the CD-ROM
- *%L* is the locale of the documentation that you want to use, for example, en_US

Language	**Locale**	**Identifier**	**Language**	**Locale**	**Identifier**
Arabic	ar_AA	w	Japanese	ja_JP	j
Brazilian Portuguese	pt_BR	b	Korean	ko_KR	k
Bulgarian	bg_BG	u	Norwegian	no_NO	n
Croatian	hr_HR	9	Polish	pl_PL	p
Czech	cs_CZ	x	Portuguese	pt_PT	v
Danish	da_DK	d	Romanian	ro_RO	8
Dutch	nl_NL	q	Russian	ru_RU	r
English	en_US	e	Simplified Chinese	zh_CN	c

Finnish	fi_FI	y	Slovakian	sk_SK	7
French	fr_FR	f	Slovenian	sl_SI	l
German	de_DE	g	Spanish	es_ES	z
Greek	el_GR	a	Swedish	sv_SE	s
Hungarian	hu_HU	h	Traditional Chinese	zh_TW	t
Italian	it_IT	i	Turkish	tr_TR	m

The character in the sixth position of each PDF file name indicates the language version of a book (see the following table). For example, the file name `iiyig`**`e`**`80` identifies the English version of the *IBM DB2 Information Integrator Installation Guide*, and the file name `iiyig`**`g`**`80` identifies the German version of the same book.

The books in the following table are available for DB2 Information Integrator.

Table 97. DB2 Information Integrator documentation

Name	**Form number**	**Install category**	**PDF file name**
IBM DB2 Information Integrator Solutions Guide	SC18-7037	getting_started	iiyis**x**80
IBM DB2 Information Integrator Installation Guide	GC18-7036	getting_started	iiyig**x**80
IBM DB2 Information Integrator Migration Guide	SC18-7360	getting_started	iiymg**x**80
IBM DB2 Information Integrator Federated Systems Guide	SC18-7364	admin	iiyfp**x**80
IBM DB2 Information Integrator Data Source Configuration Guide	Available online only	optional	iiyls**x**80
IBM DB2 Information Integrator Developer's Guide	SC18-7359	ad	iiyfs**x**80

Printing books from PDF files

You can print DB2 Information Integrator books from the PDF files on the DB2 Information Integrator PDF Documentation CD. You can use Adobe Acrobat Reader to print the entire book, a range of pages, or specific pages.

Prerequisites:

Ensure that you have Adobe Acrobat Reader. It is available from the Adobe Web site at www.adobe.com.

Procedure:

To print a DB2 Information Integrator book from a PDF file:

1. Insert the DB2 Information Integrator PDF Documentation CD. On UNIX operating systems, mount the CD.
2. Start the Adobe Acrobat Reader.
3. Open the PDF file from one of the following locations:
   - On Windows operating systems: *x*:\doc\*%L*
   - On UNIX operating systems: /*cdrom*/doc/*%L*/

   where:
   - *x* represents the Windows CD-ROM drive letter
   - *cdrom* refers to the UNIX mount point of the CD-ROM
   - *%L* is the locale of the documentation that you want to print, for example, en_US
4. Click **File –> Print**.
5. In the Print window, specify whether to print all of the pages, the current pages, or a range of pages.
6. Click **OK**.

Ordering printed books

You can get printed manuals by ordering the documentation package (doc pack) for your DB2 Information Integrator product from your IBM reseller. The doc packs are a subset of the manuals from the DB2 Information Integrator library. These doc packs are designed to help you to get started using the DB2 Information Integrator product that you purchased.

The manuals in the doc packs are the same as those on the DB2 Information Integrator PDF Documentation CD that was provided with your DB2 Information Integrator product.

You can also use one of the following methods to order individual books:

- Contact your IBM marketing representative or authorized dealer. To find a local IBM representative, check the IBM Worldwide Directory of Contacts at www.ibm.com/planetwide.
- Phone 1-800-879-2755 in the United States or 1-800-IBM-4YOU in Canada.
- Go to the IBM Publications Center at www.ibm.com/shop/publications/order.

Release notes

The release notes provide additional information that is specific to your product's release and fix pack level. They also provide summaries of the documentation updates that are incorporated in each release and fix pack.

The release notes are available on the product CD-ROM:

- On Windows operating systems: *x*:\doc\*%L*
- On UNIX operating systems: /*cdrom*/doc/*%L*/

where:

- *x* represents the Windows CD-ROM drive letter
- *cdrom* refers to the UNIX mount point of the CD-ROM
- *%L* is the locale of the documentation that you want to use; for example, en_US

Table 98. Release notes

Name	Location	File name
DB2 Information Integrator Release Notes	Available from the product CD-ROM in the following formats: • Text • HTML Also available in the DB2 Information Integrator Information Center	• ReleaseNotes.txt • ReleaseNotes.html
DB2 Information Integrator Installation Requirements	Available on product CD-ROM in the following formats: • Text • HTML Also available from the DB2 Information Integration Installation Launchpad	• Prereqs.txt • Prereqs.html

To view the ASCII file on UNIX-based systems, see the Release.Notes file. This file is in the *DB2DIR*/Readme/*%L* directory, where *%L* represents the locale name, and *DB2DIR* represents:

- /usr/opt/db2_08_01 on AIX
- /opt/IBM/db2/V8.1 on all other UNIX operating systems

FixPaks for DB2 Information Integrator documentation

IBM might periodically make documentation fix packs available. You can use documentation fix packs to update the information that you installed from the DB2 HTML Documentation CD when new information becomes available.

Documentation FixPaks are cumulative. For example, if you install the documentation for Version 8.1 and then apply FixPak 2, you will get documentation updates for FixPak 1 and FixPak 2.

When you install documentation fix packs, your HTML documentation will be more up-to-date than either the printed or online PDF manuals for your product.

Accessing topics using the DB2 Information Integrator Information Center or the DB2 HTML Documentation CD

The DB2 Information Integrator Information Center gives you access to the information that you need to take advantage of DB2 Information Integrator in your business.

Features of the DB2 Information Integrator Information Center

The DB2 Information Integrator Information Center has the following features:

Integrated navigation tree
: Locate any topic in the DB2 Information Integrator library from a single navigation tree. The DB2 Information Integrator library includes the following types of information:

  Tasks Key tasks you can perform using DB2 Information Integrator. Tasks provide step-by-step instructions on how to complete a goal.

  Concepts
  Key concepts for DB2 Information Integrator. Concepts provide an overview of a subject.

  Reference
  Reference topics provide detailed information about a subject, including statement syntax, command syntax, message help, requirements, keywords, commands, and APIs.

Search
: Search all of the topics on your workstation by clicking **Search** in the navigation toolbar.

Master index
: Access the information in topics and tools help from one master index. The index contains entries from the entire DB2 library.

Master glossary
The master glossary defines terms that are used in the DB2 Information Integrator library.

Regularly updated documentation
Keep your topics up to date by downloading updated HTML topics.

Finding topics in the DB2 Information Integrator Information Center

The following major elements comprise the DB2 Information Integrator Information Center:

Navigation tree
The navigation tree is located in the left frame of the browser window. The tree expands and collapses to show and hide topic links, the glossary, and the master index in the DB2 Information Integrator Information Center.

Navigation toolbar
The navigation toolbar is located in the top right frame of the browser window. Use the pushbuttons in the navigation toolbar to search the DB2 Information Integrator Information Center, hide the navigation tree, and find the currently displayed topic in the navigation tree.

Content frame
The content frame is located in the bottom right frame of the browser window. When you click a link in the navigation tree, click on a search result, or follow a link from another topic or from the master index, the content frame displays the appropriate topic.

Prerequisites:

To access the DB2 Information Integrator Information Center from a browser, you must use one of the following browsers:

- Microsoft Explorer Version 5 or later
- Netscape Navigator Version 6.1 or later

Restrictions:

The DB2 Information Integrator Information Center contains only those sets of topics that you chose to install from the DB2 HTML Documentation CD. If your Web browser returns a `File not found` error when you follow a link to a topic, install one or more additional sets of topics from the DB2 HTML Documentation CD.

Procedure:

To find a topic by searching with keywords:

1. In the navigation toolbar, click **Search**.
2. In the top text entry field of the Search window, enter two or more terms related to your area of interest, then click **Search**. The **Results** field displays a list of topics. The topics that most closely match your search string are at the top of the list.

   Enter more terms to increase the precision of your query and reduce the number of topics that are returned from your query.
3. In the **Results** field, click the title of the topic that you want to read. The topic is displayed in the content frame.

To find a topic in the navigation tree:

1. In the navigation tree, click the book icon next to the category of topics in which you are interested. A list of subcategories is displayed under the icon.
2. Continue to click the book icons until you find the category that contains the topics in which you are interested. Categories that link to topics show the category title as a link when you move the cursor over the category title. A page icon is used in the navigation tree to identify topics.
3. Click the topic link. The topic is displayed in the content frame.

To find a topic by using the master index:

1. In the navigation tree, click **Index**. The index expands to display a list of links arranged in alphabetical order.
2. In the navigation tree, click the first character of the subject that you are looking for. A list of entries with that initial character is displayed in the content frame.
3. If a book icon is displayed, there are multiple index entries for a subject. Click the book icon that corresponds to the subject in which you are interested. A list of topics is displayed below the term that you clicked.
4. Click on the title of the topic that meets your needs. The topic is displayed in the content frame.

Using the DB2 HTML Documentation

This topic describes how to install, view, and copy the documentation on the DB2 HTML Documentation CD, and how to update the documentation after you install it.

Installing the DB2 HTML documentation

The installation directory for the DB2 HTML Documentation CD differs for each category of information:

htmlcdpath/doc/htmlcd/*%L*/*category*

htmlcdpath
: The directory where the DB2 HTML Documentation CD is installed.

%L The locale of the documentation that you want to use, for example, en_US.

category
The category identifier, for example, getting_started for the installation information.

Viewing technical documentation directly from the DB2 HTML Documentation CD

All of the HTML topics that you can install from the DB2 HTML Documentation CD can also be read directly from the CD. Therefore, you can view the documentation without installing it.

Restrictions:

You must install the DB2 product to view the online help and release notes.

Procedure:

To view the HTML documentation from the DB2 HTML Documentation CD:

- For Windows operating systems:
  1. Insert the DB2 HTML Documentation CD.
  2. Start your Web browser and open the following file:
     ```
     x:\Program Files\sqllib\doc\htmlcd\%L\index.htm
     ```
     where *x* represents the CD drive, and *%L* is the locale of the documentation that you want to use, for example, en_US for English.
- For UNIX operating systems:
  1. Mount the DB2 HTML Documentation CD.
  2. Start your Web browser and open the following file:
     ```
     /cdrom/Program Files/sqllib/doc/htmlcd/%L/index.htm
     ```
     where *cdrom* represents where the CD is mounted, and *%L* is the locale of the documentation that you want to use, for example, en_US for English.

Copying files from the DB2 HTML Documentation CD to a Web Server

The entire DB2 information library is available on the DB2 HTML Documentation CD, so you can copy the library on a Web server for easier access.

Procedure:

Copy files from the DB2 HTML Documentation CD to the appropriate path on your Web server (the default path is shown):

- For Windows operating systems: *x*:\Program Files\IBM\sqllib\doc\htmlcd\*%L**.*
- For UNIX operating systems: /*cdrom*/Program Files/IBM/sqllib/doc/htmlcd/*%L*

where:

- *x* represents the Windows CD-ROM drive letter
- *cdrom* refers to UNIX mount point of the CD-ROM
- *%L* is the locale of the documentation that you want to use, for example, en_US

Updating the HTML documentation on your computer

When IBM makes updates available, you can update the HTML files that you installed from the DB2 HTML Documentation CD by either:

- Using the Information Center (if you have the DB2 administration graphical user interface tools installed)
- Downloading and applying a DB2 HTML Documentation FixPak

This procedure does not update the DB2 code.

Documentation FixPaks are cumulative. For example, if you install the documentation for Version 8.1 and then apply FixPak 2, you will get documentation updates for FixPak 1 and FixPak 2.

Prerequisites:

Ensure that your computer has access to the Internet, because the updater downloads the latest documentation fix pack from the IBM server, if required. To connect to the Internet, you might need to supply your proxy information.

Procedure:

To use the Information Center to update your local HTML documentation:

1. Start the DB2 Information Center:
   - From the graphical administration tools, click the **Information Center** icon in the toolbar.
   - At the command line, enter db2ic.
2. Click **Information Center** –> **Update Local Documentation** to start the update.

   If a documentation update is available, it is downloaded and applied.

To manually download and apply the documentation update:

1. Open the DB2 support page in your Web browser at www.ibm.com/software/data/db2/udb/winos2unix/support

2. Click **DB2 Version 8** and find the documentation fix pack link for your operating system.
3. Determine if your local DB2 documentation is out of date by comparing the documentation fix pack level to the documentation level that you have installed.
4. If a more recent version of the documentation is available, then download the fix pack for your operating system. There is one fix pack for all Windows operating systems, and one fix pack for all UNIX operating systems.
5. Apply the fix pack:
   - For Windows operating systems: The documentation fix pack is a self-extracting zip file. Download the documentation fix pack to an empty directory and unzip it. Run the **setup** command in that directory to install the documentation fix pack.
   - For UNIX operating systems: The documentation fix pack is a compressed tar.Z file. Uncompress and untar the file to create a directory named `delta_install`. Run the script installdocfix inside that directory to install the documentation fix pack.

Searching the DB2 documentation

To search the DB2 documentation, use Netscape Version 6.1 (or later) or Microsoft Internet Explorer Version 5 (or later). Ensure that your browser's Java support is enabled.

A search window opens when you click the search icon in the navigation toolbar of the DB2 Information Integrator Information Center in a browser. If you are using the search function for the first time, it might take a minute or so for the search window to load.

Restrictions:

The following restrictions apply when you use the documentation search:
- Boolean searches are not supported. The Boolean search qualifiers *and* and *or* are ignored in a search. For example, the following searches produce the same results:
  – servlets *and* beans
  – servlets *or* beans
- Wildcard searches are not supported. A search on *java** will look for the literal string *java** and will not, for example, find *javadoc*.

In general, you will get better search results if you search for phrases instead of single words.

Procedure:

To search the DB2 documentation:

1. In the navigation toolbar, click **Search**.
2. In the top text entry field of the Search window, enter two or more terms related to your area of interest, then click **Search**. The **Results** field displays a list of topics that are ranked by accuracy.

   Enter more terms to increase the precision of your query and reduce the number of topics that are returned by your query.
3. In the **Results** field, click the title of the topic you want to read. The topic displays in the content frame.

When you perform a search, the first result is automatically loaded into your browser frame. To view the contents of other search results, click on the result in results lists.

Troubleshooting DB2 documentation search with Netscape 4.x

Most search problems are related to the Java support provided by Web browsers. This task describes possible solutions.

Procedure:

A common problem with Netscape 4.x involves a missing or misplaced security class. Try the following solution, especially if you see the following line in the browser's Java console:

```
Cannot find class  java/security/InvalidParameterException
```

Copy the following file from the DB2 HTML Documentation CD to the `java\classes\java\security\` directory within the directory where your Netscape browser is installed. You might have to create the `java\security\` subdirectory structure.

- On Windows operating systems:

  x:`Program Files\sqllib\doc\htmlcd\`*%L*`\InvalidParameterException.class`
- On UNIX operating systems:

  /*cdrom*`/Program Files/sqllib/doc/htmlcd/`*%L*
  `/InvalidParameterException.class`

where:

- *x* represents the Windows CD-ROM drive letter
- *cdrom* refers to the UNIX mount point of the CD-ROM
- *%L* is the locale of the documentation that you want to use, for example, en_US

If your Netscape browser still fails to display the search input window, try the following actions:

- Stop all instances of Netscape browsers to ensure that there is no Netscape code running on the computer. Then open a new instance of the Netscape browser and start the search again.
- Purge the browser’s cache.
- Try a different version of Netscape, or a different browser.

Accessibility

Users with physical disabilities, such as restricted mobility or limited vision, can use software products successfully by using accessibility features. These are the major accessibility features in DB2 Information Integrator Version 8:

- You can operate all features by using the keyboard instead of the mouse.
- You can customize the size and color of your fonts.
- You can receive either visual or audio alert cues.
- DB2 supports accessibility applications that use the Java™ Accessibility API.
- DB2 documentation is provided in an accessible format.

Keyboard input and navigation

You can operate the DB2 Tools by using only the keyboard. You can use keys or key combinations instead of a mouse to perform most operations.

In UNIX-based systems, the position of the keyboard focus is highlighted. This highlighting indicates which area of the window is active and where your keystrokes will have an effect.

Accessible display

The DB2 Tools have features that enhance the user interface and improve accessibility for users with low vision. These accessibility enhancements include support for customizable font properties.

Font settings

For the DB2 Tools, you can use the Tools Settings notebook to select the color, size, and font for the text in menus and dialog windows.

Nondependence on color

You do not need to distinguish between colors to use any of the functions in this product.

Alternative alert cues

You can specify whether you want to receive alerts through audio or visual cues, using the Tools Settings notebook.

Compatibility with assistive technologies

The DB2 Tools interface supports the Java Accessibility API, enabling the use of screen readers and other assistive technologies that are used by people with disabilities.

Accessible documentation

Documentation for the DB2 family of products is available in HTML format. You can view documentation according to the display preferences set in your browser. You can use screen readers and other assistive technologies.

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 all 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 give 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/region or send inquiries, in writing, to:

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

The following paragraph does not apply to the United Kingdom or any other country/region 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

Trademarks

Index

Contacting IBM

To contact IBM in the United States or Canada, call one of the following numbers:

- For customer service: 1-800-IBM-SERV (1-800-426-7378)
- For DB2 marketing and sales: 1-800-IBM-4YOU (1-800-426-4968)

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

To locate an IBM office in your country or region, see the IBM Directory of Worldwide Contacts on the Web at www.ibm.com/planetwide.

Product information

Information about DB2 Information Integrator is available by telephone or on the Web.

If you live in the United States, you can call one of the following numbers:

- To order products or to obtain general information: 1-800-IBM-CALL (1-800-426-2255)
- To order publications: 1-800-879-2755

On the Web, go to www.ibm.com/software/data/integration/solution. This site contains the latest information on the technical library, ordering books, client downloads, newsgroups, fix packs, news, and links to Web resources.

To locate an IBM office in your country or region, see the IBM Directory of Worldwide Contacts on the Web at www.ibm.com/planetwide.

Comments on the documentation

Your feedback helps IBM to provide quality information. Please send any comments that you have about this book or other DB2 Information Integrator documentation. 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/data/rcf.
- Send your comments by electronic mail (e-mail) to comments@vnet.ibm.com. Be sure to include the name of the product, the

version number of the product, and the name and part number of the book (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).

Printed in U.S.A.

Spine information:

IBM DB2 Information Integrator

Data Source Configuration Guide

Version 8