IBM® DB2 Universal Database™

Federated Systems Guide

Version 8

GC27-1224-00

IBM® DB2 Universal Database™

Federated Systems Guide

Version 8

GC27-1224-00

Before using this information and the product it supports, be sure to read the general information under *Notices.*

You can order IBM publications online or through your local IBM representative.

- To order publications online, go to the IBM Publications Center at www.ibm.com/shop/publications/order
- To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at www.ibm.com/planetwide

To order DB2 publications from DB2 Marketing and Sales in the United States or Canada, call 1-800-IBM-4YOU (426-4968).

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

About this book

This book describes how to plan, configure, and administer a federated system. It includes information about accessing heterogeneous data using DB2 Universal Database for UNIX and Windows, Version 8. Depending on the data sources that you want to access, you might need to install and configure DB2 Relational Connect or DB2 Life Sciences Data Connect.

This book contains:

- An introduction to federated systems, and the concepts and terminology used when discussing DB2 federated systems.
- Hardware, software, and other requirements for setting up a federated system.
- General information about migrating your federated system from DB2 DataJoiner 2.1.1 and DB2 Universal Database for UNIX and Windows, Version 7 to DB2 Version 8. Detailed information about migrating from DB2 DataJoiner 2.1.1 is in the DB2 Universal Database for UNIX and Windows Version 8 release notes.
- Instructions for setting up a DB2 server and database to perform as the federated server and database.
- Steps to configure the federated database to access data sources. This book describes how to configure access to the following data sources:
  - DB2 for UNIX and Windows
  - DB2 for z/OS and OS/390
  - DB2 for iSeries
  - DB2 Server for VM and VSE
  - Informix
  - Oracle
  - Sybase
  - Microsoft SQL Server
  - ODBC
  - OLE DB

  Note: To configure access to BLAST, Documentum, Microsoft Excel, table-structured files, and XML data sources, see the *DB2 Life Sciences Data Connect: Planning, Installation, and Configuration Guide*.
- Suggestions for tuning your federated system for optimum performance.

- Issues you need to consider when developing applications for federated systems.
- A glossary of federated terms.

Who should read this book

This book is intended for system administrators, database administrators, security administrators, and system operators who need to set up, configure, maintain, or use a DB2 federated system. Use this book to implement a federated system to access data from relational and nonrelational data sources. This book can also be used by programmers and other users who require an understanding of the configuration, administration, and use of a federated system.

This book assumes that you are familiar with DB2. You should be familiar with standard database terminology, and have experience with database design and database administration. This book assumes that you are familiar with your own applications and the data sources that you want access.

Conventions and terminology used in this book

Federated terminology:

This book defines the terms that are used when discussing federated systems in the glossary.

DB2 Commands:

This book assumes that DB2 commands are entered in the DB2 Command Line Processor (CLP), unless otherwise specified.

Highlighting Conventions:

This book uses these highlighting conventions:

Boldface type

Indicates commands and graphical user interface controls (such as names of fields, names of push buttons, and menu choices). Boldface type indicates keywords in SQL examples. Boldface type is used to designate notes, restrictions, prerequisites, and recommendations.

`Monospace type`

Indicates text that you type, file names, and code examples.

Italic type

Indicates SQL or command variables that you replace with an appropriate value. Italic type is used to emphasize words, including new term, and indicates document titles.

UPPERCASE TYPE

Indicates the names of DB2 commands and SQL statements, and their keywords. Uppercase is also used for data type names and acronyms.

Prerequisite and related information

If you are migrating your federated system from DB2 for UNIX and Windows Version 7 or DataJoiner Verison 2.1.1 to DB2 for UNIX and Windows Version 8, read the *DB2 Quick Beginnings for Servers* book. It contains information about migrating your instances and databases. Detailed information about migrating from DB2 DataJoiner 2.1.1 is in the DB2 Universal Database for UNIX and Windows release notes.

How to send your comments

Your feedback helps IBM to provide quality information. Please send any comments that you have about this book or other DB2 documentation. You can use any of the following methods to provide comments:

- Send your comments from the Web. You can access the IBM Data Management online readers' comment form at:

  http://www.ibm.com/software/data/rcf
- Send your comments by 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 chapter and section title, a table number, a page number, or a help topic title).

What's new about federated systems in DB2 Version 8?

The federated changes since DB2 Version 7.1 are significant. DB2 Version 8 provides additional support for data sources.

Expanded support:

DB2 Version 8 federated systems support additional data sources and operating systems:

- You can access Informix data sources with DB2 Enterprise Server Edition, DB2 Workgroup Server Edition, or DB2 Personal Edition.
- Through DB2 Relational Connect, you can access Microsoft SQL Server, Sybase, and ODBC data sources.
- A new product, Life Sciences Data Connect, enables you to access BLAST search algorithms, Documentum data files, Microsoft Excel spreadsheets, table-structured files, and XML tagged files.
- DB2 servers that useLinux, HP-UX, and Windows 2000 operating systems can now be federated servers.

Write capability:

A significant enhancement to federated functionality in DB2 Version 8, is the ability to write to a data source. You can now issue INSERT, UPDATE, and DELETE statements on nicknames. This enables DB2 replication to support replication to and from non-DB2 data sources. In addition, you can create remote tables on relational data sources. This feature is sometimes referred to as *transparent DDL*.

New federated DB2 Control Center support:

Use the DB2 Control Center to quickly setup a federated system. You can create the wrappers, supply the server definitions, idenify user mappings, and create nicknames for the data source objects.

Additionally, the DB2 Control Center is the easiest way to create remote tables using transparent DDL. Through the DB2 Cotnrol Center, you can:

- Create a remote table.
- Define a primary key for the new remote table.
- Add a new column to the remote table by selecting from a list of predefined columns or by specifying the attributes for a new column.
- Alter the attributes of existing columns, such as the column name, data type, and length.
- Alter the remote table primary keys.
- Drop the remote table.

Changes to the SYSCAT views:

As noted in the DB2 Universal Database for UNIX and Windows Version 6 and Version 7 SQL Reference manuals, the DB2 Version 8 SYSCAT views are now read-only.

Materialized query table support:

A materialized query table is a summary table that is created based on the result set of a query. Unlike a view, a materialized query table stores the actual data from the result set. You can create a materialized query table on a nickname to provide better performance when accessing data sources. Summary tables can be created in DB2 for UNIX and Windows that reference a combination of nicknames and local tables. These summary tables can be created with the refresh deferred option only.

Part 1. Introduction to federated systems and concepts

Chapter 1. Overview of a federated system

This chapter describes the features of a federated system, defines federated concepts and terminology used throughout the book, provides an overview of tasks needed to set up and configure a federated system, and outlines the ways in which you can interface with a federated system.

Federated systems

A DB2® *federated system* is a special type of distributed database management system (DBMS). A federated system consists of a DB2 instance that operates as a federated server, a database that acts as the federated database, one or more data sources, and clients (users and applications) that access the database and data sources. With a federated system you can send distributed requests to multiple data sources within a single SQL statement. For example, you can join data that is located in a DB2 Universal Database™ table, an Oracle table, and a Sybase view in a single SQL statement.

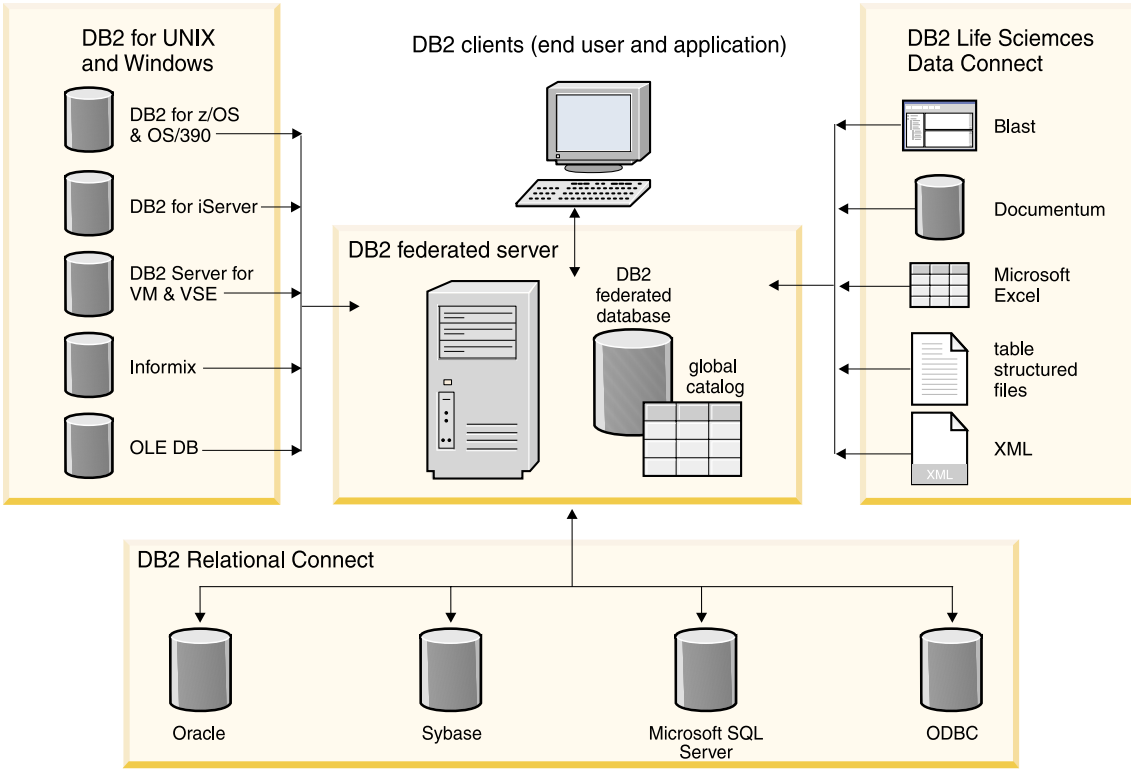

Figure 1. The components of a federated system and the supported data sources

The power of a DB2 federated system is in its ability to:

- Join data from local tables and remote data sources, as if all the data is local.
- Take advantage of the data source processing strengths, by sending distributed requests to the data sources for processing.
- Compensate for SQL limitations at the data source by processing parts of a distributed request at the federated server.

The DB2 server in a federated system is referred to as the *federated server*. Any number of DB2 instances can be configured to function as federated servers. You can use existing DB2 instances as your federated server, or you can create new ones specifically for the federated system.

The DB2 federated instance that manages the federated system is called a *server* because responds to requests from end users and client applications. The federated server often sends parts of the requests it receives to the data sources for processing. A *pushdown* operation is an operation that is processed

remotely. The federated instance is referred to as the *federated server*, even though it acts as a client when it pushes down requests to the data sources.

Like any other application server, the federated server is a database manager instance to which application processes connect and submit requests. However, two main features distinguish it from other application servers:

- A federated server is configured to receive requests that might be partially or entirely intended for data sources. The federated server distributes these requests to the data sources.
- Like other application servers, a federated server uses DRDA® communication protocols (such as SNA and TCP/IP) to communicate with DB2 family instances. However, unlike other application servers, a federated server uses other protocols to communicate with non-DB2 family instances.

Related concepts:

- "Data sources" on page 5
- "The federated database" on page 7
- "The SQL Compiler and the query optimizer" on page 8
- "Compensation" on page 9
- "Pushdown analysis" on page 233

Data sources

Typically, a federated system *data source* is a relational DBMS instance (such as Oracle or Sybase) and one or more databases that are supported by the instance. However, there are other types of data sources (such as life sciences data sources and search algorithms) that you can include in your federated system:

- Spreadsheets, such as Microsoft® Excel.
- Search algorithms, such as BLAST.
- Table-structured files. These type of files have a regular structure that consists of a series of records. Each record contains the same number of fields that are separated by an arbitrary delimiter. Two sequential delimiters represent null values.
- Documentum document management software that includes a repository to store document content, attributes, relationships, versions, renditions, formats, workflow, and security.
- XML tagged files.

In DB2® Universal Database for UNIX® and Windows, the supported data sources are:

Table 1. Supported data source versions and access methods.

Data source	Supported data source versions	Access method	Notes
DB2 Universal Database™ for UNIX and Windows®	6.1, 7.1, 7.2, 8.1	DRDA®	Directly integrated in DB2 Version 8
DB2 Universal Database for z/OS™ and OS/390®	5 with PTF PQ07537 (or later)	DRDA	Directly integrated in DB2 Version 8
DB2 Universal Database for iSeries™	4.2 (or later)	DRDA	Directly integrated in DB2 Version 8
DB2 Server for VM and VSE	3.3 (or later)	DRDA	Directly integrated in DB2 Version 8
Informix™	7, 8, 9	Informix Client SDK	Directly integrated in DB2 Version 8
ODBC		ODBC 3.0 driver.	Requires DB2 Relational Connect
OLE DB		OLE DB 2.0 (or later)	Directly integrated in DB2 Version 8
Oracle	7.x, 8.x, 9.x	SQL*Net or Net8 client software	Requires DB2 Relational Connect
Microsoft SQL Server	6.5, 7.0, 2000	On Windows the Microsoft SQL Server Client ODBC 3.0 (or higher) driver. On UNIX the Data Direct Technologies (formerly MERANT) Connect ODBC 3.6 driver.	Requires DB2 Relational Connect
Sybase	10.0, 11.0, 11.1, 11.5, 11.9, 12.0	Sybase Open Client	Requires DB2 Relational Connect
BLAST	2.1.2	BLAST daemon (supplied with the wrapper)	Requires DB2 Life Sciences Data Connect
Documentum	Documentum server: EDMS 98 (also referred to as version 3) and 4i.	Documentum Client API/Library	Requires DB2 Life Sciences Data Connect

Table 1. Supported data source versions and access methods. (continued)

Data source	Supported data source versions	Access method	Notes
Microsoft Excel	97, 2000	none	Requires DB2 Life Sciences Data Connect
table-structured files		none	Requires DB2 Life Sciences Data Connect
XML	1.0 specification	none	Requires DB2 Life Sciences Data Connect

Data sources are semi-autonomous. For example, the federated server can send queries to Oracle data sources at the same time that Oracle applications can access these data sources. A DB2 federated system does not monopolize or restrict access to the other data sources, beyond integrity and locking constraints.

The federated database

To end users and client applications, data sources appear as a single collective database in DB2. Users and applications interface with the *federated database* managed by the federated server. The federated database contains catalog entries that identify data sources and their characteristics. The federated server consults the information stored in the federated database system catalog and the data source wrapper to determine the best plan for processing SQL statements.

The federated database system catalog contains information about the objects in the federated database and information about objects at the data sources. The catalog in a federated database is called the *global catalog* because it contains information about the entire federated system. DB2® query optimizer uses the information in the global catalog and the data source wrapper to plan the best way to process SQL statements. The information stored in the global catalog includes remote and local information, such as column names, column data types, column default values and index information.

Remote catalog information is the information or name used by the data source. *Local* catalog information is the information or name used by the federated database. For example, suppose a remote table includes a column with the name of *EMPNO*. The global catalog would store the remote column name as *EMPNO*. Unless you designate a different name, the local column

name will be stored as *EMPNO*. You can change the local column name to *Employee_Number*. Users submitting queries which include this column will use *Employee_Number* in their queries instead of *EMPNO*. You use column options to change the local name of data source column.

For relational data sources, the information stored in the global catalog includes both remote and local information. For non-relational data sources, the information stored in the global catalog varies from data source to data source.

To see the data source table information that is stored in the global catalog, query the federated SYSCAT.TABLES, SYSCAT.TABOPTIONS, SYSCAT.COLUMNS, and SYSCAT.COLOPTIONS catalog views.

The federated system processes SQL statements as if the data sources were ordinary relational tables or views within the federated database. This enables the federated system to join relational data with data in non-relational formats. This is true even when the data sources use different SQL dialects, or do not support SQL at all.

The global catalog also includes other information about the data sources. For example, it includes information the federated server uses to connect to the data source and map the federated user authorizations to the data source user authorizations.

Related concepts:

- "Federated systems" on page 3
- "The SQL Compiler and the query optimizer" on page 8
- "Tuning query processing" on page 231

Related reference:

- Appendix A, "Views in the global catalog table containing federated information" on page 281

The SQL Compiler and the query optimizer

To obtain data from data sources, users and applications submit queries in DB2® SQL to the federated database. When a query is submitted, the DB2 SQL Compiler consults information in the global catalog and the data source wrapper to help it process the query. This includes information about connecting to the data source, server attributes, mappings, index information, and processing statistics.

As part of the SQL Compiler process, the *query optimizer* analyzes a query. The Compiler develops alternative strategies, called *access plans*, for processing the query. Access plans might call for the query to be:

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

DB2 evaluates the access plans primarily on the basis of information about the data source capabilities and the data. The wrapper and the global catalog contain this information. DB2 decomposes the query into segments that are called *query fragments*. Typically it is more efficient to *pushdown* a query fragment to a data source, if the data source can process the fragment. However, the query optimizer takes into account other factors such as:

- The amount of data that needs to be processed.
- The processing speed of the data source.
- The amount of data that the fragment will return.
- The communication bandwidth.

The query optimizer generates local and remote access plans for processing a query fragment, based on resource cost. DB2 then chooses the plan it believes will process the query with the least resource cost.

If any of the fragments are to be processed by data sources, DB2 submits these fragments to the data sources. After the data sources process the fragments, the results are retrieved and returned to DB2. If DB2 performed any part of the processing, it combines its results with the results retrieved from the data source. DB2 then returns all results to the client.

Related concepts:

- "Tuning query processing" on page 231
- "Pushdown analysis" on page 233

Related tasks:

- "Global optimization" on page 246

Compensation

The DB2® federated server does not push down a query fragment if the data source cannot process it, or if the federated server can process it faster than the data source can process it. For example, suppose that the SQL dialect of a data source does not support a CUBE grouping in the GROUP BY clause. A query that contains a CUBE grouping and references a table in that data source is submitted to the federated server. DB2 does not pushdown the

CUBE grouping to the data source, but processes the CUBE itself. The ability by DB2 to process SQL that is not supported by a data source is called *compensation*.

The federated server compensates for lack of functionality at the data source in two ways:

- It can ask the data source to use one or more operations that are equivalent to the DB2 function stated in the query. Suppose a data source does not support the cotangent (COT(x)) function, but supports the tangent (TAN(x)) function. DB2 can ask the data source to perform the calculation (1/TAN(x)), which is equivalent to the cotangent (COT(x)) function.
- It can return the set of data to the federated server, and perform the function locally.

Each type of RDBMS supports a subset of the international SQL standard. In addition, some types of RDBMSs support SQL constructs that exceed this standard. An *SQL dialect*, is the totality of SQL that a type of RDBMS supports. If an SQL construct is found in the DB2 SQL dialect, but not in a data source dialect, the federated server can implement this construct on behalf of the data source.

The following examples show the ability of DB2 to compensate for differences in SQL dialects:

- DB2 SQL includes the clause, common-table-expression. In this clause, a name can be specified by which all FROM clauses in a fullselect can reference a result set. The federated server will process a common-table-expression for a data source, even though the SQL dialect used by the data source does not include common-table-expression.
- When connecting to a data source that does not support multiple open cursors within an application, the federated server can simulate this function. The federated server does this by establishing separate, simultaneous connections to the data source. Similarly, the federated server can simulate CURSOR WITH HOLD capability for a data source that does not provide that function.

With compensation, the federated server can support the full DB2 SQL dialect for queries against data sources. Even data sources with weak SQL support or no SQL support. You must use the DB2 SQL dialect with a federated system, except in a pass-through session.

Related concepts:

- "The SQL Compiler and the query optimizer" on page 8
- "Pass-through sessions" on page 11
- "Function mappings and function templates" on page 20

Pass-through sessions

You can submit SQL statements directly to data sources by using a special mode called *pass-through*. You submit SQL statements in the SQL dialect used by the data source. Use a pass-through session when you want to perform an operation that is not possible with the DB2® SQL/API. For example, use a pass-through session to create a procedure, create an index, or perform queries in the native dialect of the data source.

Note: Currently, the data sources that support pass-through, support pass-through using SQL. In the future, it is possible that data sources will support pass-though using a data source language other than SQL.

Similarly, you can use a pass-through session to perform actions that are not supported by SQL, such as certain administrative tasks. However, you cannot use a pass-through session to perform all administrative tasks. For example, you can create or drop tables at the data source, but you cannot start or stop the remote database.

You can use both static and dynamic SQL in a pass-through session.

The federated server provides the following SQL statements to manage pass-through sessions:

SET PASSTHRU
: Opens a pass-through session. When you issue another SET PASSTHRU statement to start a new pass-through session, the current pass-through session is terminated.

SET PASSTHRU RESET
: Terminates the current pass-through session.

GRANT (Server Privileges)
: Grants a user, group, list of authorization IDs, or PUBLIC the authority to initiate pass-through sessions to a specific data source.

REVOKE (Server Privileges)
: Revokes the authority to initiate pass-through sessions.

The following restrictions apply to pass-through sessions:

- You must use the SQL dialect or language commands of the data source — you cannot use the DB2 SQL dialect. As a result, you do not query a nickname, but the data source objects directly.
- When performing UPDATE or DELETE operations in a pass-through session, you cannot use the WHERE CURRENT OF CURSOR condition.

Related concepts:

- "How client applications interact with data sources" on page 255
- "Using pass-through to query data sources directly" on page 277

Related tasks:

- "Using pass-through with Oracle data sources" on page 278
- "Working with nicknames" on page 175

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 system.

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 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.

There are wrappers for each supported data source. Some wrappers have default wrapper names. When you use the default name to create the wrapper, the federated server automatically picks up the data source library associated with the wrapper.

Table 2. Default wrapper names for each data source.

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

Table 2. Default wrapper names for each data source. (continued)

Data source	Default wrapper name(s)
Informix	INFORMIX
Oracle	SQLNet or 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

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 do not support SQL, one of two actions will occur:

  – 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. Data type mappings 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 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. Function mappings 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. Currently there is only one wrapper option, DB2_FENCED. The DB2_FENCED wrapper option indicates if the wrapper is fenced or trusted by DB2. A fenced wrapper operates under some restrictions.

Related concepts:

- "Create the wrapper" on page 88
- "Fast track to configuring your data sources" on page 85

Related reference:

- Appendix B, "Wrapper options for federated systems" on page 285

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 are specified as parameters in the CREATE SERVER and ALTER SERVER statements.

Server options are set to values that persist over successive connections to the data source. These values are stored in the global catalog. For example, the name for the data source on the RDBMS is set in the NODE server option. Some data sources have multiple databases on each instance. For these data source, the name of the database which the federated server connects to is set in the DBNAME server option.

To set a server option value temporarily, use the SET SERVER OPTION statement, This statement overrides the value for the duration of a single connection to the federated database. The overriding value does not get stored in the global catalog.

Related concepts:

- "Supply the server definition" on page 91

Related reference:

- Appendix C, "Server options for federated systems" on page 287

User mappings and user options

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. By default, the federated server attempts to access the data source with the user ID and password that are used to connect to DB2. If the user ID and password are the same between the federated server and the data source, the connection is established. If the user ID and password to access the federated server differs from the user ID and password to access a data source, you must define an association between the two authorizations. Once you define the association, distributed requests can be sent to the data source. This association is called a *user mapping*.

You define and modify user mappings with the CREATE USER MAPPING and ALTER USER MAPPING statements. These statements include parameters, called *user options*, which values related to authorization are assigned to. For example, suppose that a user has the same ID, but different passwords, for the federated database and a data source. For the user to access the data source, it is necessary to map the passwords to one another. You use the CREATE USER MAPPING statement and the user option REMOTE_PASSWORD to map the passwords. Use the ALTER USER MAPPING statement to modify an existing user mapping.

Related concepts:

- "Create the user mappings and test the connection to the data source" on page 94

Related reference:

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

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*.

The following table shows the data source objects you can reference when you create a nickname.

Table 3. Data sources and the objects that you can create a nickname for

Data source	Objects you can reference
DB2® for UNIX® and Windows®	nicknames, summary tables, tables, views
DB2 for z/OS™ and OS/390®	tables, views
DB2 for iSeries™	tables, views
DB2 Server for VM and VSE	tables, views
Informix™	tables, views, synonyms
Microsoft® SQL Server	tables, views
ODBC	tables, views
Oracle	tables, views
Sybase	tables, views
BLAST	FASTA files indexed for BLAST search algorithms
document management software	objects and registered tables in a Documentum Docbase
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

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, it references the data source objects by their nicknames. The nicknames are mapped to specific objects at the data source. The 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 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 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 utility operations (LOAD, REORG, REORGCHK, IMPORT, RUNSTATS, and so on) on nicknames.

Related concepts:

• "Create nicknames for each data source object" on page 96

Related reference:

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

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. 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. The SQL Complier and query optimizer use the metadata to develop better plans for accessing the data.

DB2® treats the object that a nickname references as if it is a 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 only be applied 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 and 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.

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

Related tasks:

- "Working with nicknames" on page 175

Related reference:

- Appendix E, "Column options for federated systems" on page 299

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. 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 non-relational data sources, you must specify data type information in the CREATE NICKNAME statement.

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:

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

When values from a data source column are returned to the federated database, the values conform fully to the DB2 data type that the data source column is mapped to. If this is a default mapping, the values also conform fully to the data source type in the mapping. For example, suppose an Oracle table with a FLOAT column is defined to the federated database. The default mapping of Oracle FLOAT to DB2 DOUBLE automatically applies to that column. Consequently, the values that are returned from the column will conform fully to both FLOAT and DOUBLE.

For some wrappers, you can change the format or length of values that are returned. You do this by changing the DB2 data type that the values must conform to. For example, the Oracle data type DATE is used as a time stamp; the Oracle DATE data type contains century, year, month, day, hour, minute, and second. By default, the Oracle DATE data type maps to the DB2 TIMESTAMP data type. Suppose that several Oracle table columns have a data type of DATE. You want queries of these columns to return only the hour, minute, and second. You can override the default data type mapping so that the Oracle DATE data type maps to the DB2 TIME data type. When Oracle DATE columns are queried, only the time portion of the time stamp values is returned to DB2.

Use the CREATE TYPE MAPPING statement to create:

- A data type mapping that overrides a default data type mapping
- A data type mapping for which there currently is no mapping. For example, 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.

In the CREATE TYPE MAPPING statement, you can specify if the mapping applies each time that you access that data source, or if the mapping applies to a specific server.

Use the ALTER TYPE MAPPING statement to change a type mapping that you originally created with the CREATE TYPE MAPPING statement. The ALTER TYPE MAPPING statement cannot be used to change the default type mappings.

To modify a data type mapping for a specific column of a specific data source object, use the column option parameters in the ALTER NICKNAME statement. This statement enables you to specify data type mappings for individual tables, views, or other data source objects.

If you change a type mapping, nicknames created before the type mapping change do not reflect the new mapping.

Unsupported data types:

DB2 federated servers do not support:

- LONG VARCHAR
- LONG VARGRAPHIC
- DATALINK
- User-defined data types (UDTs) created at the data source

You cannot create a user-defined mapping for these data types. However, you create a nickname for view at the data source that is identical to the table that contains the user-defined data types. The view must 'cast' the user-defined type column to the built-in, or system, type.

A nickname can be created for a remote table that contains LONG VARCHAR columns. However, the results will be mapped to a local DB2 data type that is not LONG VARCHAR.

Related concepts:

- "Modifying wrappers" on page 195

Related tasks:

- "Modifying default data type mappings" on page 208

Related reference:

- "ALTER NICKNAME statement" in the *SQL Reference, Volume 2*
- "CREATE TYPE MAPPING statement" in the *SQL Reference, Volume 2*
- Appendix H, "Default forward data type mappings" on page 307

Function mappings and function templates

For the federated server to recognize a data source function, the function must be mapped against an existing DB2® function. 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. The default function mappings from DB2 for UNIX® and Windows® functions to DB2 for z/OS™ functions are in the DRDA® wrapper. The default function mappings from DB2 for UNIX and Windows functions to Sybase functions are in the CTLIB and DBLIB wrappers, and so forth.

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 and a new user-defined function becomes available at the data source. Function mappings are also used when a DB2 counterpart function does not exist, you must create one on the DB2 federated server that meets the following requirements:

- If the data source function has input parameters:
  - The DB2 counterpart function must have the same number of input parameters that the data source function has.
  - The data types of the input parameters for the DB2 counterpart function must be compatible with the corresponding data types of the input parameters for data source function.
- If the data source function has no input parameters:
  - The DB2 counterpart function cannot have any input parameters.

Note: When you create a function mapping, it is possible that the return values from a function evaluated at the data source will be different than the return values from a compatible function evaluated at the DB2 federated database. DB2 will use the function mapping, but it might result in an SQL syntax error or unexpected results.

The DB2 counterpart function can be either a complete function or a function template.

A *function template* is a DB2 function that you create to invoke a function on a data source. The federated server recognizes a data source function when there is a mapping between the data source function and a counterpart function at the federated database. You can create a function template to act as the counterpart when no counterpart exists.

However, unlike a regular function, a function template has no executable code. After you create a function template, you must then create the function mapping between the template and the data source function. You creat a function template with the CREATE FUNCTION statement, using the AS TEMPLATE parameter. You create a function mapping by using the CREATE FUNCTION MAPPING statement. When the federated server receives queries which specify the function template, the federated server will invoke the data source function.

Related concepts:

- “Function mappings options” on page 22

Related reference:

- Appendix F, “Function mapping options for federated systems” on page 301

Function mappings options

The CREATE FUNCTION MAPPING statement includes parameters called *function mapping options*. You can assign values that pertain to the mapping, or to the data source function within the mapping. For example, you can include estimated statistics on the overhead that will be consumed when the data source function is invoked. The query optimizer uses these estimates to decide if the function should be invoked by the data source or by the DB2® federated database.

Related reference:

- Appendix F, “Function mapping options for federated systems” on page 301

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.

Note: You cannot create an index specification for an Informix™ view.

Suppose that a nickname is created for a table that has no index, but the table acquires an index later. 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 indexes. 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.

In a federated system, you use the CREATE INDEX statement against a nickname to supply index specification information to the global catalog. If a table acquires a new index, the CREATE INDEX statement that you create will reference the nickname for the table and contain information about the index of the data source table. If a nickname is created for a view, the CREATE INDEX statement that you create will reference the nickname for the view and contain information about the index of the underlying table for the view.

Related concepts:

- "The SQL Compiler and the query optimizer" on page 8
- "Overview of the tasks to set up a federated system" on page 26
- "Modifying wrappers" on page 195

Related reference:

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

DB2 Relational Connect

Relational Connect is a separately orderable IBM® product that is used with DB2® for UNIX® and Windows, Enterprise Server Edition. Relational Connect feature contains wrappers for the non-IBM relational databases. In DB2 Version 8.1, Relational Connect is required if you want to access data stored in Oracle, Sybase, Microsoft® SQL Server, and ODBC data sources.

Use the Relational Connect edition that matches the operating system that your federated server is using:

- DB2 Relational Connect for AIX®
- DB2 Relational Connect for HP-UX
- DB2 Relational Connect for Linux (Intel 32 bit only)
- DB2 Relational Connect for Solaris Operating Environment
- DB2 Relational Connect for Windows®

Access to data stored in IBM databases (DB2 and Informix) is built into DB2 for UNIX and Windows.

Related concepts:

- "DB2 Relational Connect and the other Connect products" on page 24
- "Using Life Sciences Data Connect with DB2 Relational Connect" on page 26

About the other IBM Connect products

IBM offers several products in addition to Relational Connect that can be used to access data on other servers. This section discusses the differences between:

- DB2 Connect
- DB2 Life Sciences Data Connect
- The Classic Connect driver

This section also discusses how to use DB2 Life Sciences Data Connect with DB2 Relational Connect.

DB2 Relational Connect and the other Connect products

What is DB2 Connect?

DB2® Connect enables distributed database applications to connect to DB2 for z/OS™ and DB2 for iSeries™ data sources, and issue SQL statements. Built into DB2 Connect™ is the DRDA® wrapper. Using DB2 Connect, an individual SQL statement can access tables at only one data source at a time.

DB2 Connect provides extremely fast and robust connectivity to IBM® host and DB2 for iSeries databases for e-business and other applications. DB2 Connect can be used on the following Intel and UNIX® operating systems:

- AIX®
- HP-UX
- Windows®
- Solaris Operating Environment
- Linux

DB2 Connect is supported by the following IBM database products when operating as DRDA Application Servers:

- DB2 for iSeries and AS/400, Version 2.1.1 or higher
- DB2 for MVS/ESA, Version 3.1 and Version 4
- DB2 for z/OS and OS/390, Version 5.1 or higher
- DB2 Universal Database, Version 5
- DB2 Server for VSE and VM, Version 5
- SQL/DS™ Version, 3.5

Using DB2 Connect with DB2 Relational Connect:

You can use DB2 Connect Enterprise Edition with DB2 Relational Connect to issue distributed requests. For example you can query data from Oracle and DB2 for z/OS and OS/390. This type of federated system does not require DB2 for UNIX and Windows. When you install DB2 Relational Connect on the

same server as DB2 Connect, a database is created to store metadata about the data sources. This metadata information is stored in the database catalog. Since this configuration does not use DB2 for UNIX and Windows, no other information is stored locally. This configuration is used to perform transactions between the DB2 family of products supported by DB2 Connect and and the RDBMSs supported by Relational Connect. Additionally, this configuration can be used to improve query performance by allowing joins to be performed on the same server.

A DB2 for UNIX and Windows federated server with Relational Connect provides a capability that the DB2 Connect configuration does not. With a DB2 for UNIX and Windows federated server, you can use the DRDA wrapper to join or update data from multiple data sources using only one SQL statement.

What is DB2 Life Sciences Data Connect?

DB2 Life Sciences Data Connect is a feature of DiscoveryLink. DiscoveryLink is a solution that allows you to access many data sources through a single query interface. Life Sciences Data Connect enables you to integrate a wide variety of genetic, chemical, biological, and other research data from heterogeneous distributed sources. You can access these data sources regardless of the underlying structure or design of each database. You can construct a single query that accesses data from several data sources. You do not have to be concerned with the specific locations of the data, or the specific languages recognized by the data sources. DiscoveryLink uses the DB2 federated database as its underlying engine.

DB2 Life Sciences Data Connect supports the following data sources:

- BLAST search algorithms
- Documentum
- Microsoft® Excel
- Table-structured files
- XML tagged files

What is Classic Connect?

DB2 Classic Connect is a separately orderable database server product you can install on DB2 for z/OS and OS/390® to provide read-only access to IMS™ and VSAM data sources. DB2 for UNIX and Windows does not currently have a Classic Connect wrapper.

Related reference:

- Appendix J, “Quick reference - useful Internet Web sites” on page 335

Using Life Sciences Data Connect with DB2 Relational Connect

A federated system enables data from a life sciences data source to be joined with other relational and non-relational data. The information in these data sources can be accessed as if it was one large database.

The federated server communicates with a data source by using a wrapper. The wrapper allows the server to perform operations such as connecting to a data source and retrieving data from it. Life Science Data Connect contains non-relational and life sciences specific wrappers. A custom wrapper has been designed for each data source supported by Life Sciences Data Connect. The wrapper serves as a transparent interface between DB2® and the data sources.

The query optimizer capabilities in a federated system are based on standard SQL. However, many of the Life Sciences Data Connect data sources use proprietary API calls rather than SQL. The federated server uses the data source information in the wrappers and the global catalog to automatically restructure queries, as necessary for each data source. It then returns the combined results of a search to the end user through DB2.

When used in conjunction with DB2 Relational Connect, SQL statements are used to query, retrieve, and join data located in life sciences data sources as well as relational databases from IBM, Oracle, Sybase, and Microsoft.

Related reference:

- Appendix J, “Quick reference - useful Internet Web sites” on page 335

Overview of the tasks to set up a federated system

This section describes the tasks required to establish and use a federated system. Although the tasks identify the types of users who typically execute the tasks, other types of users can also perform these tasks. For example, the list indicates that Database Administrators typically create mappings between the authorizations to access the federated database and the authorizations to access data sources. However, application programmers and end users can also execute this task.

To establish and use a DB2® federated system:

1. The System Administrator designates a server as a DB2 federated server.
2. The System Administrator installs and configures the data source client software and links DB2 to the client software.
3. The System Administrator installs DB2 and creates the DB2 instance.
4. The System Administrator installs DB2 Relational Connect (if necessary).

5. The Database Administrator creates a wrapper for each category of data source that is to be included in the federated system.
6. The Database Administrator supplies the federated server with a server definition for each data source, and might supply server options to assist in optimizing distributed requests.
7. The Database Administrator defines an association between the authorization IDs that are used to access the federated database and the authorization IDs that are used to access a data source.
8. The Database Administrator and application programmers create nicknames for the data source objects that are to be accessed.
9. Application programmers and end users retrieve information from data sources:
   - Using the DB2 SQL dialect, application programmers and end users submit queries using the nicknames associated with the data source objects.
   - Application programmers and end users occasionally use a pass-through session to submit queries, DML statements, and DDL statements directly to data sources. In a pass-through session, you use the SQL dialect of the data source, rather than the SQL dialect of DB2.

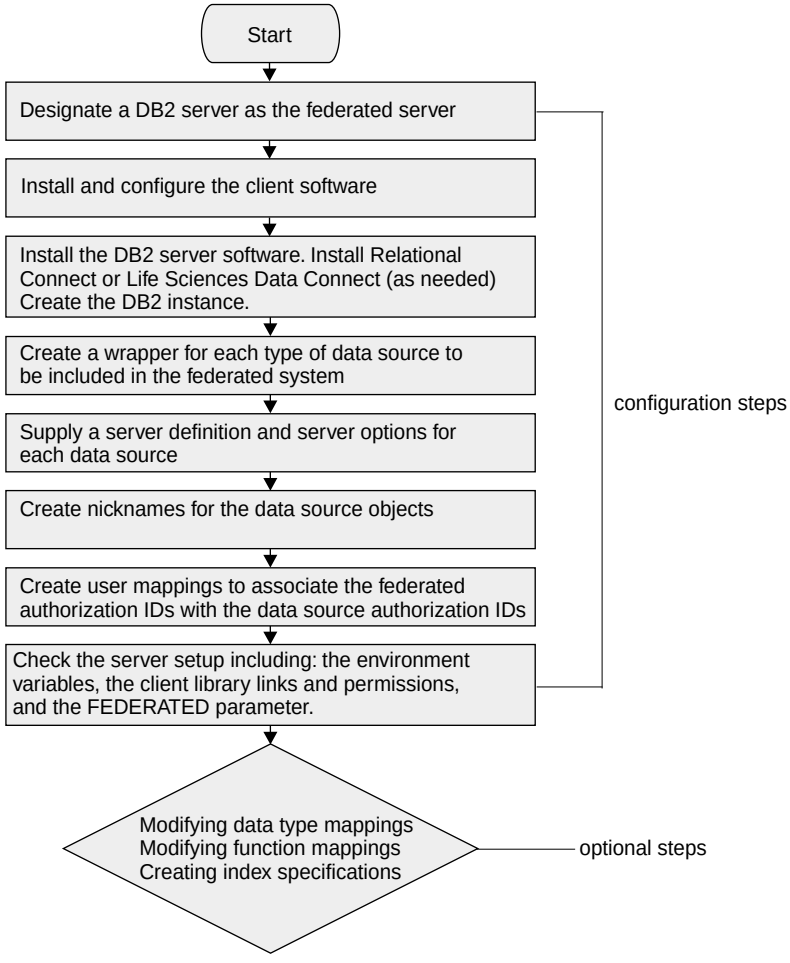

Figure 2. Flowchart of tasks to set up and configure a federated server and database

The following optional tasks are often also performed by the Database Administrator and application programmers:

- Modifying the mapping between a DB2 data type and a data source data type to override the default data type mapping.
- Modifying the mapping between a DB2 function and a data source function to override the default function mapping.
- Providing the federated server with the index specification information if a data source table has an index that the federated server is unaware of.

Related concepts:

- "Fast track to configuring your data sources" on page 85

Related tasks:

- "Creating and modifying function mappings" on page 223
- "Creating index specifications for data source objects" on page 216
- "Modifying default data type mappings" on page 208
- "Fast track to setting up your server and database" on page 39

How you interact with a federated system

You interface with a federated server exactly the same as you do with any other DB2® Universal Database. Typically you interact with the federated system using one of these methods:

- The DB2 command line processor (CLP)
- The DB2 Command Center GUI
- The DB2 Control Center GUI
- Application programs

Note: The federated examples assume you are using the DB2 command line processor (CLP) or the DB2 Command Center GUI to issue DB2 commands, unless otherwise noted.

The steps in the federated documentation specify which tasks can be performed through the Control Center. These steps provide the corresponding commands and SQL statements that can be entered in the DB2 CLP or the DB2 Command Center GUI.

DB2 command line processor (CLP)

You can perform all of the tasks necessary to setup, configure, tune, and maintain the federated system through the CLP. In some cases it is the only way to perform certain tasks you must use either the DB2 CLP or the DB2 Command Center. For example:

- Modify wrapper options.
- Create, alter, or drop user-defined data type mappings.
- Create, alter, or drop user-defined function mappings.

DB2 Command Center

Through the Command Center, you can create and run distributed requests without having to manually type out lengthy SQL statements. Use the Command Center when you are tuning the performance of the federated system. The Command Center is a convenient way to use the DB2 Explain functionality to look at the access plans for distributed requests. The Command Center can also be used to work with the SQL Assistant tool.

DB2 Control Center

The Control Center GUI allows you to perform most of the tasks necessary to setup, configure, and modify the federated system. The Control Center uses panels—dialog boxes and wizards—to guide you through a task. These panels contain interactive help when your mouse hovers over a control such as a list box or command button. Additionally, each panel has a help button that provides information about the panel task, and links to related concepts and reference information.

The Control Center GUI is the easiest way to perform the essential data source configurations:

- Connect to the DB2 federated instance
- Create the wrappers
- Create the server definitions and server options
- Create the user mappings
- Create the nicknames

You can also use the Control Center to modify the data source configuration. You can alter or drop wrappers, server definitions, server options, user mappings, and nicknames.

Application programs

Applications do not require any special coding to work with federated data. Applications access the system just like any other DB2 client application. Applications interface with the federated database that is within the federated server. To obtain data from data sources, they submit queries in DB2 SQL to the federated database. DB2 then distributes the queries to the appropriate data sources, collects the requested data, and returns this data to the applications. However, since DB2 interacts with the data sources through nicknames, you need to be aware of:

- The SQL restrictions you have when working with nicknames
- How to perform operations on nicknamed objects.

Related concepts:

- “How client applications interact with data sources” on page 255

Chapter 2. Business Solutions with federated systems

Use federated functionality with data replication, spatial analysis, and data wearhousing.

Leverage the federated functionality to solve your business needs

Database users want to ask increasingly complex questions about their data in an effort to uncover new and valuable business intelligence. This is a driving force behind the dramatic growth in the development of decision-support and data-warehousing applications. The need for data integration and the ability to model complex data and objects—geo-spatial data, text, images, and other user-defined data types—directly in the DBMS provides users with four key benefits:

- Enhances the business value of existing applications and data.
- Improves business intelligence with integrated searching across all data types.
- Facilitates the development of new applications and queries.
- Improves overall application performance.

To meet this need, IBM® has developed several products that can exploit the federated functionality in DB2® and apply business intelligence to data stored in heterogeneous data sources. These products can address business intelligence requirements across both IBM and non-IBM data sources without having to physically move any data into DB2. This is an important differentiator for IBM.

Replication with a federated system

Replication is the process of maintaining a defined set of data in more than one location. It involves copying designated changes from one location (a source) to another (a target), and synchronizing the data in both locations.

Replication capability is included with DB2® for UNIX® and Windows. For DB2 for iSeries™ and DB2 for z/OS™ and OS/390, a separately orderable product is required to perform replication. This product is DB2 Propagator. Other DB2 products provide distributed computing, including DB2 Connect™ and DB2 Relational Connect. These Connect products allow communication between remote DB2 databases in various operating system environments, or between DB2 databases and non-DB2 relational databases.

DB2 replication helps you integrate your distributed database environment by replicating data between DB2 databases and also between DB2 databases and non-DB2 relational databases using the federated features of DB2. DB2 replication does not require you to unload and load your databases manually, but automates the copying of data between remote systems.

For a non-DB2 relational *source*, DB2 replication creates Capture triggers to capture changes to the source and write them to a staging table. DB2 replication includes the Apply program, which runs on a DB2 server. The Apply program uses a DB2 nickname for the staging table to copy the changes from the staging table to a target table in DB2 or in another non-DB2 relational database.

For a non-DB2 relational *target*, the Apply program copies changes from a DB2 source or another non-DB2 relational source to the target table using a DB2 nickname. You administer DB2 replication for DB2 and non-DB2 sources and targets using the DB2 Replication Center.

For more information about DB2 replication, see the *DB2 Replication Guide and Reference*.

Spatial analysis with a federated system

Traditionally, geo-spatial data have been managed by specialized geographic information systems (GISs) that cannot integrate spatial data with other business data stored in the RDBMS and other data sources. With the addition of object extensions to the RDBMS, GIS intelligence can now be incorporated directly into the database. IBM® has collaborated with partner Environmental Systems Research Institute (ESRI), a major developer of GIS systems, to accomplish this through DB2® Spatial Extender. The integration of Spatial Extender with the DB2 Universal Datbase has several ignificant strengths:

- Full integration of geo-spatial intelligence with the DB2 database and SQL, including indexing and cost-based query optimization. DB2 has new extensions to support Spatial Extender, such as abstract data types and user-defined index structures.
- Compliance with emerging industry standards (SQL3, SQL/MM, and OGIS) and full support for existing GIS data in three industry formats (ESRI shape format, OGIS well-known text format, and OGIS well-known binary format).
- Full support for popular GIS tools plus built-in support for sophisticated rendering of map data for business-intelligence visualization, a geocoding function for U.S. addresses, and a sample set of world map data.

By enabling any organization to enhance its understanding of its business, leverage the value of existing data, and build sophisticated new applications, DB2 Spatial Extender can help users answer many types of questions. Selecting a retail site location , assessing and insurance risk, and selecting a location to market test a new product are examples of these types of questions.

Retail site selection

″Where should we open our new stores?″ A store or restaurant chain wants to expand and is evaluating possible new locations. In addition to typical business criteria, such as lease terms and the size of available buildings, the organization also wants to consider the following for each location: the demographics of the surrounding neighborhood (do the demographics fit our targeted customer base?), the crime rate in the area (a low crime rate is important for retail operations), proximity of the site to major highways (to attract customers from outside the immediate area), proximity of major competitors (a site with little competition will most likely mean higher sales), and proximity to any known problem areas that must be avoided (a restaurant obviously doesn't want to be close to a landfill).

Insurance risk assessment

″Is this home location within our risk parameters? What price should we charge for insuring it?″ Again, standard business considerations (age of the home, construction quality, size, etc.) are not enough in assessing candidates and cost for homeowners insurance. Other factors are the crime rate in the neighborhood, proximity to local emergency services, values of comparable properties in the area, and whether or not the house is within a known flood or earthquake zone.

″What types of accidents happened within 500 feet of this intersection and resulted in total claims payments of more than $10,000? How many of these accidents involved a pedestrian and a compact car?″ In this case, the insurance company is assessing its automobile claims experience in relation to a specific accident location in addition to other factors.

Targeted marketing campaigns

″In which region should we test-market this new product?″ A consumer-products or mail-order company wants to test-market a new product on a limited basis before rolling it out across the entire sales area. Selecting the appropriate location for the test involves finding the best match for the target customer profile among the demographics of each region. Other factors in the test-market decision could include availability or absence of similar competitive products, the popularity or name-brand recognition of various products, and relative shipping costs. The ability to graphically display all of this information on a map will help the company visualize how the criteria overlap and to perhaps identify unexpected criteria that may be important.

Using DB2 Spatial Extender with a federated system

Queries such as these can only be answered if all of the data are available and the DBMS knows how to interpret the data. In the second insurance example, while the DBMS may have information about the location of accidents, it may not have the intelligence to figure out which ones were within a specified distance of the intersection, or which ones involved both a pedestrian and a particular type of car. Traditional DBMSs do not know how to handle spatial data or do complex text searches. They only understand typical business data expressed as numbers, characters, dates, etc. The applications described above illustrate the need to integrate geo-spatial data with traditional business data, to provide advanced query-analysis functions for correlating the data, and to offer end-user tools that can visually display the data in a geo-spatial context.

For detailed information about using Spatial Extender, refer to the *DB2 Spatial Extender User's Guide and Reference*.

Data warehousing with a federated system

Systems that contain operational data—data that captures the daily transactions of your business—are valuable when you need to perform business analysis. However, several problems can arise when the operational data is accessed directly:

- Performance is critical for many operational databases, and the systems cannot process ad hoc queries.
- Some databases require applications programs that use a specialized type of data manipulation language. Users trying to access the data might not have the expertise to query the operational database.
- The operational data is not typically not organized in the most effective way for business analysis. For example, it is more useful to query sales data that has been summarized by product, region, and season than it is to query the raw data.

Data warehousing solves these problems. You can create stores of informational data — data that is extracted from operational data and then transformed for decision making. For example, a data warehousing tool might copy all the sales data from the operational database, perform calculations to summarize the data, and write the summarized data in to a separate database. Users can then query the separate database, called the warehouse, without impacting the operational databases.

The DB2® Data Warehouse Center is a DB2 component that automates data warehouse processing. For detailed information about data warehousing, refer to the Data Warehouse Center web site and the *DB2 Data Warehouse Center Administration Guide*.

Related concepts:

- "What is data warehousing?" in the *Data Warehouse Center Administration Guide*

Part 2. Planning, setting up, and configuring a federated system

Chapter 3. Setting up the federated server and database

There are several tasks that you need to complete before you can access a data source. These tasks are divided into two main catagories:

- Tasks to set up the server and database.
- Tasks to configure the server and database.

To set up the server and database, you need to install the required software on the server and create the database. To configure the server and database, you need to provide the server and database with information about the data sources that you want to access. This chapter discusses all the steps required to set up the federated server and database. Subsequent chapters discuss the steps to configure access to specific data sources.

Fast track to setting up your server and database

A federated server and database are simply a DB2 server and database, that you setup and configure to access data sources. To set up your federated system, you need to:

- Install the required software on the federated server and create the DB2 instance
- Check the server setup
- Create the DB2 database

Note: If you plan to access Life Sciences Data Connect data sources, consult the *DB2 Life Sciences Data Connect: Planning, Installation, and Configuration Guide*, for specific steps to setup the server and database, and configure access to the life sciences data sources.

Note: If you plan to use DB2 Connect, consult the *DB2 Connect Quick Beginnings for DB2 Connect Enterprise Edition*, for specific steps to setup DB2 Connect.

Hardware and software requirements:

The edition of DB2 server software you install depends on the data sources you want to access. This table provides a list of the DB2 server editions and other software you will need to setup a federated server and database.

Table 4. DB2 software required to setup a federated server.

Data source	DB2 V8 server edition	Type of server installation	Client software	Other software
BLAST	• DB2 Enterprise Server Edition • DB2 Connect Enterprise Edition or Unlimited Edition	Any	None. However, the BLAST daemon needs to be installed and configured on BLAST server machine.	DB2 Life Sciences Data Connect
DB2 family of products	• DB2 Enterprise Server Edition • DB2 Workgroup Server Edition or Unlimited Edition • DB2 Personal Edition • DB2 Connect Enterprise Edition or Unlimited Edition	Any	None	
Documentum	• DB2 Enterprise Server Edition • DB2 Connect Enterprise Edition or Unlimited Edition	Any	Documentum client API/Library 3.1.7a (or later)	DB2 Life Sciences Data Connect
Informix	• DB2 Enterprise Server Edition • DB2 Workgroup Server Edition or Unlimited Edition • DB2 Personal Edition • DB2 Connect Enterprise Edition or Unlimited Edition	Custom Use the set up Informix data source support option.	Informix Client SDK	None
Microsoft Excel	DB2 Enterprise Server Edition	Any	Microsoft Excel installed on the federated server	DB2 Life Sciences Data Connect

Table 4. DB2 software required to setup a federated server. (continued)

Data source	DB2 V8 server edition	Type of server installation	Client software	Other software
Microsoft SQL Server	• DB2 Enterprise Server Edition • DB2 Connect Enterprise Edition or Unlimited Edition	Any	• Microsoft SQL Server Client (Windows) • Connect ODBC 3.7 driver (AIX, HP-UX, Linux, Solaris Operating Environment)	DB2 Relational Connect
OLE DB	• DB2 Enterprise Server Edition • DB2 Workgroup Server Edition or Unlimited Edition • DB2 Personal Edition • DB2 Connect Enterprise Edition or Unlimited Edition	Any	OLE DB 2.0 (or later)	OLE DB provider Note: OLE DB components are part of MDAC (Microsoft Data Access Components) and are available on the DB2 CD or from the Microsoft web site.
ODBC	• DB2 Enterprise Server Edition • DB2 Connect Enterprise Edition or Unlimited Edition	Any	ODBC	DB2 Relational Connect
Oracle	• DB2 Enterprise Server Edition • DB2 Connect Enterprise Edition or Unlimited Edition	Any	SQL*Net or Net8	DB2 Relational Connect

Table 4. DB2 software required to setup a federated server. (continued)

Data source	DB2 V8 server edition	Type of server installation	Client software	Other software
Sybase	• DB2 Enterprise Server Edition • DB2 Connect Enterprise Edition or Unlimited Edition	Any	Sybase Open Client	DB2 Relational Connect
Table structured files	• DB2 Enterprise Server Edition • DB2 Connect Enterprise Edition or Unlimited Edition	Any	None	DB2 Life Sciences Data Connect
XML	• DB2 Enterprise Server Edition • DB2 Connect Enterprise Edition or Unlimited Edition	Any	None	DB2 Life Sciences Data Connect

Consult the installation documentation for the specific software you are installing to determine the disk space required.

Table 5. Locating information on software disk space requirements

Software	Disk space information
DB2 Enterprise Server Edition	Consult the *DB2 Universal Database Quick Beginnings for DB2 Servers*
DB2 Workgroup Server Edition and DB2 Workgroup Unlimited Edition	Consult the *DB2 Universal Database Quick Beginnings for DB2 Servers*
DB2 Personal Edition	Consult the *DB2 Universal Database Quick Beginnings for DB2 Personal Edition*
DB2 Connect Enterprise Edition and DB2 Connect Unlimited Edition	Consult the *DB2 Connect Quick Beginnings for DB2 Connect Enterprise Edition*
DB2 Life Sciences Data Connect	Consult the *DB2 Universal Database Life Sciences Data Connect: Planning, Installation, and Configuration Guide*
DB2 Relational Connect	Requires approximately 10 MB of disk space

The steps to accomplish the setup of the federated server and database depend on:

- What you already have setup.
- The data sources you want to access.

For example, the steps for setting up a new server are different than the steps for setting up an existing server. Likewise, the steps to set up the server to access data on Informix data sources are different than the steps to set up the server to access Oracle data sources.

Suppose that you already have a federated server set up, but need to set up access to additional data sources. For each data source, you will need to install and configure any required software. For example, if you have a federated server set up to access Oracle data sources, you will need to install the client software for Sybase to access Sybase data sources. You will also need to install the Sybase data source option from the DB2 Relational Connect CD, if you have not already done so.

The basic steps to setup the federated server are:

1. Install and configure the client configuration software.
2. Install the DB2 server software on the server that will act as the federated server. This includes:
   - Creating a DB2 instance on the federated server.
   - Specifying the user authorities information for the instance.

   If you already have a DB2 for UNIX and Windows Version 8 server set up, confirm that the server software you have installed meets the requirements for the data sources you want to access. For example, if you have the DB2 Workgroup Server Edition installed, you will be able to access DB2 family, OLE DB, and Informix data sources. However, to access Oracle data sources, the DB2 Enterprise Server Edition is required.
3. Install and configure any additional required software on the federated server. This might include DB2 Relational Connect and DB2 Life Sciences Data Connect.
4. Check the server setup. This includes:
   - Confirming the link between the client libraries and DB2.
   - Ensuring the proper permissions are on the wrapper library files.
   - Checking the data source environment variables.
   - Verifying the FEDERATED parameter is set to *YES*.
5. Create a DB2 database on the federated server instance that will act as the federated database.

The DB2 instance owner then proceeds with the steps to configure access to specific data sources.

Because the setup steps vary from data source to data source, the specific steps are provided separate topics.

Related concepts:

- "DB2 Life Sciences Data Connect" in the *DB2 Life Sciences Data Connect Planning, Installation, and Configuration Guide*
- "Typical steps required to install and configure DB2 Connect EE" in the *Quick Beginnings for DB2 Connect Personal Edition*

Related tasks:

- "Setting up the server to access DB2 family data sources" on page 44
- "Setting up the server to access Informix data sources" on page 47
- "Setting up the server to access Oracle data sources" on page 50
- "Setting up the server to access Sybase data sources" on page 54
- "Setting up the server to access Microsoft SQL Server data sources" on page 57
- "Setting up the server to access ODBC data sources" on page 62
- "Setting up the server to access OLE DB data sources" on page 65

Setting up the server to access DB2 family data sources

To set up a server to access DB2 family data sources, you need to install the proper DB2 server software. The DB2 server software that you install, depends on the data sources that you want to access. Use the following table to select the correct DB2 server software.

Table 6. The DB2 UDB editions and the DB2 family data sources they can access.

DB2 server software	Data sources the edition can access
DB2 Universal Database Enterprise Server Edition	• 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
DB2 Universal Database Workgroup Server Edition	• DB2 Universal Database for UNIX and Windows. This includes both remote and local data sources.
DB2 Universal Database Workgroup Unlimited Edition	• DB2 Universal Database for UNIX and Windows. This includes both remote and local data sources.

Table 6. The DB2 UDB editions and the DB2 family data sources they can access. (continued)

DB2 server software	Data sources the edition can access
DB2 Universal Database Personal Edition	• DB2 Universal Database for UNIX and Windows. This includes only local data sources.

Prerequisites:

Before you start the setup, ensure that your system meets installation, memory, and disk requirements. Additionally:

- On UNIX, the DB2 product CD-ROM must be mounted on your system. See the *DB2 Universal Database Quick Beginnings for DB2 Servers* for the steps to mount the CD.
- On Windows, if you plan to use LDAP on Windows 2000 or Windows .NET to register the DB2 server in Active Directory, you must extend the directory schema before you install the DB2 server software.

Restrictions:

On UNIX, you must have root authority to perform the installation.

On Windows, you must have a local Administrator user account with the recommended user rights to perform the installation.

Procedure:

To set up the federated server for DB2 family data sources, install the DB2 server software on the server that will act as the federated server. You install the DB2 server software by using the DB2 Setup Wizard.

To install the DB2 server software:

1. Log on to the system.
   - On UNIX, log on under a user ID that has root authority.
   - On Windows, log on with the Administrator account that you have defined for DB2 installation.
2. Close all open programs so that the DB2 installation program can update files as required.
3. Insert the DB2 CD and start the setup program—the DB2 Setup Wizard—to install the DB2 server software.

- On UNIX, insert and mount the DB2 CD into the CD-ROM. Change to the directory where the CD-ROM is mounted. Enter the **./db2setup** command to start the setup program.
- On Windows, insert the CD-ROM into the drive. The auto-run feature automatically starts the DB2 Setup Wizard. If the setup program fails to auto-start, you can start the DB2 Setup Wizard manually.

  To start the DB2 Setup Wizard manually, click **Start** and select the **Run** option. In the **Open** field, enter `x:\setup`, where `x:` represents your CD-ROM drive. Then click **OK**.

4. The DB2 Setup Launchpad opens. From this window review the installation prerequisites and release notes for late-breaking setup information.
5. Proceed through the DB2 Setup Wizard installation panels and make your selections. As part of the installation:
   - Create a DB2 instance on the federated server.
   - Specify the user authorities information for the instance.

   Installation help is available to guide you through the steps. To invoke the installation help, click **Help** or press F1. You can click **Cancel** at any time to end the installation.
6. Click **Finish** on the last DB2 Setup Wizard installation panel to copy the DB2 files to your system.

   When you complete the installation, DB2 is installed in the one of the following directories, depending on your operating system:

   `/usr/opt/db2_08_01` (AIX)

   `/opt/IBM/db2/V8.1` (HP-UX, Linux, Solaris Operating Environment)

   `\Program Files\IBM\SQLLIB` (Windows)
7. To enable the DB2 server to access data sources, set the FEDERATED parameter to Yes, by issuing this DB2 command:

   ```
   UPDATE DATABASE MANAGER CONFIGURATION USING FEDERATED YES
   ```

After the DB2 server software is installed, a user with SYSADM authority should check the setup and create the federated database. The DB2 instance owner then configures the server to access the DB2 Family data sources.

Related concepts:

- "Instance creation" in the *Administration Guide: Implementation*
- "Installation overview for DB2 servers (UNIX)" in the *Quick Beginnings for DB2 Servers*
- "Installation overview for DB2 servers (Windows)" in the *Quick Beginnings for DB2 Servers*

- “Installation overview for a partitioned DB2 server (UNIX)” in the *Quick Beginnings for DB2 Servers*
- “Installation overview for partitioned DB2 servers (Windows)” in the *Quick Beginnings for DB2 Servers*
- “Fast track to configuring your data sources” on page 85

Related tasks:

- “Mounting the DB2 CD-ROM (HP-UX)” in the *Quick Beginnings for DB2 Servers*
- “Mounting the DB2 CD-ROM (AIX)” in the *Quick Beginnings for DB2 Servers*
- “Mounting the CD-ROM (Solaris)” in the *Quick Beginnings for DB2 Servers*
- “Fast track to setting up your server and database” on page 39
- “Checking the federated server setup” on page 67
- “Creating the federated database” on page 82

Setting up the server to access Informix data sources

If you want to access Informix data sources, you need to install the Informix client software and DB2 server software on the server that will perform as the federated server. You have the choice of installing one of these DB2 server editions:

- DB2 Universal Database Enterprise Server Edition, Version 8
- DB2 Universal Database Workgroup Edition, Version 8
- DB2 Universal Database Workgroup Unlimited Edition, Version 8
- DB2 Universal Database Personal Edition, Version 8

Prerequisites:

Before you start the setup, ensure that your system meets installation, memory, and disk requirements. Additionally:

- On UNIX, the DB2 product CD-ROM must be mounted on your system. See the *DB2 Universal Database Quick Beginnings for DB2 Servers* for the steps to mount the CD.
- On Windows, if you plan to use LDAP on Windows 2000 or Windows .NET to register the DB2 server in Active Directory, you must extend the directory schema before you install the DB2 server software.

Restrictions:

On UNIX, you must have root authority to perform the installation.

On Windows, you must have a local Administrator user account with the recommended user rights to perform the installation.

Procedure:

The steps to set up the federated server for Informix data sources are:

1. Log on to the system.
   - On UNIX, log on under a user ID that has root authority.
   - On Windows, log on with the Administrator account that you have defined for DB2 installation.
2. Close all open programs so that the DB2 installation program can update files as required.
3. Install and configure the Informix Client SDK software on the server that will act as the DB2 federated server.

   See the installation procedures in the documentation that comes with the Informix database software for specific details on how to install the client software.
4. To ensure that the client software is able to connect to the Informix server, run the Informix demo program to test the connection.
5. Insert the DB2 CD and start the setup program—the DB2 Setup Wizard—to install the DB2 server software.
   - On UNIX, insert and mount the DB2 CD into the CD-ROM. Change to the directory where the CD-ROM is mounted. Enter the **./db2setup** command to start the setup program.
   - On Windows, insert the CD-ROM into the drive. The auto-run feature automatically starts the DB2 Setup Wizard. If the setup program fails to auto-start, you can start the DB2 Setup Wizard manually.

     To start the DB2 Setup Wizard manually, click **Start** and select the **Run** option. In the **Open** field, enter `x:\setup`, where `x:` represents your CD-ROM drive. Then click **OK**.
6. The DB2 Setup Launchpad opens. From this window review the installation prerequisites and release notes for late-breaking setup information.
7. Select the Custom installation option. You must use the Custom installation to setup support for Informix data sources.

   Installation help is available to guide you through the steps. To invoke the installation help, click **Help** or press F1. You can click **Cancel** at any time to end the installation.
8. As you proceed through the DB2 Setup Wizard installation panels, choose **Set up Informix data source support**. The set up will require you to identify:
   - The local path where you installed the Informix client software.

- The name of the default Informix server.

The Custom installation will update the `sqllib/cfg/db2dj.ini` file to set several data source environment variables: INFORMIXDIR and INFORMIXSERVER. If you need to set the INFORMIXSQLHOSTS environment variable, you will need to set it manually. The steps are described in the topic, Checking the data source environment variables.

On UNIX, the installation will also link DB2 to the Informix client software.

Caution: If you do not install the Informix client software before you run the DB2 Custom installation, you will have to manually set the environment variables and link DB2 to the client software. These steps are in the topic, Checking the federated server setup.

9. As part of the installation, create a DB2 instance on the federated server and specify the user authorities information for the instance.
10. Click **Finish** on the last DB2 Setup Wizard installation panel to copy the DB2 files to your system.

    When you complete the installation, DB2 is installed in the one of the following directories, depending on your operating system:

    `/usr/opt/db2_08_01` (AIX)

    `/opt/IBM/db2/V8.1` (HP-UX, Linux, Solaris Operating Environment)

    `\Program Files\IBM\SQLLIB` (Windows)
11. To enable the DB2 server to access data sources, set the FEDERATED parameter to Yes, by issuing this DB2 command:

    ```
    UPDATE DATABASE MANAGER CONFIGURATION USING FEDERATED YES
    ```

After the DB2 server software is installed, a user with SYSADM authority should check the setup and create the federated database. The DB2 instance owner then configures the server to access the Informix data sources.

Related concepts:

- “Instance creation” in the *Administration Guide: Implementation*
- “Installation overview for DB2 servers (UNIX)” in the *Quick Beginnings for DB2 Servers*
- “Installation overview for DB2 servers (Windows)” in the *Quick Beginnings for DB2 Servers*
- “Installation overview for a partitioned DB2 server (UNIX)” in the *Quick Beginnings for DB2 Servers*
- “Installation overview for partitioned DB2 servers (Windows)” in the *Quick Beginnings for DB2 Servers*
- “Fast track to configuring your data sources” on page 85

Related tasks:

- "Mounting the DB2 CD-ROM (HP-UX)" in the *Quick Beginnings for DB2 Servers*
- "Mounting the DB2 CD-ROM (AIX)" in the *Quick Beginnings for DB2 Servers*
- "Mounting the CD-ROM (Solaris)" in the *Quick Beginnings for DB2 Servers*
- "Fast track to setting up your server and database" on page 39
- "Checking the federated server setup" on page 67
- "Checking the data source environment variables" on page 68
- "Creating the federated database" on page 82

Setting up the server to access Oracle data sources

If you want to access Oracle data sources, you need to install the following software on the server that will perform as the federated server:

- The Oracle client software. The supported client software versions are 7, 8, and 9.
- DB2 Universal Database Enterprise Server Edition, Version 8
- DB2 Relational Connect, Version 8

Prerequisites:

Before you start the setup, ensure that your system meets installation, memory, and disk requirements. Additionally:

- On UNIX, the DB2 product CD-ROM must be mounted on your system. See the *DB2 Universal Database Quick Beginnings for DB2 Servers* for the steps to mount the CD.
- On Windows, if you plan to use LDAP on Windows 2000 or Windows .NET to register the DB2 server in Active Directory, you must extend the directory schema before you install the DB2 server software.

Restrictions:

On UNIX, you must have root authority to perform the installation.

On Windows, you must have a local Administrator user account with the recommended user rights to perform the installation.

Procedure:

The steps to set up the federated server for Oracle data sources are:

1. Log on to the system.
   - On UNIX, log on under a user ID that has root authority.

- On Windows, log on with the Administrator account that you have defined for DB2 installation.

2. Close all open programs so that the DB2 installation program can update files as required.
3. Install and configure the Oracle client software on the server that will act as the DB2 federated server.
   - You can install the Oracle Version 7 client software on servers that use AIX, and Windows
   - You can install the Oracle Version 8 client software on servers that use AIX, HP-UX, Linux, Solaris Operating Environment, and Windows.

   See the installation procedures in the documentation that comes with the Oracle database software for specific details on how to install the client software.
4. To ensure that the client software is able to connect to the Oracle server, use the Oracle **sqlplus** tool to test the connection.
5. Insert the DB2 CD and start the setup program—the DB2 Setup Wizard—to install the DB2 server software.
   - On UNIX, insert and mount the DB2 CD into the CD-ROM. Change to the directory where the CD-ROM is mounted. Enter the **./db2setup** command to start the DB2 Setup Wizard.
   - On Windows, insert the CD-ROM into the drive. The auto-run feature automatically starts the DB2 Setup Wizard. If the setup program fails to auto-start, you can start the DB2 Setup Wizard manually.

     To start the DB2 Setup Wizard manually, click **Start** and select the **Run** option. In the **Open** field, enter `x:\setup`, where `x:` represents your CD-ROM drive. Then click **OK**.
6. The DB2 Setup Launchpad opens. From this window review the installation prerequisites and release notes for late-breaking setup information.
7. Proceed through the DB2 Setup Wizard installation panels and make your selections.

   Note: As part of the installation, do not create a DB2 instance. You will create the instance when you install DB2 Relational Connect.

   Installation help is available to guide you through the steps. To invoke the installation help, click **Help** or press F1. You can click **Cancel** at any time to end the installation.
8. Click **Finish** on the last DB2 Setup Wizard installation panel to copy the DB2 files to your system.

   When you complete the installation, DB2 is installed in the one of the following directories, depending on your operating system:

   `/usr/opt/db2_08_01` (AIX)

/opt/IBM/db2/V8.1 (HP-UX, Linux, Solaris Operating Environment)

\Program Files\IBM\SQLLIB (Windows)

After you install the client software and the DB2 server software, you need to install DB2 Relational Connect, Version 8 on the DB2 server. DB2 Relational Connect contains the software that you need to access Oracle data sources.

1. Log on to the system.
   - On UNIX, log on under a user ID that has root authority.
   - On Windows, log on with the Administrator account that you have defined for DB2 installation.
2. Close all open programs so that the installation program can update files as required.
3. Insert the DB2 Relational Connect CD, and start the setup program to install DB2 Relational Connect.
   - On UNIX, insert and mount the DB2 Relational Connect CD into the CD-ROM. Change to the directory where the CD-ROM is mounted. Enter the **./db2setup** command to start the setup program.
   - On Windows, insert the DB2 Relational Connect CD into the CD-ROM drive. The auto-run feature automatically starts the setup program. If the setup program fails to auto-start, you can start the setup program manually.

     To start the setup program manually, click **Start** and select the **Run** option. In the **Open** field, enter x:\setup, where x: represents your CD-ROM drive. Then click **OK**.
4. The DB2 Relational Connect Setup Launchpad opens. From this window review the installation prerequisites and release notes for late-breaking setup information.
5. From the **Select the features to install** panel in the setup program, choose **Relational Connect for Oracle Data Sources**. The set up will require you to identify the local path where you installed the Oracle client software.

   The Relational Connect installation will update the sqllib/cfg/db2dj.ini file to set the ORACLE_HOME environment variable. If you need to set the ORACLE_BASE and ORA_NLS environment variables, you will need to set them manually. The steps are described in the topic, Checking the data source environment variables.

   On UNIX, the installation will also link DB2 to the Oracle client software.

   Caution: If you do not install the Oracle client software before you run the DB2 Relational Connect installation, you will have to manually set the environment variables and link DB2 to the client software. These steps are in the topic, Checking the federated server setup.

Installation help is available to guide you through the steps. To invoke the installation help, click **Help** or press F1. You can click **Cancel** at any time to end the installation.

6. As part of the installation:
   - Create a DB2 instance on the federated server. This will set the DB2 database manager FEDERATED parameter to YES, which enables the DB2 server to access the data sources.
   - Specify the user authorities information for the instance.
7. Click **Finish** on the last setup installation panel to copy the DB2 Relational Connect files to your system.

   When you complete the installation, DB2 Relational Connect is installed in the same directory as the DB2 server software.

After the software is installed, a user with SYSADM authority should check the setup and create the federated database. The DB2 instance owner then configures the server to access the Oracle data sources.

Related concepts:

- "Instance creation" in the *Administration Guide: Implementation*
- "Installation overview for DB2 servers (UNIX)" in the *Quick Beginnings for DB2 Servers*
- "Installation overview for DB2 servers (Windows)" in the *Quick Beginnings for DB2 Servers*
- "Installation overview for a partitioned DB2 server (UNIX)" in the *Quick Beginnings for DB2 Servers*
- "Installation overview for partitioned DB2 servers (Windows)" in the *Quick Beginnings for DB2 Servers*
- "Fast track to configuring your data sources" on page 85

Related tasks:

- "Mounting the DB2 CD-ROM (HP-UX)" in the *Quick Beginnings for DB2 Servers*
- "Mounting the DB2 CD-ROM (AIX)" in the *Quick Beginnings for DB2 Servers*
- "Mounting the CD-ROM (Solaris)" in the *Quick Beginnings for DB2 Servers*
- "Fast track to setting up your server and database" on page 39
- "Checking the federated server setup" on page 67
- "Checking the data source environment variables" on page 68
- "Creating the federated database" on page 82

Setting up the server to access Sybase data sources

If you want to access Sybase data sources, you need to install the following software on the server that will perform as the federated server:

- Sybase Open Client software, Version 11.1 or later
- DB2 Universal Database Enterprise Server Edition, Version 8
- DB2 Relational Connect, Version 8

Prerequisites:

Before you start the setup, ensure that your system meets installation, memory, and disk requirements. Additionally:

- On UNIX, the DB2 product CD-ROM must be mounted on your system. See the *DB2 Universal Database Quick Beginnings for DB2 Servers* for the steps to mount the CD.
- On Windows, if you plan to use LDAP on Windows 2000 or Windows .NET to register the DB2 server in Active Directory, you must extend the directory schema before you install the DB2 server software.

Restrictions:

On UNIX, you must have root authority to perform the installation.

On Windows, you must have a local Administrator user account with the recommended user rights to perform the installation.

Procedure:

The steps to set up the federated server for Sybase data sources are:

1. Log on to the system.
   - On UNIX, log on under a user ID that has root authority.
   - On Windows, log on with the Administrator account that you have defined for DB2 installation.
2. Close all open programs so that the DB2 installation program can update files as required.
3. Install and configure the Sybase Open Client software on the server that will act as the DB2 federated server.

   See the installation procedures in the documentation that comes with the Sybase database software for specific details on how to install the client software.
4. To ensure that the client software is able to connect to the Sybase server, use a Sybase tool such as **isql** to test the connection.

5. Insert the DB2 CD and start the setup program—the DB2 Setup Wizard—to install the DB2 server software.
   - On UNIX, insert and mount the DB2 CD into the CD-ROM. Change to the directory where the CD-ROM is mounted. Enter the **./db2setup** command to start the DB2 Setup Wizard.
   - On Windows, insert the CD-ROM into the drive. The auto-run feature automatically starts the DB2 Setup Wizard. If the setup program fails to auto-start, you can start the DB2 Setup Wizard manually.

     To start the DB2 Setup Wizard manually, click **Start** and select the **Run** option. In the **Open** field, enter `x:\setup`, where `x:` represents your CD-ROM drive. Then click **OK**.
6. The DB2 Setup Launchpad opens. From this window review the installation prerequisites and release notes for late-breaking setup information.
7. Proceed through the DB2 Setup Wizard installation panels and make your selections.

   Note: As part of the installation, do not create a DB2 instance. You will create the instance when you install DB2 Relational Connect.

   Installation help is available to guide you through the steps. To invoke the installation help, click **Help** or press F1. You can click **Cancel** at any time to end the installation.
8. Click **Finish** on the last DB2 Setup Wizard installation panel to copy the DB2 files to your system.

   When you complete the installation, DB2 is installed in the one of the following directories, depending on your operating system:

   `/usr/opt/db2_08_01` (AIX)

   `/opt/IBM/db2/V8.1` (HP-UX, Linux, Solaris Operating Environment)

   `\Program Files\IBM\SQLLIB` (Windows)

After you install the client software and the DB2 server software, you need to install DB2 Relational Connect, Version 8 on the DB2 server. DB2 Relational Connect contains the software that you need to access Sybase data sources.

1. Log on to the system.
   - On UNIX, log on under a user ID that has root authority.
   - On Windows, log on with the Administrator account that you have defined for DB2 installation.
2. Close all open programs so that the installation program can update files as required.
3. Insert the DB2 Relational Connect CD, and start the setup program to install DB2 Relational Connect.

- On UNIX, insert and mount the DB2 Relational Connect CD into the CD-ROM. Change to the directory where the CD-ROM is mounted. Enter the **./db2setup** command to start the setup program.
- On Windows, insert the DB2 Relational Connect CD into the CD-ROM drive. The auto-run feature automatically starts the setup program. If the setup program fails to auto-start, you can start the setup program manually.

  To start the setup program manually, click **Start** and select the **Run** option. In the **Open** field, enter `x:\setup`, where `x:` represents your CD-ROM drive. Then click **OK**.

4. The DB2 Relational Connect Setup Launchpad opens. From this window review the installation prerequisites and release notes for late-breaking setup information.
5. From the **Select the features to install** panel in the setup program, choose **Relational Connect for Sybase Data Sources**. The setup will require you to identify the local path where you installed the Sybase client software.

   The Relational Connect installation will update the `sqllib/cfg/db2dj.ini` file to set the SYBASE environment variable. If you need to set the SYBASE_OCS environment variable, you will need to set it manually. The steps are described in the topic, Checking the data source environment variables.

   On UNIX, the installation will also link DB2 to the Sybase client software.

   Caution: If you do not install the Sybase Open client software before you run the DB2 Relational Connect installation, you will have to manually set the environment variables and link DB2 to the client software. These steps are in the topic, Checking the federated server setup.

   Installation help is available to guide you through the steps. To invoke the installation help, click **Help** or press F1. You can click **Cancel** at any time to end the installation.
6. As part of the installation:
   - Create a DB2 instance on the federated server. This will set the DB2 database manager FEDERATED parameter to YES, which enables the DB2 server to access the data sources.
   - Specify the user authorities information for the instance.
7. Click **Finish** on the last setup installation panel to copy the DB2 Relational Connect files to your system.

   When you complete the installation, DB2 Relational Connect is installed in the same directory as the DB2 server software.

After the software is installed, a user with SYSADM authority should check the setup and create the federated database. The DB2 instance owner then configures the server to access the Sybase data sources.

Related concepts:

- "Instance creation" in the *Administration Guide: Implementation*
- "Installation overview for DB2 servers (UNIX)" in the *Quick Beginnings for DB2 Servers*
- "Installation overview for DB2 servers (Windows)" in the *Quick Beginnings for DB2 Servers*
- "Installation overview for a partitioned DB2 server (UNIX)" in the *Quick Beginnings for DB2 Servers*
- "Installation overview for partitioned DB2 servers (Windows)" in the *Quick Beginnings for DB2 Servers*
- "Fast track to configuring your data sources" on page 85

Related tasks:

- "Mounting the DB2 CD-ROM (HP-UX)" in the *Quick Beginnings for DB2 Servers*
- "Mounting the DB2 CD-ROM (AIX)" in the *Quick Beginnings for DB2 Servers*
- "Mounting the CD-ROM (Solaris)" in the *Quick Beginnings for DB2 Servers*
- "Fast track to setting up your server and database" on page 39
- "Checking the federated server setup" on page 67
- "Checking the data source environment variables" on page 68
- "Creating the federated database" on page 82

Setting up the server to access Microsoft SQL Server data sources

If you want to access Microsoft SQL Server data sources, you need to install the following software on the server that will perform as the federated server:

- The proper ODBC driver:
  - On UNIX, the Connect ODBC Version 3.7 driver
  - On Windows, the Microsoft SQL Server Client Version 2000.8 driver
- DB2 Universal Database Enterprise Server Edition, Version 8
- DB2 Relational Connect, Version 8

Prerequisites:

Before you start the setup, ensure that your system meets installation, memory, and disk requirements. Additionally:

- On UNIX, the DB2 product CD-ROM must be mounted on your system. See the *DB2 Universal Database Quick Beginnings for DB2 Servers* for the steps to mount the CD.

- On Windows, if you plan to use LDAP on Windows 2000 or Windows .NET to register the DB2 server in Active Directory, you must extend the directory schema before you install the DB2 server software.

Restrictions:

On UNIX, you must have root authority to perform the installation.

On Windows, you must have a local Administrator user account with the recommended user rights to perform the installation.

Procedure:

The steps to set up the federated server for Microsoft SQL Server data sources are:

1. Log on to the system.
   - On UNIX, log on under a user ID that has root authority.
   - On Windows, log on with the Administrator account that you have defined for DB2 installation.
2. Close all open programs so that the DB2 installation program can update files as required.
3. Install and configure the ODBC driver on the server that will act as the DB2 federated server.
   - On UNIX, install the Connect ODBC 3.7 driver and test the connection to the Microsoft SQL Server.
     - Specify the Connect library directory as the first entry in the LIBPATH.
     - Make the Connect ODBC libraries available to other users by checking the permissions on the Connect ODBC libraries.
     - Test the configuration of the `.odbc.ini` and the connection to the Microsoft SQL Server data source using the Connect ODBC **demoodbc** test tool. The **demoodbc** test tool is located in the `/demo` subdirectory of Connect ODBC. The **demoodbc** test tool attempts to connect to a requested SQL Server data source and query the EMP table. Since it is unlikely that the Microsoft SQL Server data source has an EMP table, you should expect to receive error messages. The test is successful if:
       - Messages indicate there is no EMP table.
       - Records from an EMP table are returned.
       - Messages indicate that there is an EMP table, but that the requested columns are not present.

The **demoodbc** test tool needs to be run by a user on the UNIX system other than root authority. If no other user is on the system, a user with root authority can create a group and user ID for the DB2 instance. Use this user ID to run the **demoodbc** test tool. For example, root can create the group *db2admin1* and the user *db2inst1*. This does not create the instance, it adds a new user ID that will be the instance owner. To run **demoodbc**, the *db2inst1* user needs to:

- Add the Connect ODBC `/lib` subdirectory to the LIBPATH system environment variable value. Typically the directory is `/opt/odbc/lib` and can be set with this command:

  ```
  export LIBPATH=/opt/odbc/lib:$LIBPATH
  ```

- Set the ODBCINI system environment variable to point to the location of the `odbc.ini`. file that has ODBC connection information for the SQL Server data source. Use the export command to set the ODBCINI system environment variable. For example, if the location of the `odbc.ini` file is the home directory of DB2 instance owner user *db2inst1* and the federated server operating system is AIX, the command is:

  ```
  export ODBCINI=/home/db2inst1/.odbc.ini
  ```

- On Windows, the Microsoft SQL Server Client Version 2000.8 driver should be installed when you install Windows. You need to confirm that the driver is installed, and configure the driver to access Microsoft SQL Server data sources.
  - Confirm that the driver is installed. Access the Microsoft ODBC Data Source Administrator through the Windows Control Panel. Click the **Drivers** tab to confirm that the 2000.8 driver is installed.
  - Register the Microsoft SQL Server data source as a System DSN. Use the **Configure** button to test the connection to the Microsoft SQL Server data source.

  Note: If you are using Microsoft SQL Server 2000 Personal Edition, you must use the SQL Server Client Network Utility to add a new SQL Server ODBC data source to your ODBC System DSN list.

See the installation procedures in the documentation that comes with the ODBC driver for specific details on how to install and configure the driver.

4. Insert the DB2 CD and start the setup program—the DB2 Setup Wizard—to install the DB2 server software.
   - On UNIX, insert and mount the DB2 CD into the CD-ROM. Change to the directory where the CD-ROM is mounted. Enter the **./db2setup** command to start the DB2 Setup Wizard.
   - On Windows, insert the CD-ROM into the drive. The auto-run feature automatically starts the DB2 Setup Wizard. If the setup program fails to auto-start, you can start the DB2 Setup Wizard manually.

To start the DB2 Setup Wizard manually, click **Start** and select the **Run** option. In the **Open** field, enter `x:\setup`, where `x:` represents your CD-ROM drive. Then click **OK**.

5. The DB2 Setup Launchpad opens. From this window review the installation prerequisites and release notes for late-breaking setup information.
6. Proceed through the DB2 Setup Wizard installation panels and make your selections.

   Note: As part of the installation, do not create a DB2 instance. You will create the instance when you install DB2 Relational Connect.

   Installation help is available to guide you through the steps. To invoke the installation help, click **Help** or press F1. You can click **Cancel** at any time to end the installation.
7. Click **Finish** on the last DB2 Setup Wizard installation panel to copy the DB2 files to your system.

   When you complete the installation, DB2 is installed in the one of the following directories, depending on your operating system:

   `/usr/opt/db2_08_01` (AIX)

   `/opt/IBM/db2/V8.1` (HP-UX, Linux, Solaris Operating Environment)

   `\Program Files\IBM\SQLLIB` (Windows)

After you install the client software and the DB2 server software, you need to install DB2 Relational Connect, Version 8 on the DB2 server. DB2 Relational Connect contains the software that you need to access Microsoft SQL Server data sources.

1. Log on to the system.
   - On UNIX, log on under a user ID that has root authority.
   - On Windows, log on with the Administrator account that you have defined for DB2 installation.
2. Close all open programs so that the installation program can update files as required.
3. Insert the DB2 Relational Connect CD, and start the setup program to install DB2 Relational Connect.
   - On UNIX, insert and mount the DB2 Relational Connect CD into the CD-ROM. Change to the directory where the CD-ROM is mounted. Enter the **./db2setup** command to start the setup program.
   - On Windows, insert the DB2 Relational Connect CD into the CD-ROM drive. The auto-run feature automatically starts the setup program. If the setup program fails to auto-start, you can start the setup program manually.

To start the setup program manually, click **Start** and select the **Run** option. In the **Open** field, enter `x:\setup`, where `x:` represents your CD-ROM drive. Then click **OK**.

4. The DB2 Relational Connect Setup Launchpad opens. From this window review the installation prerequisites and release notes for late-breaking setup information.
5. From the **Select the features to install** panel in the setup program, choose **Relational Connect for SQL Server Data Sources**. The setup will require you to identify:
   - The local path where the ODBC driver is installed.
   - The local path of the ODBC Driver Manager directory.
   - The local path of the ODBC trace directory.
   - The local path of the ODBC library.

   The Relational Connect installation will update the `sqllib/cfg/db2dj.ini` file to set several data source environment variables: ODBCINI, DJXODBCTRACE, and DJX_ODBC_LIBRARY_PATH 7. If you need to set other environment variables, you will need to set it manually. The steps are described in the topic, Checking the data source environment variables.

   On UNIX, the installation will also link DB2 to the ODBC driver.

   Caution: If you do not install the Microsoft SQL Server driver before you run the DB2 Relational Connect installation, you will have to manually set the environment variables and link DB2 to the client software. These steps are in the topic, Checking the federated server setup.

   Installation help is available to guide you through the steps. To invoke the installation help, click **Help** or press F1. You can click **Cancel** at any time to end the installation.
6. As part of the installation:
   - Create a DB2 instance on the federated server. This will set the DB2 database manager FEDERATED parameter to YES, which enables the DB2 server to access the data sources.
   - Specify the user authorities information for the instance.
7. Click **Finish** on the last setup installation panel to copy the DB2 Relational Connect files to your system.

   When you complete the installation, DB2 Relational Connect is installed in the same directory as the DB2 server software.
8. Link or copy the `.odbc.ini` file to the home directory of the DB2 instance owner. The `.odbc.ini` file that comes with the ODBC driver is located in the client directory.

After the software is installed, a user with SYSADM authority should check the setup and create the federated database. The DB2 instance owner then configures the server to access the Microsoft SQL Server data sources.

Related concepts:

- "Instance creation" in the *Administration Guide: Implementation*
- "Installation overview for DB2 servers (UNIX)" in the *Quick Beginnings for DB2 Servers*
- "Installation overview for DB2 servers (Windows)" in the *Quick Beginnings for DB2 Servers*
- "Installation overview for a partitioned DB2 server (UNIX)" in the *Quick Beginnings for DB2 Servers*
- "Installation overview for partitioned DB2 servers (Windows)" in the *Quick Beginnings for DB2 Servers*
- "Fast track to configuring your data sources" on page 85

Related tasks:

- "Mounting the DB2 CD-ROM (HP-UX)" in the *Quick Beginnings for DB2 Servers*
- "Mounting the DB2 CD-ROM (AIX)" in the *Quick Beginnings for DB2 Servers*
- "Mounting the CD-ROM (Solaris)" in the *Quick Beginnings for DB2 Servers*
- "Fast track to setting up your server and database" on page 39
- "Checking the federated server setup" on page 67
- "Checking the data source environment variables" on page 68
- "Creating the federated database" on page 82

Setting up the server to access ODBC data sources

If you want to access ODBC data sources, you need to install the following software on the server that will perform as the federated server:

- ODBC Version 3.0 driver
- DB2 Universal Database Enterprise Server Edition, Version 8
- DB2 Relational Connect, Version 8

Prerequisites:

Before you start the setup, ensure that your system meets installation, memory, and disk requirements. Additionally:

- On Windows, if you plan to use LDAP on Windows 2000 or Windows .NET to register the DB2 server in Active Directory, you must extend the directory schema before you install the DB2 server software.

Restrictions:

On Windows, you must have a local Administrator user account with the recommended user rights to perform the installation.

Procedure:

The steps to set up the federated server for ODBC data sources are:

1. Log on to the system with the Administrator account that you have defined for DB2 installation.
2. Close all open programs so that the DB2 installation program can update files as required.
3. Install and configure the ODBC Version 3.0 driver on the server that will act as the DB2 federated server. Register the ODBC data source as a System DSN. Access the Microsoft ODBC Data Source Administrator through the Windows Control Panel. Use the **Configure** button to test the connection to the ODBC data source.

   See the installation procedures in the documentation that comes with the ODBC driver for specific details on how to install and configure the driver.
4. Insert the DB2 CD and start the setup program—the DB2 Setup Wizard—to install the DB2 server software.
   - On Windows, insert the CD-ROM into the drive. The auto-run feature automatically starts the DB2 Setup Wizard. If the setup program fails to auto-start, you can start the DB2 Setup Wizard manually.

     To start the DB2 Setup Wizard manually, click **Start** and select the **Run** option. In the **Open** field, enter `x:\setup`, where `x:` represents your CD-ROM drive. Then click **OK**.
5. The DB2 Setup Launchpad opens. From this window review the installation prerequisites and release notes for late-breaking setup information.
6. Proceed through the DB2 Setup Wizard installation panels and make your selections.

   Note: As part of the installation, do not create a DB2 instance. You will create the instance when you install DB2 Relational Connect.

   Installation help is available to guide you through the steps. To invoke the installation help, click **Help** or press F1. You can click **Cancel** at any time to end the installation.
7. Click **Finish** on the last DB2 Setup Wizard installation panel to copy the DB2 files to your system.

   When you complete the installation, DB2 is installed in the following directory:

   `\Program Files\IBM\SQLLIB`

After you install the client software and the DB2 server software, you need to install DB2 Relational Connect, Version 8 on the DB2 server. DB2 Relational Connect contains the software that you need to access ODBC data sources.

1. Log on to the system with the Administrator account that you have defined for DB2 installation.
2. Close all open programs so that the installation program can update files as required.
3. Insert the DB2 Relational Connect CD, and start the setup program to install DB2 Relational Connect.
   - On Windows, insert the DB2 Relational Connect CD into the CD-ROM drive. The auto-run feature automatically starts the setup program. If the setup program fails to auto-start, you can start the setup program manually.

     To start the setup program manually, click **Start** and select the **Run** option. In the **Open** field, enter `x:\setup`, where `x:` represents your CD-ROM drive. Then click **OK**.
4. The DB2 Relational Connect Setup Launchpad opens. From this window review the installation prerequisites and release notes for late-breaking setup information.
5. From the **Select the features to install** panel in the setup program, choose **Relational Connect for ODBC Data Sources**.

   Installation help is available to guide you through the steps. To invoke the installation help, click **Help** or press F1. You can click **Cancel** at any time to end the installation.
6. As part of the installation:
   - Create a DB2 instance on the federated server. This will set the DB2 database manager FEDERATED parameter to YES, which enables the DB2 server to access the data sources.
   - Specify the user authorities information for the instance.
7. Click **Finish** on the last setup installation panel to copy the DB2 Relational Connect files to your system.

   When you complete the installation, DB2 Relational Connect is installed in the same directory as the DB2 server software.

After the software is installed, a user with SYSADM authority should check the setup and create the federated database. The DB2 instance owner then configures the server to access the ODBC data sources.

Related concepts:

- "Instance creation" in the *Administration Guide: Implementation*
- "Installation overview for DB2 servers (UNIX)" in the *Quick Beginnings for DB2 Servers*

- “Installation overview for DB2 servers (Windows)” in the *Quick Beginnings for DB2 Servers*
- “Installation overview for a partitioned DB2 server (UNIX)” in the *Quick Beginnings for DB2 Servers*
- “Installation overview for partitioned DB2 servers (Windows)” in the *Quick Beginnings for DB2 Servers*
- “Fast track to configuring your data sources” on page 85

Related tasks:

- “Mounting the DB2 CD-ROM (HP-UX)” in the *Quick Beginnings for DB2 Servers*
- “Mounting the DB2 CD-ROM (AIX)” in the *Quick Beginnings for DB2 Servers*
- “Mounting the CD-ROM (Solaris)” in the *Quick Beginnings for DB2 Servers*
- “Fast track to setting up your server and database” on page 39
- “Checking the federated server setup” on page 67
- “Creating the federated database” on page 82

Setting up the server to access OLE DB data sources

If you want to access data that is stored in OLE DB data sources, you have the choice of installing one of these DB2 server editions:

- DB2 Universal Database Enterprise Server Edition, Version 8
- DB2 Universal Database Workgroup Edition, Version 8
- DB2 Universal Database Workgroup Unlimited Edition, Version 8
- DB2 Universal Database Personal Edition, Version 8

Prerequisites:

Before you start the setup, ensure that your system meets installation, memory, and disk requirements. Additionally:

- On UNIX, the DB2 product CD-ROM must be mounted on your system. See the *Quick Beginnings for DB2 Servers* for the steps to mount the CD.
- On Windows, if you plan to use LDAP on Windows 2000 or Windows .NET to register the DB2 server in Active Directory, you must extend the directory schema before you install the DB2 server software.

Restrictions:

On UNIX, you must have root authority to perform the installation.

On Windows, you must have a local Administrator user account with the recommended user rights to perform the installation.

Procedure:

To set up the federated server for OLE DB data sources, install the DB2 server software on the server that will act as the federated server. You install the DB2 server software by using the DB2 Setup Wizard.

To install the DB2 server software:

1. Log on to the system.
   - On UNIX, log on under a user ID that has root authority.
   - On Windows, log on with the Administrator account that you have defined for DB2 installation.
2. Close all open programs so that the DB2 installation program can update files as required.
3. Insert the DB2 CD and start the setup program—the DB2 Setup Wizard—to install the DB2 server software.
   - On UNIX, insert and mount the DB2 CD into the CD-ROM. Change to the directory where the CD-ROM is mounted. Enter the **./db2setup** command to start the setup program.
   - On Windows, insert the CD-ROM into the drive. The auto-run feature automatically starts the DB2 Setup Wizard. If the setup program fails to auto-start, you can start the DB2 Setup Wizard manually.

     To start the DB2 Setup Wizard manually, click **Start** and select the **Run** option. In the **Open** field, enter `x:\setup`, where `x:` represents your CD-ROM drive. Then click **OK**.
4. The DB2 Setup Launchpad opens. From this window review the installation prerequisites and release notes for late-breaking setup information.
5. Proceed through the DB2 Setup Wizard installation panels and make your selections. As part of the installation:
   - Create a DB2 instance on the federated server.
   - Specify the user authorities information for the instance.

   Installation help is available to guide you through the steps. To invoke the installation help, click **Help** or press F1. You can click **Cancel** at any time to end the installation.
6. Click **Finish** on the last DB2 Setup Wizard installation panel to copy the DB2 files to your system.

   When you complete the installation, DB2 is installed in the one of the following directories, depending on your operating system:

`/usr/opt/db2_08_01` (AIX)

`/opt/IBM/db2/V8.1` (HP-UX, Linux, Solaris Operating Environment)

`\Program Files\IBM\SQLLIB` (Windows)

7. To enable the DB2 server to access data sources, set the FEDERATED parameter to Yes, by issuing this DB2 command:

```
UPDATE DATABASE MANAGER CONFIGURATION USING FEDERATED YES
```

After the DB2 server software is installed, a user with SYSADM authority should check the setup and create the federated database. The DB2 instance owner then configures the server to access the OLE DB data sources.

Related concepts:

- "Instance creation" in the *Administration Guide: Implementation*
- "Installation overview for DB2 servers (UNIX)" in the *Quick Beginnings for DB2 Servers*
- "Installation overview for DB2 servers (Windows)" in the *Quick Beginnings for DB2 Servers*
- "Installation overview for a partitioned DB2 server (UNIX)" in the *Quick Beginnings for DB2 Servers*
- "Installation overview for partitioned DB2 servers (Windows)" in the *Quick Beginnings for DB2 Servers*
- "Fast track to configuring your data sources" on page 85

Related tasks:

- "Mounting the DB2 CD-ROM (HP-UX)" in the *Quick Beginnings for DB2 Servers*
- "Mounting the DB2 CD-ROM (AIX)" in the *Quick Beginnings for DB2 Servers*
- "Mounting the CD-ROM (Solaris)" in the *Quick Beginnings for DB2 Servers*
- "Fast track to setting up your server and database" on page 39
- "Checking the federated server setup" on page 67
- "Creating the federated database" on page 82

Checking the federated server setup

After the federated server is setup, you can avoid potential problems by checking several key settings:

- Check the data source environment variables (UNIX, and Windows for Informix data sources).
- Confirm the link between DB2 and the data source client libraries (UNIX only).
- Check the wrapper library file permissions (UNIX only).

- Ensure that the FEDERATED parameter is set to YES (UNIX and Windows).

Checking the federated server setup—details

Checking the data source environment variables

When you setup the federated server, the installation process attempts to set the environment variables for the Informix, Oracle, Sybase, and Microsoft SQL Server data sources.

- For Oracle, Sybase, and Microsoft SQL Server data sources, you only need to check the environment variables if your federated server uses a UNIX operating system.
- For Informix data sources, you need to check the environment variables on both UNIX and Windows operating systems.

Prerequisites:

A federated server that is properly setup to access your data sources. This includes the installation and configuration of any required software, such as: the client software, DB2 Relational Connect, or DB2 Life Sciences Data Connect.

Restrictions:

Procedure:

Check to make certain that the environment variables for the data sources you want access are set in the `sqllib/cfg/db2dj.ini` file.

The system administrator should check for the data source environment variables.

The following table lists the valid data source environment variables.

Table 7. Valid data source environment variables.

Data source	Valid environment variables
Informix	INFORMIXDIR
	INFORMIXSERVER
	INFORMIXSQLHOSTS

Table 7. Valid data source environment variables. (continued)

Data source	Valid environment variables
Oracle	ORACLE_HOME
	ORACLE_BASE
	ORA_NLS
	TNS_ADMIN
Microsoft SQL Server	ODBCINI
	DJXODBCTRACE
	DJX_ODBC_LIBRARY_PATH 7
Sybase	SYBASE
	SYBASE_OCS

The data source environment variables will not be set in the `sqllib/cfg/db2dj.ini` file if you:

- Install the data source client software *after* the DB2 federated server is setup.
- Have not installed the data source client software.

To set the environment variables:

1. Install the client software (if necessary).
2. Set the environment variables. The quickest way to set the data source environment variables is:
   - For Informix data sources, run the DB2 server Custom installation again.
   - For Oracle, Microsoft SQL Server, and Sybase data sources, run the DB2 Relational Connect installation again.

You can also manually set the environment variables.

Manually setting the Informix environment variables

To manually set the Informix environment variables, follow these steps:

1. Edit the `db2dj.ini` file located in `sqllib/cfg` directory. The `db2dj.ini` file contains configuration information about the Informix client software installed on your federated server. If the file does not exist, you can create a new file with this name. In the `db2dj.ini` you must specify the fully qualified path for the variable, otherwise you will encounter errors. Set the following environment variables as necessary.

   INFORMIXDIR

Set the INFORMIXDIR environment variable to the directory path where the Informix Client SDK software is installed; for example:

```
INFORMIXDIR=/informix/csdk
```

INFORMIXSERVER

This variable identifies the name of the default Informix server. This setting must be a valid entry in the `sqlhosts` file (UNIX) or the SQLHOSTS registry key (WINDOWS). For example:

```
INFORMIXSERVER=inf93
```

Note: Although the Informix wrapper does not use the value of this variable, the Informix client requires that this variable be set. The wrapper uses the value of the **node** server option, which specifies the Informix database server that you want to access.

INFORMIXSQLHOSTS

If you are using the default path for the Informix `sqlhosts` file, you do not need to set this variable. However, if you are using some other path for the Informix `sqlhosts` file, then you need to set this variable to the full path name where the Informix `sqlhosts` file resides.

- On UNIX, the default path is `$INFORMIXDIR/etc/sqlhosts`.
- On Windows, if the SQLHOSTS registry key does not reside on the local computer, then the INFORMIXSQLHOSTS is the name of the Windows computer that stores the registry.

An example of setting this variable to another path is:

```
INFORMIXSQLHOSTS=/informix/csdk/etc/my_sqlhosts
```

2. To ensure that the environment variables are set in the program, recycle the DB2 instance. Issue the following commands to recycle the DB2 instance:

```
db2stop
db2start
```

Manually setting the Oracle environment variables

To manually set the Oracle environment variables, follow these steps:

1. Edit the `db2dj.ini` file located in `sqllib/cfg` directory. The `db2dj.ini` file contains configuration information about the Oracle client software installed on your federated server. If the file does not exist, you can create a new file with this name. In the `db2dj.ini` you must specify the fully qualified path for the variable, otherwise you will encounter errors. Set the following environment variables as necessary.

ORACLE_HOME

Set the ORACLE_HOME environment variable to the directory path where the Oracle client software is installed. Specify the fully qualified path for the variable, ORACLE_HOME=<oracle_home_directory>. For example, if the Oracle home directory is /usr/oracle/8.1.7, the entry in the db2dj.ini is:

```
ORACLE_HOME=/usr/oracle/8.1.7
```

Note: If an individual user of the federated instance has the ORACLE_HOME environment variable set, federated instance does not use that setting. The federated instance uses only the value of ORACLE_HOME that you set in the DB2 profile registry.

ORACLE_BASE

ORACLE_BASE represents the root of the Oracle client directory tree. If you set the ORACLE_BASE variable when you installed the Oracle client software, set the ORACLE_BASE environment variable on the federated server. For example:

```
ORACLE_BASE=<oracle_root_directory>
```

ORA_NLS

If your system is using multiple versions of Oracle, you must ensure that:

- The appropriate ORA_NLS variable is set.
- The corresponding NLS data files for the versions you are using are available.

The location-specific data is stored in a directory specified by the ORA_NLS environment variable. For each new version of Oracle, there is a different ORA_NLS data directory.

Table 8. Oracle ORA_NLS directory name, by version.

Oracle version	Environment variable
7.2	ORA_NLS
7.3	ORA_NLS32
8.0, 8.1, 9.0.1	ORA_NLS33

For example, for federated servers that access Oracle 8.1 data sources, set the ORA_NLS environment variable:

```
ORA_NLS32=<oracle_home_directory>/ocommon/nls/admin/data>
```

TNS_ADMIN

The Oracle client expects to locate the tnsnames.ora file in the /NETWORK/ADMIN directory. On UNIX, the client will also look for

the tnsnames.ora file in the /etc directory. If the tnsnames.ora file is not located in one of these directories, you need to set the TNS_ADMIN environment variable on the federated server. For example:

```
TNS_ADMIN=<tnsnames.ora_directory>
```

2. Update the .profile file of the DB2 instance with the Oracle environment variable. You can do this by issuing the following command:

   ```
   export PATH=$ORACLE_HOME/bin:$PATH
   export ORACLE_HOME=<oracle_home_directory>
   ```

   where <oracle_home_directory> is the directory where the Oracle client software is installed.

3. Execute the DB2 instance .profile by entering:

   ```
   . .profile
   ```

4. Ensure that the environment variables are set in the program by recycling the DB2 instance. Issue the following commands to recycle the DB2 instance:

   ```
   db2stop
   db2start
   ```

Manually setting the Sybase environment variables

To manually set the Sybase environment variables, follow these steps:

1. Edit the db2dj.ini file located in sqllib/cfg directory. The db2dj.ini file contains configuration information about the Sybase Open Client client software installed on your federated server. If the file does not exist, you can create a new file with this name. In the db2dj.ini you must specify the fully qualified path for the variable, otherwise you will encounter errors. Set the following environment variables as necessary:

   SYBASE

   Set the SYBASE environment variable to the directory path where the Sybase Open Client software is installed. For example:

   ```
   SYBASE=<sybase_home_directory>
   ```

   SYBASE_OCS

   For Sybase Open Client Version 12 or later, set the SYBASE_OCS environment variable to the name of the OCS directory. For example:

   ```
   SYBASE_OCS=OCS-<version>_<release>
   ```

   where:

   - <version> is the version number of the Sybase Open Client that is installed.

- <release> is the release number of the Sybase Open Client that is installed.

2. Update the .profile file of the DB2 instance with the Sybase environment variable. You can do this by issuing the following command:

```
export PATH=$SYBASE/bin:$PATH
export SYBASE=<sybase_home_directory>
```

   where <sybase_home_directory> is the directory where the Sybase Open Client software is installed.

3. Execute the DB2 instance .profile by entering:

```
. .profile
```

4. To ensure that the environment variables are set in the program, recycle the DB2 instance. Issue the following commands to recycle the DB2 instance:

```
db2stop
db2start
```

Manually setting the Microsoft SQL Server ODBC driver environment variables

To manually set the Microsoft SQL Server ODBC driver environment variables, follow these steps:

1. Edit the db2dj.ini file located in sqllib/cfg directory. The db2dj.ini file contains configuration information about the Microsoft SQL Server ODBC driver installed on your federated server. If the file does not exist, you can create a new file with this name. In the db2dj.ini you must specify the fully qualified path for the variable, otherwise you will encounter errors. Set the following environment variables as necessary:

   ODBCINI

   Set the ODBCINI environment variable to the directory path where the ODBC driver is installed. For example:

```
ODBCINI=<ODBC_ home_directory>
```

   DJXODBCTRACE

   Set the DJXODBCTRACE environment variable to the directory path where the ODBC trace is installed. For example:

```
DJXODBCTRACE=<ODBC_trace_directory>
```

   DJX_ODBC_LIBRARY_PATH

   Set the directory path to the ODBC library files. For example:

```
DJX_ODBC_LIBRARY_PATH=<ODBC_home_directory>/lib
```

   where:

   - <ODBC_home_directory> is the directory path where the ODBC driver is installed.

DB2LIBPATH

To access to Microsoft SQL Server, you need to set the directory path to the ODBC library files in the/lib subdirectory. For example:

```
DB2LIBPATH=<ODBC_driver_directory>/lib
```

where:

- <ODBC_driver_directory> is the directory path where the ODBC driver is installed.

DB2ENVLIST

To use the Connect ODBC driver to access Microsoft SQL Server data sources, set DB2ENVLIST with a value of LIBPATH. For example:

```
DB2ENVLIST=LIBPATH
```

where:

- <ODBC_driver_directory> is the directory path where the ODBC driver is installed.

2. To ensure that the environment variables are set in the program, recycle the DB2 instance. Issue the following commands to recycle the DB2 instance:

```
db2stop
db2start
```

Applying the environment variables in a multi-partition instance configuration

To apply the data source environment variables to the appropriate nodes on your federated server, use the **db2set** command. This step is only necessary if you are running DB2 for UNIX on your federated server and have a multiple-partition instance configuration. The **db2set** command displays, sets or removes DB2 profile variables. The syntax of the command is dependent upon your database system structure. The db2dj.ini file contains the data source environment variables. This file was added to the federated server when you installed Relational Connect.

Applying the environment variables to all nodes: If you want the values in the db2dj.ini file to apply to all nodes within this instance, issue:

```
db2set -g DB2_DJ_INI=$HOME/sqllib/cfg/db2dj.ini
```

Applying the environment variables to the current node only: If you are using the db2dj.ini file in a non-partitioned database system, or if you want the db2dj.ini file to apply to the current node only, issue:

```
db2set DB2_DJ_INI=$HOME/sqllib/cfg/db2dj.ini
```

Applying the environment variables to a specific node: If you want the values in the db2dj.ini file to apply to a specific node, issue:

```
db2set -i INSTANCEX  3 DB2_DJ_INI=$HOME/sqllib/cfg/node3.ini
```

where:

INSTANCEX
Is the name of the instance.

3 Is the node number as listed in the `db2nodes.cfg` file.

node3.ini
Is the modified and renamed version of the db2dj.ini file.

Related tasks:

- "Checking the federated server setup" on page 67
- "Checking the FEDERATED parameter" on page 81

Confirming the link between DB2 and the data source client libraries (UNIX)

A federated server, using a UNIX operating system, must be link-edited to the data source client libraries. This applies to the following data sources:

- Informix. The link-edit step is attempted when you install the DB2 server software using the Custom installation option.
- Oracle. The link-edit step is attempted when you install DB2 Relational Connect.
- Sybase. The link-edit step is attempted when you install DB2 Relational Connect.
- Microsoft SQL Server. The link-edit step is attempted when you install DB2 Relational Connect.
- Documentum. The link-edit step is attempted when you install DB2 Life Sciences Data Connect.

  Note: Consult the DB2 Life Sciences Data Connect documentation for information on linking the federated server to the Documentum data source client libraries.

The link-edit step creates a wrapper library for each data source that the federated server will communicate with.

If the data source client software was not installed before you installed the DB2 server software, the link-edit step will fail. You will then need to perform the link manually.

Prerequisites:

A federated server that is properly setup to access your data sources. This includes the installation and configuration of any required software, such as: the client software, DB2 Relational Connect, or DB2 Life Sciences Data Connect.

Restrictions:

You need root authorization to run the link scripts.

Procedure:

Determine the status of the link between DB2 and the data source client libraries:

- If the link-edit was successful, the wrapper library file appears in the directory.
- If the link-edit failed, check the error message file in the directory.
- If the link-edit was not performed, neither the library file or message file appears in the directory. You will have to manually run the link script.

The following sections contain information on how to confirm the status of the link-edit, and provide instructions on how to perform the links manually.

Checking for the wrapper library files

The link-edit scripts create the wrapper libraries in specific directories, depending on the operating system. The following tables list the directory path for the library file names by data source. If the wrapper library file appears in the directory, the link-edit was successful.

Informix:

The directory paths and wrapper library file names for Informix.

Table 9. Informix wrapper library locations and file names

Data source	Operating system	Directory path	Wrapper library file
Informix	AIX	usr/opt/db2_08_01/lib/	libdb2informix.a
	HP-UX	/opt/IBM/db2/V8.1/lib	libdb2informix.sl
	Linux	/opt/IBM/db2/V8.1/lib	libdb2informix.so
	Solaris Operating Environment	/opt/IBM/db2/V8.1/lib	libdb2informix.so

Microsoft SQL Server:

The directory paths and wrapper library file names for Microsoft SQL Server.

Table 10. Microsoft SQL Server client library locations and file names

Data source	Operating system	Directory path	Wrapper library file
Microsoft SQL Server	AIX	usr/opt/db2_08_01/lib/	libdb2mssql3.a
	HP-UX	/opt/IBM/db2/V8.1/lib	libdb2mssql3.sl
	Linux	/opt/IBM/db2/V8.1/lib	libdb2mssql3.so
	Solaris Operating Environment	/opt/IBM/db2/V8.1/lib	libdb2mssql3.so

Oracle:

The directory paths and wrapper library file names for Oracle.

Table 11. Oracle client library locations and file names

Data source	Operating system	Directory path	Wrapper library file
Oracle	AIX	usr/opt/db2_08_01/lib/	libdb2sqlnet.a (SQLNET) libdb2net8.a (NET8)
	HP-UX	/opt/IBM/db2/V8.1/lib	libdb2sqlnet.sl (SQLNET) libdb2net8.sl (NET8)
	Linux	/opt/IBM/db2/V8.1/lib	libdb2sqlnet.so (SQLNET) libdb2net8.so (NET8)
	Solaris Operating Environment	/opt/IBM/db2/V8.1/lib	libdb2sqlnet.so (SQLNET) libdb2net8.so (NET8)

Sybase:

The directory paths and wrapper library file names for Sybase.

Table 12. Sybase client library locations and file names

Data source	Operating system	Directory path	Wrapper library file
Sybase	AIX	usr/opt/db2_08_01/lib/	libdb2ctlib.a (CTLIB) libdb2dblib.a (DBLIB)
	HP-UX	/opt/IBM/db2/V8.1/lib	libdb2ctlib.sl (CTLIB) libdb2dblib.sl (DBLIB)
	Linux	/opt/IBM/db2/V8.1/lib	libdb2ctlib.so (CTLIB) libdb2dblib.so (DBLIB)
	Solaris Operating Environment	/opt/IBM/db2/V8.1/lib	libdb2ctlib.so (CTLIB) libdb2dblib.so (DBLIB)

Checking the link-edit error message files

If the link-edit fails, there will be errors listed in the error message file in the library directory. There may be an error message file in the library directory, even if the link-edit is successful. You need to open the error message file to determine if the link-edit failed. The link-edit error message file names are listed in the following table.

Table 13. Link-edit error message file names by data source

Data source	Error message file names
Informix	djxlinkInformix.out
Microsoft SQL Server	djxlinkMssql.out
Oracle	djxlinkOracle.out
Sybase	djxlinkSybase.out

Manually linking DB2 to the data source client libraries

The link script creates the wrapper libraries on the federated server for the data source you are setting up. There are several reasons why the link might fail when you setup the federated server:

- If the client software is not installed before the link-edit is attempted, then the link-edit will fail. For example, if you do not install the Informix client software before you install the DB2 server software, the link-edit will fail. Likewise, if you do not install the Sybase Open Client software before you install DB2 Relational Connect, the link-edit will fail. In these situations, you will have to perform the link manually.
- Check to make sure the version of the data source client is supported. The latest information is on the product Web sites. For Informix, Oracle, Sybase, and Microsoft SQL Server, check the DB2 Relational Connect Web site www.ibm.com/software/data/db2/relconnect/. For Documentum, check the DB2 Life Sciences Data Connect Web site www.ibm.com/solutions/lifesciences/. If the version of the data source client you have installed is not supported, the link-edit will fail. You will have to install a client version that is supported and then perform the link manually.

You need root authorization to run the link scripts. The quickest way to link DB2 to the data source client libraries is:

1. Install and configure the client software on the DB2 federated server (if necessary).
2. Use the product CDs:
   - For Informix data sources, run the DB2 server Custom installation again.
   - For Oracle, Sybase, and Microsoft SQL Server data sources, run the DB2 Relational Connect installation again.
   - For Documentum data sources, run the DB2 Life Sciences Data Connect installation again.

Alternatively, you can run the link scripts from the UNIX command prompt.

The following table lists the link script names for each data source.

Table 14. Link scripts by data source

Data source	Link script name
Informix	djxlinkInformix
Oracle	djxlinkOracle
Sybase	djxlinkSybase
Microsoft SQL Server	djxlinkMssql

For example, if you are setting up the federated server to access Informix data sources, issue `djxlinkInformix` script from the UNIX command prompt:

```
djxlinkInformix
```

If you manually run a link script, you must issue the **db2iupdt** command on each DB2 instance to enable federated access to the data sources.

Note: There is another script, the `djxlink` script, that attempts to create a wrapper library for every data source that DB2 for UNIX and Windows supports. If you only have the client software for some of the data sources installed, you will receive an error message for each of the missing data sources when you issue the `djxlink` script.

Once the link is performed, check the permissions on the wrapper libraries after they are created. Make sure that the libraries can be read and executed by the DB2 instance owners.

Related concepts:

- "What is Documentum?" in the *DB2 Life Sciences Data Connect Planning, Installation, and Configuration Guide*

Related tasks:

- "Checking the federated server setup" on page 67
- "Checking the wrapper library file permissions (UNIX)" on page 80

Checking the wrapper library file permissions (UNIX)

The link-edit step creates a wrapper library for each data source that the federated server will communicate with. The federated server uses these files to access the Informix, Oracle, Sybase, Microsoft SQL Server, and Documentum data sources.

The link-edit step is performed by someone with root authority. When the library files are created, the file permissions only allow root and users in the systems group to read and execute the library files. Change the file permissions so that other users can read and execute the wrapper libraries.

Prerequisites:

A federated server that is properly setup to access your data sources. This includes the installation and configuration of any required software, such as: the client software, DB2 Relational Connect, or DB2 Life Sciences Data Connect.

Restrictions:

Although any user can check the permissions, only a user with root authority can change the permissions.

Procedure:

The system administrator should check the permissions on the wrapper library files to ensure that the DB2 instance owner can read and execute the files.

1. To check permissions, change to the directory where the wrapper library was created.

Table 15. Default path for data source wrapper libraries.

Operating system	Directory path
AIX	usr/opt/db2_08_01/lib/
HP-UX	/opt/IBM/db2/V8.1/lib
Linux	/opt/IBM/db2/V8.1/lib
Solaris Operating Environment	/opt/IBM/db2/V8.1/lib

2. List the statistics for the wrapper library file. The list will show the permissions, owner, and group for the file. Use the **ls** command, for example:

   ```
   ls -l <filename>
   ```

   The permissions will probably be -rwxr-x---. These permissions allow only the owner and users in the group to read and execute the wrapper library. The owner will probably be *root*, and the group will probably be *system*.
3. Use the **chmod** command to change the permissions so that other users can read and execute the wrapper library. The proper permissions are:

   ```
   rwx r-x r-x
   ```

   To permit other users (such as the DB2 instance owner) to read and execute the library, use:

   ```
   chmod o+rx <filename>
   ```

Related tasks:

- "Checking the federated server setup" on page 67
- "Confirming the link between DB2 and the data source client libraries (UNIX)" on page 75

Checking the FEDERATED parameter

The FEDERATED parameter must be set to YES to enable access to the data sources. It is possible that this parameter was set when you created the DB2 instance. However, it is important to make certain that the FEDERATED parameter to *YES*.

Prerequisites:

A federated server that is properly setup to access your data sources. This includes the installation and configuration of any required software, such as: the client software, DB2 Relational Connect, or DB2 Life Sciences Data Connect.

Procedure:

To check the FEDERATED parameter setting, issue this command:

```
GET DATABASE MANAGER CONFIGURATION
```

This will display all of the parameters and their current settings. Check that the Federated Database System Support (FEDERATED) parameter is set to *YES*.

If the FEDERATED parameter is set to *NO*, issue this command to change the setting:

```
UPDATE DATABASE MANAGER CONFIGURATION USING FEDERATED YES
```

Note: The CONCENTRATOR parameter and the FEDERATED parameter cannot be configured to YES at the same time. If the CONCENTRATOR parameter is set to *YES*, you must change it to *NO* before you can set the FEDERATED parameter.

Related tasks:

- "Checking the federated server setup" on page 67
- "Checking the data source environment variables" on page 68

Creating the federated database

After you setup the federated server, the DB2 instance owner creates a DB2 database on the federated server instance that will act as the federated database.

You can create the database two ways:

- Through the DB2 Control Center
- Through the DB2 Command Center or DB2 command line processor (CLP).

The advantage of using the DB2 Control Center is that you do not have to key in each statement and command. It is the easiest way to quickly create a database.

The steps in this section assume that you are using the DB2 Command Center or the command line processor (CLP) to create the database.

Prerequisites:

A federated server that is properly setup to access your data sources. This includes the installation and configuration of any required software, such as: the client software, DB2 Relational Connect, or DB2 Life Sciences Data Connect.

Restrictions:

You need SYSADM or SYSCTRL authority to create a DB2 database.

Procedure:

Create a DB2 database on the federated server instance that will act as the federated database. For example:

```
CREATE DATABASE federated
```

This command:

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

In a multi-node environment, this command affects all nodes that are listed in the `db2nodes.cfg` file. The node from which this command is issued, becomes the catalog node for the new database.

Related reference:

- "CREATE DATABASE Command" in the *Command Reference*

Obtaining updates for DB2 and Relational Connect

Take advantage of the latest updates to DB2 and Relational Connect by installing the most current DB2 FixPak for your operating system.

Keep your DB2 environment running at the latest FixPak level. DB2 FixPaks contain updates and fixes for problems—Authorized Program Analysis Reports (APARs)—found during testing at IBM, as well as fixes for problems that are reported by our customers. A document that is called APARLIST.TXT, accompanies every FixPak.. The APARLIST.TXT document describes the problem fixes contained in the FixPak.

FixPaks are cumulative. This means that the latest FixPak for any given version of DB2 contains all of the updates from previous FixPaks for the same version of DB2.

Procedure:

To obtain the latest FixPak, visit this IBM support Web site: www.ibm.com/software/data/db2/udb/winos2unix/support/download.d2w/report

To learn about specific updates to Relational Connect, visit the Relational Connect Web site: www.ibm.com/software/data/db2/relconnect/

Related reference:

- Appendix J, “Quick reference - useful Internet Web sites” on page 335

Chapter 4. Overview of configuring access to data sources

This chapter is a concise guide to configuring a federated server and database to access your data sources:

- It contains information about the basic steps needed to quickly perform the configuration steps.
- It outlines several optional steps, if you need them, to fine-tune the data source configuration.
- To help you avoid problems, the end of this chapter contains configuration troubleshooting advice.

There are individual configuration chapters for each data source. Additionally, there is an appendix at the end of the book which contains the syntax required for each SQL statement used to configure the data sources.

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 16. 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	
4. Create the user mappings	DB2 Control Center	

Table 16. The recommended interface and configuration steps (continued)

Configuration step	Recommended interface	Notes
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:

- "Object Linking and Embedding Database (OLE DB) Table Functions" in the *Application Development Guide: Building and Running Applications*
- "Prepare the federated database" on page 86
- "Optional configuration steps" on page 98

Related tasks:

- "Checking the federated server setup" on page 67
- "Administering federated database systems : Federated Systems help" in the *Help: Federated Systems*
- "Contents : Federated Systems help" in the *Help: Federated Systems*
- "Adding a data source to a federated system : Federated Systems help" in the *Help: Federated Systems*

Prepare the federated database

Configuring access to a data source involves supplying the federated database with information about the data source. The first configuration step is to prepare the federated database.

For DB2 family data sources:

There are two steps that are required to prepare the federated database to access DB2® family data sources:

- Cataloging a node entry in the federated node directory. The federated server uses this entry to determine the proper access method it will use to connect to a DB2 data source.
- Catalog the remote database in the federated system database directory. This identifies the DB2 data source database to which you want to establish a connection. Use the Client Configuration Assistant (CCA) to catalog the remote database. For federated servers on UNIX, you can also use the CATALOG DATABASE command.

Note: Do not use the CATALOG DCS DATABASE command to catalog the remote database.

You can catalog the node and database either in the DB2 Command Center or through the command line processor (CLP). For the specific steps, consult the topic Configuring access to DB2 family data sources.

For Informix, Oracle, and Sybase data sources:

These data sources require a client configuration file to connect to a data source. This file specifies the location of each data source server and type of connection (protocol) to that server.

The way you set up this file depends on the data source you are accessing. For Oracle data sources or Sybase data sources, you can use a utility that comes with the client software to set up the file. For Informix™ data sources, you can copy the file from another system or configure the client software on the DB2 federated server to create the file.

You need to set up a client configuration file on each instance in the DB2 federated server that will connect to the data source.

After you set up the client configuration file, test the connection to the data source. The test confirms that the data source client software on the federated server is able to connect to the data source server. If the data source client software includes a query tool, use this tool to test the connection. For example, if the data source client is Oracle Version 8, you can use the **sqlplus** tool. Query the data source catalog table and a user table for a thorough test.

You can set up the client the client configuration file either in the DB2 Command Center or through the command line processor (CLP). The steps to create the client configuration file are different for each data source. You will find specific steps in the configuration topics for each data source.

For Microsoft SQL Server data sources:

The way you prepare the federated server for Microsoft® SQL Server data sources depends on the operating system on your federated server.

If your federated server uses Windows® NT or Windows 2000 for its operating system, you need to confirm that the ODBC System DSN is properly set up. You should also test the connection to the Microsoft SQL Server remote server. The test confirms that the ODBC driver on the federated server is able to connect to the Microsoft SQL Server remote server.

If your federated server runs on AIX, HP-UX, Linux, or Solaris Operating Environment, you need to update or create an `odbc.ini` file and test the connection to the Microsoft SQL Server remote server.

For ODBC data sources:

On your Windows federated server, you need to confirm that the ODBC System DSN is properly set up. You should also test the connection to the remote data source. The test confirms that the ODBC driver on the federated server is able to connect to the remote data source.

For OLE DB data sources:

This data source does not require additional federated database preparation. The next step is to create the data source wrappers.

Related tasks:

- "Setting up the server to access DB2 family data sources" on page 44
- "Setting up the server to access Informix data sources" on page 47
- "Setting up the server to access Oracle data sources" on page 50
- "Setting up the server to access Sybase data sources" on page 54

Create the wrapper

A wrapper performs many tasks:

- It connects to the data source. Most wrappers use the standard connection API of the data source.
- It submits SQL to the data source. Most wrappers use the data source standard APIs for submitting dynamic SQL.
- It receives results sets from the data source. Most wrappers use 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.

- It responds to federated server queries about the default function mappings for a data source. The wrapper contains information that the federated server uses to determine if a DB2® function is mapped to data source function. The wrapper uses these mappings when translating the DB2 SQL to the data source SQL dialect.

Note: The Microsoft® SQL Server and ODBC wrappers use ODBC APIs.

To define and register a wrapper in the federated database, use DB2 Control Center. You can also use the CREATE WRAPPER statement in the DB2 Command Center or the command line processor (CLP).

You only need to create one wrapper for each 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 Sybase tables, and one Sybase view as shown in the following figure. You only need to create two wrappers — one for the DB2 data source objects and one for the Sybase data source objects. DB2 for z/OS and DB2 for iSeries will use the DRDA® wrapper, and Sybase will use the either the CTLIB or DBLIB wrapper.

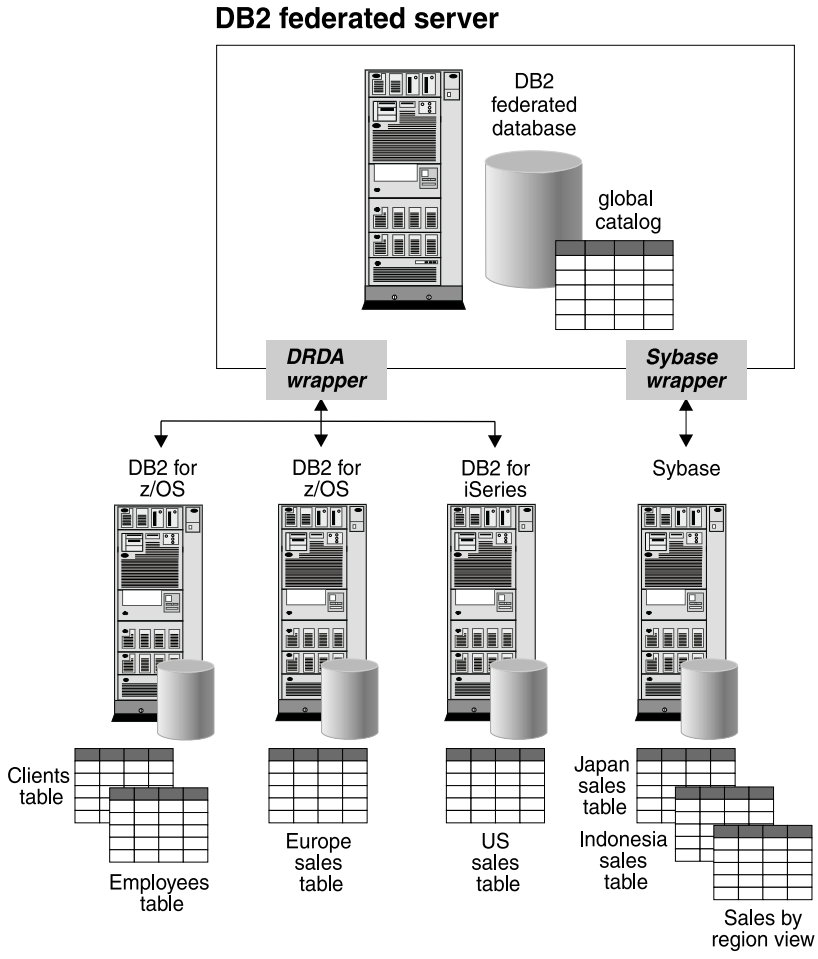

Figure 3. Create one wrapper for each data source, a wrapper for DB2 and a wrapper for Sybase

For each data source there is a default wrapper name. The syntax of the CREATE WRAPPER statement is:

```
CREATE WRAPPER wrapper_name
```

For example, to create a wrapper to access the DB2 family of products the command would be:

```
CREATE WRAPPER DRDA
```

IBM® recommends that you use the default wrapper name. When you use the default name to create the wrapper, the federated server automatically picks up the library associated with the wrapper.

Recommendation: Use the default wrapper name. When the wrapper is created using the default name, the federated server automatically picks up the default library name associated with the wrapper. If the wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name you choose. If you use a name that is different than the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement. An example of the CREATE WRAPPER statement with the LIBRARY parameter is:

```
CREATE WRAPPER wrapper_name LIBRARY 'library_name'
```

You will find the default wrapper names and the library names listed in the configuration topics for each data source.

Related tasks:

- "Wrappers : Federated Systems help" in the *Help: Federated Systems*
- "Creating a wrapper: Federated Systems help" in the *Help: Federated Systems*

Related reference:

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

Supply the server definition

After you create the wrapper, you need to identify each data source server that you want to access. To create a server definition, use the DB2® Control Center. You can also issue the CREATE SERVER statement in the DB2 Command Center, or the command line processor (CLP). The parameters and options required with the CREATE SERVER statement, depend on the data source you want to access.

For DB2 family data sources:

Suppose that you have two DB2 for OS/390® databases. The NEWYORKDB database is on the NEWYORK390 server, and contains a client table and an employee table. The LONDONDB database is on the LONDON390 server, and contains a sales table. You will need to create two server definitions: one for the NEWYORK390 server and one for the LONDON390 server. You will then need to create three nicknames, one for each of the tables.

An example of the server definition for the NEWYORK390 server is:

```
CREATE SERVER NEWYORK TYPE DB2/390
VERSION 5 WRAPPER DRDA
AUTHORIZATION 'STEWART' PASSWORD 'BONNIE',
OPTIONS (DBNAME 'NYCLIENTS')
```

where:

NEWYORK
Is a name that you assign to the data source server. This name must be unique.

TYPE *DB2/390*
Specifies the type of data source to which you are configuring access.

VERSION *5*
Is the version of data source server software that you want to access.

WRAPPER *DRDA*
Is the wrapper name that you specified in the CREATE WRAPPER statement.

AUTHORIZATION *'STEWART'*
Is the authorization ID at the data source. This value is case-sensitive.

PASSWORD *'BONNIE'*
Is the password associated with the authorization ID at the data source. This value is case-sensitive.

DBNAME *'NYCLIENTS'*
Is the name of the database that you want to access. This value is case-sensitive.

The AUTHORIZATION parameter, the PASSWORD parameter, and the DBNAME option are required..

For Informix, Sybase, and OLE DB data sources:

Suppose that you have on Sybase database, called SYBDB, located on the SY6500 server. The SYBDB database contains three objects: two tables and a view. You will need to create one server definition for the SY6500 server. You will then need to create three nicknames, one for each of the tables and one for the view.

An example of the server definition for the SY6500 server is:

```
CREATE SERVER SYBSERVER TYPE SYBASE
VERSION 12.0 WRAPPER CTLIB
OPTIONS (NODE 'sybnode' DBNAME 'sybDB')
```

where:

SYBSERVER
Is a name that you assign to the data source server. This name must be unique.

TYPE *SYBASE*
Specifies the type of data source to which you are configuring access.

VERSION *12.0*
Is the version of data source server software that you want to access.

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

NODE *'sybnode'*
Is a name that a data source is defined as an instance to its RDBMS. This value is case-sensitive.

DBNAME *'sybDB'*
Is the name of the database that you want to access. This value is case-sensitive.

The NODE option and DBNAME option are required. Additionally, Informix™ requires that the IUD_APP_SVPT_ENFORCE server option is set to*'N'*

For Oracle, ODBC, and Microsoft SQL Server data sources:

The server definition for Oracle, ODBC, and Microsoft® SQL Server data sources is similar to the ones for Informix, Sybase, and OLE DB data sources. The only difference is that the DBNAME option is not required.

The NODE option is required.

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.

Additional server options

When you create the server definition, you can specify additional server options in the CREATE SERVER statement. There are general server options and data source-specific server options.

For example, when connecting to a data source, the federated server tries to connect using all possible combinations of upper and lower case for the user ID and password. This means that the server might make up to nine connect attempts before successfully connecting to the data source server. These attempts can slow down connect times. You can prevent this by specifying values for the `FOLD_ID` and `FOLD_PW` server options.

Related tasks:

- "Servers : Federated Systems help" in the *Help: Federated Systems*
- "Selecting server options: Federated Systems help" in the *Help: Federated Systems*

- "Viewing server options: Federated Systems help" in the *Help: Federated Systems*
- "Creating a server: Federated Systems help" in the *Help: Federated Systems*

Related reference:

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

Create the user mappings and test the connection to the data source

After you create the server definition, you need define a user mapping. A user is an association between your authorization ID to access the federated database and your authorization ID to access the data source. This association ensures that distributed requests can be sent to the data source.

To create a user mapping, use the DB2® Control Center. You can also issue the CREATE USER MAPPING statement in the DB2 Command Center or in the command line processor (CLP).

Not only do you need to create a user mapping for yourself, but you need to create user mappings for other groups or individuals who will be accessing the data source.

Use the CREATE USER MAPPING statement to map the local user ID to the data source server user ID and password; for example:

```
CREATE USER MAPPING FOR authorization_name SERVER server_name
OPTIONS (REMOTE_AUTHID 'remote_authorization_name',
REMOTE_PASSWORD 'remote_password')
```

where:

authorization_name
: Is the local authorization name that a user or application connects to the federated database. The local user ID that you are mapping to a user ID defined at the data source server.

SERVER *server_name*
: Is the name of the data source server that you defined in the CREATE SERVER statement.

REMOTE_AUTHID *'remote_authorization_name'*
: Is the remote authorization name that a user or application uses to connect to at the data source server. This value is case sensitive unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

REMOTE_PASSWORD *'remote_password'*
: Is the password associated with *'remote_authorization_name'*. This value

is case sensitive unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

Using Sybase as the data source, an example of the CREATE USER MAPPING statement is:

```
CREATE USER MAPPING FOR maria SERVER SYBSERVER
OPTIONS (REMOTE_AUTHID 'mary', REMOTE_PASSWORD 'day2night')
```

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. The following is an example of the CREATE USER MAPPING statement which includes the **USER** special register:

```
CREATE USER MAPPING FOR USER SERVER SYBSERVER
OPTIONS (REMOTE_AUTHID 'mary', REMOTE_PASSWORD 'day2night')
```

If you attempt to retrieve the REMOTE_PASSWORD associated with a user mapping from SYSCAT.USEROPTIONS catalog view, the REMOTE_PASSWORD value is displayed encrypted.

After you create the user mapping, it is a good idea to test the connection to the data source server. This will ensure that you can establish a connection that uses the server definition and user mappings you defined. To test the connection, open a pass-through session and issue a SELECT statement against the data source system tables. For example, if Sybase is the data source:

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

Related tasks:

- "Setting up the server to access DB2 family data sources" on page 44
- "Setting up the server to access Informix data sources" on page 47
- "Setting up the server to access Oracle data sources" on page 50
- "Setting up the server to access Sybase data sources" on page 54
- "Setting up the server to access Microsoft SQL Server data sources" on page 57
- "Setting up the server to access ODBC data sources" on page 62
- "Setting up the server to access OLE DB data sources" on page 65
- "User Mappings : Federated Systems help" in the *Help: Federated Systems*
- "Creating user mappings: Federated Systems help" in the *Help: Federated Systems*

Related reference:

- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*
- Appendix D, "User options for federated systems" on page 297

Create nicknames for each data source object

After you create the user mapping, identify the data source objects you want to access. Data source objects are typically database tables, views, and synonyms (Informix only). Using Life Sciences Data Connect, you can access other data source objects. For example: 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,

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 non-relational 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" in the *Help: Federated Systems*
- "Filtering tables and views for creating nicknames : Federated Systems help" in the *Help: Federated Systems*
- "Filtering tables for creating nicknames: Federated Systems help" in the *Help: Federated Systems*
- "Creating nicknames: Federated Systems help" in the *Help: Federated Systems*

Related reference:

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

Optional configuration steps

The previous sections describe the basic steps necessary to configure the federated server and database to access your data sources. There are other actions you can take to customize or tune the data source configuration, such as:

- 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.

Related concepts:

- "Define alternative data type mappings to the federated database" on page 101
- "Specify data source object indexes" on page 99
- "Define alternative function mappings to the federated database" on page 103

Related tasks:

- "Creating an index: Control Center help" in the *Help: Control Center*

About optional configuration steps

Specify data source object indexes

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

- A nickname is created for a table that has no index.
- A nickname is created for a data source object that does not contain indexes such as a view, Informix™ synonym, table-structured file, Documentum Docbase file, Excel spreadsheet, BLAST algorithm, or XML tagged file.
- The remote index is on a column of more than 255 bytes, or contains a total key length in excess of 1024 bytes.
- The remote index is on a LOB column.
- You want to encourage the query optimizer to use a specific nickname as the inner table of a nested loop join. You can create an index on the joining column if none exists.

However, you can supply the necessary index information to the global catalog, by creating an index specification.

To create an index specification, use the DB2® Control Center. Alternatively, you can issue the CREATE INDEX statement in the DB2 Command Center or in the command line processor (CLP). You can also create an index specification by embedding the CREATE INDEX statement in an application program.

The syntax for an index specification is:

```
CREATE UNIQUE INDEX index_name ON nickname
(column_name ASC/DESC) SPECIFICATION ONLY
```

where:

UNIQUE
: UNIQUE is an optional setting and should only be specified if the data for the index key contains unique values for every row in the data source object. The uniqueness will not be checked. You may get incorrect results if you specify an index as UNIQUE when it is not unique.

index_name
: Is the name you give the index.

ON *nickname*
: Is the nickname for the data source object.

(column_name ASC/DESC)
: Is the name by which the federated server references the column of the data source table, and the order (ascending or descending) of the column value index entries.

SPECIFICATION ONLY
: Indicates that an index specification will be created in the federated global catalog, not an actual table index.

Suppose that you create the nickname *jp_sales* for a table called JAPAN_SALES. Any indexes the table had at the time the nickname was created are logged in the federated global catalog. Later, a new index is added to the table. The new index uses the MARKUP column for the index key. To provide the information about this new index to the global catalog, the CREATE INDEX statement you create is:

```
CREATE INDEX jp_markup ON jp_sales (MARKUP) SPECIFICATION ONLY
```

where *jp_markup* is the name you give the index specification that is created in the global catalog.

Suppose that you create the nickname *employee* for a data source table called CURRENT_EMP. At the time the nickname was created, there were not indexes for that table. Later, an index is defined on the CURRENT_EMP table using the WORKDEPT and JOB columns for the index key. To provide the information about this new index to the global catalog, the CREATE INDEX statement you create is:

```
CREATE INDEX job_by_dept ON employee (WORKDEPT, JOB) SPECIFICATION ONLY
```

where *job_by_dept* is the name you give the index specification that is created in the global catalog. The specification indicates that the index entries are in ascending order by job title (JOB) within each department (WORKDEPT).

Note: If the data source table is a local table—a table on the federated server—the CREATE INDEX statement you use will be slightly different. Substitute the table name for the nickname, and omit the SPECIFICATION ONLY parameter.

Related tasks:

- "Creating index specifications for data source objects" on page 216
- "Nickname characteristics affecting global optimization" on page 249
- "Creating an index: Control Center help" in the *Help: Control Center*

Related reference:

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

Define alternative data type mappings to the federated database

When you submit a query to the federated server, the server checks for information about the data type mappings between DB2® and the data source. The two places the federated server looks for this information are:

- The wrapper. The data source wrapper contains the default data type mappings.
- The SYSCAT.TYPEMAPPINGS catalog view. This view contains entries you create that override the default type mappings that are in the wrapper.

Use the CREATE TYPE MAPPING statement to add alternative data type mapping entries into the SYSCAT.TYPEMAPPINGS catalog view.

With the CREATE TYPE MAPPING statement, you can indicate what the new mapping applies to:

- All data sources of a specific type. For example, all Oracle data sources.
- A specific data source server. For example, a server defined as ORASERVER to the federated database.

- A specific data source object. For example, a specific table that the Marketing department in your organization uses.
- All data sources of a specific type and version. For example, all Oracle 8.0.3 data sources.

You can issue the CREATE TYPE MAPPING statement in the DB2 Command Center or in the command line processor (CLP). You can also embed the CREATE TYPE MAPPING statement in an application program. The DB2 Control Center does not support creating or modifying data type mappings.

For example, by default the Oracle data type NUMBER maps to the DB2 data type DOUBLE, a floating decimal data type. Suppose that you want all Oracle tables and views that use the Oracle data type NUMBER to map to DB2 DECIMAL (8,2). The CREATE TYPE MAPPING statement would be:

```
CREATE TYPE MAPPING MY_ORACLE_DEC FROM SYSIBM.DECIMAL (8,2)
TO SERVER TYPE ORACLE TYPE NUMBER
```

where:

MY_ORACLE_DEC
: is the name you give to the type mapping. The name cannot duplicate a type mapping name already described in the catalog. It must be unique.

FROM *SYSIBM.DECIMAL (8,2)*
: is the local DB2 schema and data type. If the length or precision (and scale) are not specified, then these values are determined from the source data type.

TO SERVER TYPE *ORACLE*
: identifies the type of data source.

TYPE *NUMBER*
: is the remote data source type that you are mapping to the local data type. This must be a built-in data type. User-defined types are not allowed. If the type has a short and long form, specify the short form.

Related tasks:

- "Modifying default data type mappings" on page 208

Related reference:

- Appendix H, "Default forward data type mappings" on page 307

Define alternative function mappings to the federated database

When you submit queries to the federated server which contain one or more functions, the server checks for information about the mappings between the DB2® functions and the data source functions. The federated server checks two places for this information:

- The wrapper. The data source wrapper contains the default function mappings.
- The SYSCAT.FUNCMAPPINGS catalog view. This view contains entries you create that override or augment the default function mappings that are in the wrapper.

The function mappings are one of several important inputs to the pushdown analysis performed by the DB2 SQL Compiler. The SQL Compiler considers whether the data source can perform a particular type of SQL function or operation. If the function does not have a mapping, the function will no be sent to the data source for processing.

Use the CREATE FUNCTION MAPPING statement to add alternative function mapping entries into the SYSCAT.FUNCMAPPINGS and SYSCAT.FUNCMAPOPTIONS catalog views. With the CREATE FUNCTION MAPPING statement, you can indicate what the new mapping applies to:

- All data sources of a specific type. For example, all Informix™ data sources.
- A specific data source object. For example, a specific view that all employees in your organization use.
- All data sources of a specific type and version. For example, all Informix 9 data sources.

You can issue the CREATE FUNCTION MAPPING statement in the DB2 Command Center or in the command line processor (CLP). You can also embed the CREATE FUNCTION MAPPING statement in an application program. The DB2 Control Center does not support creating or modifying function mappings.

When you create a function mapping, you are mapping a data source function and a counterpart function at the federated database. When no DB2 counterpart exists, or when you want to force the federated server to use the data source function, you can create a function template to act as the counterpart.

The DB2 counterpart function can be either a complete function or a function template. A function template is a DB2 function you create for the purpose of forcing the federated server to invoke a data source function. However, unlike

a regular function, a function template has no executable code. The function template is created with the CREATE FUNCTION statement using the AS TEMPLATE parameter, for example:

```
CREATE FUNCTION BONUS () RETURNS DECIMAL(8,2) AS TEMPLATE
```

where:

BONUS ()
is the name you give to the function.

RETURNS *DECIMAL(8,2)*
is the data type of the output.

AS TEMPLATE
indicates this is a function template.

After you create a function template, you must then create the function mapping between the template and the data source function. When the federated server receives queries specifying the function template, it will invoke the data source function. A function mapping is created using the CREATE FUNCTION MAPPING statement, for example:

```
CREATE FUNCTION MAPPING MY_INFORMIX_FUN FOR BONUS ()
SERVER TYPE INFORMIX OPTIONS (REMOTE_NAME 'BONUS')
```

where:

MY_INFORMIX_FUN
is the name you give to the function mapping. The name cannot duplicate a function mapping name already described in the DB2 catalog. It must be unique.

FOR *BONUS ()*
is the local DB2 function template name. Include data type input parameters in parenthesis.

SERVER TYPE *INFORMIX*
identifies the type of data source which contains the function to which you want to map.

OPTIONS *(REMOTE_NAME 'BONUS')*
is an option that identifies the name of the remote data source function that you are mapping to the local DB2 function.

Most function mapping options provide information to the query optimizer about the potential cost of executing a function at the data source. Suppose that pushdown analysis determines that either the data source function or the DB2 function can be called. The statistical information you provide through

the function mapping options helps the query optimizer to compare the estimated cost of executing the data source function with the estimated cost of executing the DB2 function.

Function mapping options provide information such as:

- Name of the remote data source function.
- The estimated number of instructions processed the first and last time that the data source function is invoked.
- Estimated number of I/Os performed the first and last time that the data source function is invoked.
- Estimated number of instructions processed per invocation of the data source function.

Related concepts:

- "Function mappings and function templates" on page 20
- "Pushdown analysis" on page 233

Related tasks:

- "Creating and modifying function mappings" on page 223

Related reference:

- Appendix F, "Function mapping options for federated systems" on page 301

Troubleshoot the data source configuration

There are several steps you can take to avoid set up problems and configuration problems. Probably most significant is to test the connection to the data source between key steps:

- After you install and configure the data source client software, test the connection to the data source. This will isolate configuration problems to the client software set up and the DB2® custom installation. If you cannot connect to the data source using the client software, check the following:
  - The data source server is running.
  - The appropriate listener (for instance TCPIP listener) is configured and running.
  - The user ID and password in the user mapping can access the data source.
  - The federated server can access the data source server at the network software layer. For instance, for TCP/IP, ping the data source host system from the federated server.
  - The data source client configuration information at the federated server file matches the corresponding configuation information at the data

source. For instance, the information in the Oracle tnsnames.ora file on federated server matches the information in the listener.ora file at Oracle data source.

  - Logged into the federated server as the DB2 instance owner, connect to the data source using the data source cleint software. Specify in the connetion the user ID and password of the user mpaping to the data source.
  - Check that the environment variables are appropriate for federated server to use the client software and access the client configuration file. These variables include the: system environment variables, db2dj.ini variables (on UNIX), and db2set variables.

- After you create the server definitions and the user mappings, test the connection to the data source. This will isolate configuration problems to the server definitions and the user mappings.
- When you create the nicknames, the federated server will attempt to connect to the data source object. This isolates configuration problems to the nickname.

Related concepts:

- "Pushdown analysis" on page 233

Related tasks:

- "Tuning and troubleshooting the configuration to DB2 family data sources" on page 115
- "Tuning and troubleshooting the configuration to Informix" on page 125
- "Tuning and troubleshooting the configuration to Microsoft SQL Server data sources" on page 155
- "Tuning and troubleshooting the configuration to Oracle data sources" on page 135
- "Tuning and troubleshooting the configuration to Sybase data sources" on page 145
- "Global optimization" on page 246

Chapter 5. Configuring access to DB2 family data sources

This chapter explains how to configure your federated server to access data stored in DB2 family databases — DB2 for UNIX and Windows; DB2 for z/OS and OS/390; DB2 for iSeries; and DB2 Server for VM and VSE. It contains two sections:

- Adding DB2 family data sources to a federated server
- Tuning and troubleshooting the configuration to DB2 family data sources

Adding DB2 family data sources to a federated server

Configuring the federated server to access DB2 data sources involves supplying the server with information about the DB2 data sources and objects you want to access. You can configure access to DB2 data sources two ways

- Through the DB2 Control Center
- Through the DB2 Command Center or command line processor (CLP)

The advantage of using the DB2 Control Center is that you do not have to key in each statement and command. It is the easiest way to quickly configure access to DB2 data sources. There are a few configuration tasks that you cannot accomplish through the DB2 Control Center:

- Catalog the node
- Catalog the remote database
- Testing the connection to the data source server which validates the server definition and user mappings.
- Adding or dropping column options.

The steps in this section assume that you are using the DB2 Command Center or the command line processor (CLP) to configure access to DB2 data sources.

Prerequisites:

- A federated server and database that are setup to access DB2 family data sources.
- The proper variables setup. This includes: system environmetn variables, db2dj.ini variables (UNIX only), and DB2 Profile Registry (db2set) variables.

The steps to accomplish these tasks are discussed in Setting up a federated server and database.

Restrictions:

In Version 8.1, you can not create a nickname for a DB2 data source alias.

Procedure:

To add a DB2 data source to a federated server, you need to:

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

These steps are explained in detail in this section. Operating system-specific differences are noted where they occur.

Step 1: Catalog a node entry in the federated node directory

To point to the location of the DB2 data source, you catalog an entry in the node directory of the federated server. This entry is used by the federated server to determine the proper access method it will use to connect to a DB2 data source. For example, if TCP/IP is your communication protocol issue the CATALOG TCP/IP NODE command:

```
CATALOG TCPIP NODE DB2NODE REMOTE SYSTEM42 SERVER DB2TCP42
```

where:

DB2NODE
Is a 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 SNA is your communication protocol, issue the CATALOG APPC NODE command:

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

where:

DB2NODE
is a 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 sent to the partner LU.

Step 2: Catalog the remote database in the federated system database directory

You inform the federated server which DB2 data source database to connect to by cataloging the remote database in the federated system database directory. Use the Client Configuration Assistant (CCA) to catalog the remote database. For federated servers on UNIX , you can also use the CATALOG DATABASE command.

Note: Do not use the CATALOG DCS DATABASE command to catalog the remote database.

An example of cataloging the remote database using the CATALOG DATABASE command is:

```
CATALOG DATABASE DB2DB390 AS CLIENTS390 AT NODE DB2NODE AUTHENTICATION SERVER
```

where:

DB2DB390
Is a name of the database you are cataloging.

AS *CLIENTS390*
Is an alias for the database being cataloged.

AT NODE *DB2NODE*
Is the name of the node you specified when cataloging the node entry in the node directory.

AUTHENTICATION SERVER
Specifies that authentication takes place on the DB2 data source node.

Step 3: Create the wrapper

To specify the wrapper that will be used to access DB2 data sources, use the CREATE WRAPPER statement. Every DB2 Server Edition (Enterprise, Personal, Workgroup) includes one wrapper for the DB2 family called DRDA. The following example shows a CREATE WRAPPER statement:

```
CREATE WRAPPER DRDA
```

Recommendation: Use the default wrapper name (DRDA). When you create the wrapper using one of the default names, the federated server automatically picks up the default library name 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 than 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 use is:

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

where *mywrapper* is the name you give to the wrapper instead of using the default wrapper name.

The wrapper library names for DB2 are:

Table 17. 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

Step 4: Create the server definition

In the federated database, you must define each DB2 server that you want to access. When you create 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 You create a server definition using CREATE SERVER statement. For example:

```
CREATE SERVER DB2SERVER TYPE DB2/ZOS VERSION 5 WRAPPER DRDA
AUTHORIZATION "spalten" PASSWORD "db2guru" OPTIONS (DBNAME 'CLIENTS390')
```

where:

DB2SERVER
: Is a name 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. See the list of valid server types in Valid server types in SQL Statements.

VERSION *5*
: Is the version of the DB2 database server that you want to access.

  - The DB2 for UNIX and Windows the supported versions are 6, 7.1, 7.2, 8.1.

- The DB2 for z/OS and OS/390 the supported versions are 5 (or later).
- The DB2 for iSeries the supported versions are 4 (or later)

WRAPPER *DRDA*
The name 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 associated with the authorization ID at the data source. This value is case-sensitive.

DBNAME *'CLIENTS390'*
The alias for the DB2 database that you want to access. You defined this alias when you cataloged the database using the CATALOG DATABASE command. This value is case-sensitive.

Although the database name is specified as an option in the CREATE SERVER statement, it is required for DB2 data sources.

When you issue the CREATE SERVER statement, it will test the connection to the DB2 data source server. If you receive an error, you will need to packages to the database manually

Optional: Set additional server options

When you create the server definition, you can specify additional server options in the CREATE SERVER statement. There are general server options and data source-specific server options.

With DB2 data sources, a useful option to set is PUSHDOWN. If you set PUSHDOWN to 'Y', the federated server will consider letting the DB2 data source evaluate operations. This is the default setting. If you set PUSHDOWN to 'N', the federated server will only retrieve columns from the remote data source and will not let the data source evaluate other operations, such as joins.

An example of the CREATE SERVER statement with this server options is:

```
CREATE SERVER DB2SERVER TYPE DB2/ZOS VERSION 5 WRAPPER DRDA
AUTHORIZATION "spalten" PASSWORD "db2guru"
OPTIONS (PUSHDOWN 'N')
```

Once the server definition is created, use the ALTER SERVER statement to apply additional server options.

Step 5: Create the user mappings

When the you request access to a DB2 server, access is granted if the authorization IDs are the same between the federated database and the DB2 server. If a user's authorization ID to access the federated database differs from the user's authorization ID to access a data source, you need to define an association, a user mapping, between the two authorization IDs so that distributed requests can be sent to the data source.

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

Use the CREATE USER MAPPING statement to map the local user ID to the DB2 server user ID and password; for example:

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

where:

DB2USER
: Is the local user ID that you are mapping to a user ID defined at a DB2 family data source server.

SERVER *DB2SERVER*
: Is the name of the DB2 family data source server that you defined in the CREATE SERVER statement.

REMOTE_AUTHID *'db2admin'*
: Is the connect authorization user ID at the DB2 family data source server to which you are mapping *DB2USER*. This value is case sensitive unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

REMOTE_PASSWORD *'day2night'*
: Is the password associated with *'db2admin'*. This value is case sensitive unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

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. The following is an example of the CREATE USER MAPPING statement which includes the **USER** special register:

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

Step 6: Test the connection to the data source server

Test the connection to the DB2 server to ensure that you can establish a connection, using the server definition and user mappings you defined. Open

a pass-through session and for DB2 for z/OS and OS/390, issue a SELECT statement against the DB2 system tables. For example:

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

For DB2 for iSeries, issue a SELECT statement against the DB2 system tables. For example:

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

If the SELECT returns a count, then your server definition and user mapping are set up properly. If the SELECT returns an error, you may have to drop and recreate or modify the server definition or user mapping.

If the SELECT returns a count, then your server definition and user mapping are set up properly. If the SELECT returns an error, you may have to:

- Check the remote server to make sure 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 node and database.
- Check your DB2 federated variables to make sure that they are correct for working with the remote DB2 server. This includes the system environment variables, db2dj.ini variables, and DB2 Profile Registry (db2set) DB2COMM variable.
- Check your server definition and possibly drop it and create it again.
- Check your user mapping and possibly alter it or create another if necessary.

Step 7: Create the nicknames for the tables and views

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 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 RUNSTATS) at the data source before you create a nickname.

Use the CREATE NICKNAME statement to assign a nickname to a view or table located at your DB2 family data source. You will use these nicknames, instead of the names of the data source objects, when you query the DB2 family data source. Nicknames can be up to 128 characters in length. The following example shows a CREATE NICKNAME statement:

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

where:

DB2SALES
: Is a unique nickname used to identify the DB2 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.

DB2SERVER.SALESDATA.EUROPE
: Is a three-part identifier for the remote object.

  - *DB2SERVER* is the name 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 which you want to access.

Repeat this step for each DB2 table or view that you want to create nicknames for. When you create the nickname, the federated server will use the connection to query the data source catalog. This query tests your connection to the data source using the nickname. If the connection does not work, you will receive an error message.

Related tasks:

- "Fast track to setting up your server and database" on page 39
- "Tuning and troubleshooting the configuration to DB2 family data sources" on page 115
- "Setting up the server to access DB2 family data sources" on page 44

Related reference:

- "CATALOG DATABASE Command" in the *Command Reference*
- "CATALOG DCS DATABASE Command" in the *Command Reference*
- "CATALOG TCP/IP NODE Command" in the *Command Reference*
- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*
- Appendix C, "Server options for federated systems" on page 287

Tuning and troubleshooting the configuration to DB2 family data sources

Once you have set up the configuration to DB2 data sources, you may 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 DB2 data source is accessed.

Improving performance by setting the DB2_DJ_COMM environment variable (UNIX)

If you find that it takes an inordinate amount of time to access the DB2 data source server, you can improve the performance by setting the DB2_DJ_COMM environment 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 environment variable:

1. Set the DB2_DJ_COMM environment 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 environment variable.

Table 18. Commands to set the DB2_DJ_COMM variable for DB2 data sources

Federated server operating system	Command
AIX	DB2_DJ_COMM= 'libdb2drda.a'
Solaris	DB2_DJ_COMM= 'libdb2drda.so'
HP-UX	DB2_DJ_COMM= 'libdb2drda.sl'
Linux	DB2_DJ_COMM= 'libdb2drda.so'
Windows NT and Windows 2000	DB2_DJ_COMM= 'db2drda.dll'

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

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

2. Then export the DB2_DJ_COMM environment variable:

   ```
   export DB2_DJ_COMM
   ```

3. To ensure that the environment variables are set in the program, recycle the DB2 instance. When you recycle the instance, you refresh the DB2 instance to accept the changes that you made. Issue the following commands to recycle the DB2 instance:

```
db2stop
db2start
```

Chapter 6. Configuring access to Informix data sources

This chapter explains how to configure your federated server to access data stored in Informix databases. It contains two sections:

- Adding Informix data sources to a federated server
- Tuning and troubleshooting the configuration to Informix data sources

Adding Informix data sources to a federated server

Configuring the federated server to access Informix data sources involves supplying the server with information about the Oracle data sources and objects you want to access. You can configure access to Informix data sources two ways:

- Through the DB2 Control Center
- Through the DB2 Command Center or command line processor (CLP)

The advantage of using the DB2 Control Center is that you do not have to key in each statement and command. It is the easiest way to quickly configure access to Informix data sources. There are a few configuraton tasks that can not be accomplished through the DB2 Control Center:

- 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.

The steps in this section assume that you are using the DB2 Command Center or the command line processor (CLP) to configure access to Informix data sources.

Prerequisites:

- A federated server and database that are setup to access Informix data sources.
- The Informix Client SDK software installed and configured on the federated server.
- The proper variables setup. This includes: system environmetn variables, db2dj.ini variables (UNIX only), and DB2 Profile Registry (db2set) variables.

The steps to accomplish these tasks are discussed in Setting up a federated server and database.

Procedure:

To add an Informix data source to a federated server:

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

These steps are explained in detail in this section. The operating system-specific differences are noted where they occur.

Step 1: Set up and test the client configuration file

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 the default name is `$INFORMIXDIR/etc/sqlhosts`. The `sqlhosts` file resides on each installation of the Informix client SDK.
- On Windows the default location of the `sqlhosts` registry is the local computer.

The format of `sqlhosts` is described in the *Administrator's Guide for Informix Dynamic Server*.

There are several ways to create `sqlhosts` file or registry. You can copy it 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 federate server will use the `sqlhosts` that is in the Informix SDK directory or the Windows registry.

The location of the `sqlhosts` file or registry depends on the operating system you are running on your federated server.

- On UNIX, the `sqlhosts` file is located in the `$INFORMIXDIR/etc/sqlhosts` directory.
- On Windows, the sqlhosts information is kept in the following key in the Windows registry:

  `HKEY_LOCAL_MACHINE\SOFTWARE\INFORMIX\SQLHOSTS`

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 set up.

Setting a different location for the sqlhosts file or registry

On Unix, if the `sqlhosts` file is not in the default location, then you must set environment variable INFORMIXSQLHOSTS to the fully-qualified name of the `sqlhosts` file. On Windows, if the `sqlhosts` registry is not on the local computer, then you must set environment variable INFORMIXSQLHOSTS to the name of the Windows computer that stores the registry.

To ensure that the environment variable is set in the program, recycle the DB2 instance. Issue the following commands to recycle the DB2 instance:

```
db2stop
db2start
```

Step 2: Create the wrapper

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`. The following example shows a CREATE WRAPPER statement:

```
CREATE WRAPPER INFORMIX
```

Recommendation: Use the default wrapper name `INFORMIX`. When the wrapper is created using the default name, the federated server automatically picks up the default library name associated with the wrapper. If the wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name you choose. If you use a name that is different than the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement. Suppose that you have a federated server that is running AIX and you decide to use a wrapper name that is not one of the default names. An example of the CREATE WRAPPER statements for `INFORMIX` is:

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

The wrapper library names for Informix are:

Table 19. Informix wrapper library names

Operating system on your federated server	Wrapper library name
AIX	libdb2informix.a
HP-UX	libdb2informix.sl
Linux	libdb2informix.so

Table 19. Informix wrapper library names (continued)

Operating system on your federated server	Wrapper library name
Solaris Operating Environment	libdb2informix.so
Windows NT and Windows 2000	db2informix.dll

Step 3: Create the server definition

In the federated database, you must define each Informix server that you want to access. You create a server definition using CREATE SERVER statement. For example:

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

where:

asia Is 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*
Is 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.

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

Locating the node name

You must define the node name in the Informix `sqlhosts` file (see step 1). Although the *node_name* is specified as an option in the CREATE SERVER SQL statement, it is required for Informix data sources. This is an example of a `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. 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 forth value in each line is the service name, such as inmx724. The service name field depends on *nettype* listed in the second value.

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*.

Optional: Set additional server options

When you create the server definition, you can specify additional server options in the CREATE SERVER statement. There are general server options and data source-specific 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. 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 may result in the user ID being logged out. You can prevent this 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 need to always 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 won't keep trying the various combinations. This reduces the chance of exceeding the maximum number of failed login attempts and the ID getting locked out.

An example of the CREATE SERVER statement with these server options is:

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

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

Step 4: Create the user mappings

When you attempt to access an Informix server, access is granted if the authorization IDs are the same between the federated database and the Informix server. If a user's authorization ID to access the federated database differs from the user's authorization ID to access a data source, you need to define an association between the two authorization IDs. This association, referred to as a user mapping, ensures that distributed requests can be sent to the data source.

Use the CREATE USER MAPPING statement to map the local user ID to the Informix server user ID and password; for example:

```
CREATE USER MAPPING FOR VINCENT SERVER asia
OPTIONS (REMOTE_AUTHID 'vinnie', REMOTE_PASSWORD 'pasta8me')
```

where:

VINCENT
: Is the local user ID that you are mapping to a user ID defined at an Informix server.

SERVER *asia*
: Is the name of the Informix server that you defined in the CREATE SERVER statement.

REMOTE_AUTHID *'vinnie'*
: Is the user ID at the Informix database server to which you are mapping *VINCENT*. This value is case sensitive unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

REMOTE_PASSWORD *'pasta8me'*
: Is the password associated with *'vinnie'*. This value is case sensitive unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

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. The following is an example of the CREATE USER MAPPING statement which includes the **USER** special register:

```
CREATE USER MAPPING FOR USER SERVER asia
OPTIONS (REMOTE_AUTHID 'vinnie', REMOTE_PASSWORD 'pasta8me')
```

Step 5: Test the connection to the Informix server

Test the connection to the Informix server. This will ensure that you can establish a connection using the server definition and user mappings you defined. Open a pass-through session and issue a SELECT statement against the Informix system tables. For example:

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

If the SELECT returns a count then your server definition and user mapping are set up properly. If the SELECT returns an error you may have 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 connections to the Informix server.
- 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 your DB2 federated variables to make sure that they are correct for working with the Informix server. This includes the system environment variables, db2dj.ini variables, and DB2 Profile Registry (db2set) variable.
- Check your server definition and possibly drop it and create it again.
- Check your user mapping and possibly alter it or create another if necessary.

Step 6: Create the nicknames for tables, views, and synonyms

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 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 RUNSTATS) at the data source before you create a nickname.

For each Informix server you defined, assign a nickname to each table, view, or synonym you want to access on those servers. You will use these nicknames, instead of the names of the data source objects, when you query the Informix servers. Nicknames can be up to 128 characters in length. The

federated server will fold the Sybase server, schema, and table names to uppercase unless you enclose them in double quotation marks (″). The following example shows a CREATE NICKNAME statement:

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

where:

JPSALES
: Is 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 creating the nickname, the schema of the nickname will be the authid of the user creating the nickname.

asia."salesdata"."japan"
: Is a three-part identifier for the remote object.

  - *asia* is the name 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.

Repeat this step for each Informix table, view, or synonym for which you want to create nicknames. 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 using the nickname. If the connection does not work, you will receive an error message.

Related concepts:

- "User mappings and user options" on page 15
- "Nicknames and data source objects" on page 16
- "Index specifications" on page 22

Related tasks:

- "Fast track to setting up your server and database" on page 39
- "Setting up the server to access Informix data sources" on page 47

Related reference:

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

- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*
- Appendix C, "Server options for federated systems" on page 287
- Appendix D, "User options for federated systems" on page 297
- Appendix B, "Wrapper options for federated systems" on page 285

Tuning and troubleshooting the configuration to Informix

Once you have set up the configuration to Informix data sources, you may 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 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.. This means that the server might make up to nine connect attempts before successfully connecting to the data source server. This can slow down connect times. You can improve performance by specifying values for the FOLD_ID and FOLD_PW server options using the ALTER SERVER OPTION statement.

Procedure:

- 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 OPTION FOLD_ID FOR SERVER TYPE INFORMIX SETTING 'L'
ALTER SERVER OPTION FOLD_PW FOR SERVER TYPE INFORMIX SETTING 'L'
```

- Since the federated server attempts each combination of upper and lower case 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 / password). If you establish these settings, then you need to always specify the user ID / password in the correct case. If an invalid user ID / password is specified, the wrapper won't keep trying the various combinations. For example:

```
ALTER SERVER OPTION FOLD_ID FOR
SERVER TYPE INFORMIX SETTING 'N'

ALTER SERVER OPTION FOLD_PW FOR
SERVER TYPE INFORMIX SETTING 'N'
```

Improving performance by setting the DB2_DJ_COMM environment variable (UNIX)

If you find that it takes an inordinate amount of time to access the Informix server, you can improve the performance by setting the DB2_DJ_COMM environment 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 environment variable:

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

   ```
   db2set DB2_DJ_COMM='libdb2informix.a'
   ```

   Consult the following table for the proper library name.

Table 20. 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. Ensure that the environment variables are set in the program by recycling the DB2 instance. When you recycle the instance, you refresh the DB2 instance to accept the changes that you made. Issue the following commands to recycle the DB2 instance:

   ```
   db2stop
   db2start
   ```

Related concepts:

- "Environment Variables and the Profile Registry" in the *Administration Guide: Implementation*

Related tasks:

- "Setting up the server to access Informix data sources" on page 47

Related reference:

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

Chapter 7. Configuring access to Oracle data sources

This chapter explains how to configure your federated server to access data stored in Oracle databases. It contains two sections:

- Adding Oracle data sources to a federated server
- Tuning and troubleshooting the configuration to Oracle data sources

Adding Oracle data sources to a federated server

Configuring the federated server to access Oracle data sources involves supplying the server with information about the Informix data sources and objects you want to access. You can configure access to Oracle data sources two ways:

- Through the DB2 Control Center
- Through the DB2 Command Center or command line processor (CLP)

The advantage of using the DB2 Control Center is that you do not have to key in each statement and command. It is the easiest way to quickly configure access to Oracle data sources. There are a few configuraton tasks that can not be accomplished through the DB2 Control Center:

- 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.

The steps in this section assume that you are using the DB2 Command Center or the command line processor (CLP) to configure access to Oracle data sources.

Prerequisites:

- A federated server and database that are setup to access Oracle data sources.
- The Oracle client software installed and configured on the federated server.
- The proper variables setup. This includes: system environmetn variables, db2dj.ini variables (UNIX only), and DB2 Profile Registry (db2set) variables.

The steps to accomplish these tasks are discussed in Setting up a federated server and database.

Procedure:

To add an Oracle data source to a federated server:

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

These steps are explained in detail in this section. Operating system-specific differences are noted where they occur.

Step 1: Set up and test a client configuration file

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` .

To set up the client configuration file, 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 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.

Table 21. Default path and name of the Oracle client configuration file.

Operating system	Default path and name
UNIX	$ORACLE_HOME/network/admin
Windows	%ORACLE_HOME%\NETWORK\ADMIN

Test the connection to ensure that the client software is able to connect to the Oracle server. Use the Oracle **sqlplus** tool to test the connection.

Setting a different location for the tnsnames.ora file

If you decide to place the `tnsnames.ora` file in a path other than the default search path, you must set the TNS_ADMIN environment variable to specify the file location. To set this environment variable:

1. Edit the `db2dj.ini` file located in the `sqllib/cfg` directory, and set the TNS_ADMIN environment variable:

   ```
   TNS_ADMIN=x:\path\tnsnames.ora
   ```

2. To ensure that the environment variable is set in the program, recycle the DB2 instance. Issue the following commands to recycle the DB2 instance:

```
db2stop
db2start
```

Step 2: Create the wrapper

To specify the wrapper that will be used to access Oracle data sources, use the CREATE WRAPPER statement. DB2 Relational Connect includes two wrappers for Oracle — SQLNET and NET8. To determine which wrapper to use, consult the following table.

Table 22. 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	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

Note: 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. Since the OCI 7 does not support LOB data types, the SQLNET wrapper does not support the LOB data types. The SQLNET wrapper does not support Oracle LOB datatypes. The wrapper maps Oracle LONG datatypes to DB2 for UNIX and Windows LOB datatypes. The NET8 wrapper supports Oracle LOB datatypes, but does not Oracle LONG datatypes.

The following example shows the CREATE WRAPPER statement for the NET8 wrapper:

```
CREATE WRAPPER NET8
```

Recommendation: Use the default wrapper names (SQLNET or NET8). When you create the wrapper using one of the default names, the federated server automatically picks up the default library name associated with the wrapper. If the wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name you choose. If you use a name that is different than 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 mywrapperLIBRARY 'libdb2net8.a'
```

The wrapper library names for Oracle are:

Table 23. 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.so	libdb2net8.so
Linux	libdb2sqlnet.sl	libdb2net8.sl
Solaris Operating Environment	libdb2sqlnet.so	libdb2net8.so
Windows NT and Windows 2000	db2sqlnet.dll	db2net8.dll

Step 3: Create the server definition

In the federated database, you must define each Oracle server that you want to access. You create a server definition using CREATE SERVER statement. For example:

```
CREATE SERVER oraserver TYPE oracle VERSION 7.2 WRAPPER net8
OPTIONS (NODE 'paris_node')
```

where:

oraserver
: Is a name 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 *7.2*

Is the version of Oracle database server that you want to access. The supported Oracle versions are 7.x, 8.x, and 9.x.

WRAPPER *net8*

Is the name you specified in the CREATE WRAPPER statement.

NODE *'paris_node'*

Is 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.

Locating the node name

You must define the node name in the Oracle `tnsnames.ora` file (see step 1). Although the *node_name* is specified as an option in the CREATE SERVER statement, it is required for Oracle data sources. This is an example of a `tnsnames.ora` file:

```
ORA9I.SEEL =
  (DESCRIPTION =
   (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = somehost)(PORT = 1521)))
     (CONNECT_DATA =
      (SERVICE_NAME = ora9i.seel)))
```

The node value to use in the CREATE SERVER statement would be `ora9i.seel`.

Optional: Set additional server options

When you create the server definition, you can specify additional server options in the CREATE SERVER statement. There are general server options and data source-specific server options.

DB2 assumes that all of the Oracle VARCHAR columns contain trailing blanks. If you are certain that all VARCHAR columns in the Oracle database do not contain trailing blanks, you can set a server option to specify that the data source uses non-blank padded VARCHAR comparison semantic. An example of the CREATE SERVER statement with this server options is:

```
CREATE SERVER oraserver TYPE oracle VERSION 7.2 WRAPPER net8
OPTIONS (NODE 'paris_node', VARCHAR_NO_TRAILING_BLANKS 'Y')
```

Note: Suppose only some of the VARCHAR columns do not contain blanks. You can set an option on those specific columns with the CREATE NICKNAME or ALTER NICKNAME statements.

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

Step 4: Create the user mappings

When you attempt to access an Oracle server, access is granted if the authorization IDs are the same between the federated database and the Oracle server. If a user's authorization ID to access the federated database differs from the user's authorization ID to access a data source, you need to define an association — a user mapping — between the two authorization IDs so that distributed requests can be sent to the data source.

Use the CREATE USER MAPPING statement to map the local user ID to the Oracle server user ID and password; for example:

```
CREATE USER MAPPING FOR robert SERVER oraserver
OPTIONS (REMOTE_AUTHID 'rob', REMOTE_PASSWORD 'then4now')
```

where:

robert Is the local user ID that you are mapping to a user ID defined at an Oracle server.

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

REMOTE_AUTHID *'rob'*
Is the user ID at the Oracle database server to which you are mapping *robert*. This value is case sensitive unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

REMOTE_PASSWORD *'then4now'*
Is the password associated with *'rob'*. This value is case sensitive unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

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. The following is an example of the CREATE USER MAPPING statement which includes the **USER** special register:

```
CREATE USER MAPPING FOR USER SERVER oraserver
OPTIONS (REMOTE_AUTHID 'rob', REMOTE_PASSWORD 'then4now')
```

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.

Step 5: Test the connection to the Oracle server

Test the connection to the Oracle server to ensure that you can establish a connection, using the server definition and user mappings you defined. Open a pass-through session and issue a SELECT statement against the Oracle system tables. For example:

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

If the SELECT returns a count, then your server definition and user mapping are set up properly. If the SELECT returns an error, you may have 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 connections to the Oracle server.
- 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 your DB2 federated variables to make sure that they are correct for working with the Oracle server. This includes the system environment variables, db2dj.ini variables, and DB2 Profile Registry (db2set) variable.
- Check your server definition and possibly drop it and create it again.
- Check your user mapping and possibly alter it or create another if necessary.

Step 6: Create the nicknames for tables and views

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 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 RUNSTATS) at the data source before you create a nickname.

For each Oracle server you defined, assign a nickname to each table or view you want to access on those servers. You will use these nicknames, instead of the names of the data source objects, when you query the Oracle servers. Nicknames can be up to 128 characters in length.

The federated server will fold the Sybase server, schema, and table names to uppercase unless you enclose them in double quotation marks (″). The following example shows a CREATE NICKNAME statement:

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

where:

PARISINV
: Is 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 creating the nickname, the schema of the nickname will be the authid of the user creating the nickname.

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

  - *oraserver* is the name 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.

Repeat this step for each Oracle table or view for which you want create nicknames. 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 using the nickname. If the connection does not work, you will receive an error message.

Related concepts:

- "User mappings and user options" on page 15
- "Index specifications" on page 22

Related tasks:

- "Fast track to setting up your server and database" on page 39

Related reference:

- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*
- "CREATE SERVER statement" in the *SQL Reference, Volume 2*
- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*
- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*
- Appendix C, "Server options for federated systems" on page 287
- Appendix D, "User options for federated systems" on page 297
- Appendix B, "Wrapper options for federated systems" on page 285

Tuning and troubleshooting the configuration to Oracle data sources

Once you have set up the configuration to Oracle data sources, you may 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 Oracle data source is accessed.

Improving performance by setting the DB2_DJ_COMM environment variable (UNIX)

If you find that it takes an inordinate amount of time to access the Oracle server, you can improve the performance by setting the DB2_DJ_COMM environment variable. Setting the DB2_DJ_COMM environment variable will load the wrapper when the federated server initializes rather than when you attempt to access the data source.

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 NET8. The command to set the DB2_DJ_COMM environment variable is:

   ```
   db2db2set DB2_DJ_COMM= 'libdb2net8.a'
   ```

   Consult the following table for the proper library name.

Table 24. 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.so	libdb2net8.so
Linux	libdb2sqlnet.sl	libdb2net8.sl
Solaris Operating Environment	libdb2sqlnet.so	libdb2net8.so

2. Recycle the DB2 instance to ensure that the environment variables are set in the program. When you recycle the instance, the DB2 instance accepts the changes that you made. Issue the following commands to recycle the DB2 instance:

   ```
   db2stop
   db2start
   ```

Connectivity problems

For each HOST in the DESCRIPTION section of the `tnsnames.ora` file, you may 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 specified in the DESCRIPTION section in the `tnsnames.ora` file to an address. If your network has a name 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:

- "Setting up the server to access Oracle data sources" on page 50

Related reference:

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

Chapter 8. Configuring access to Sybase data sources

This chapter explains how to configure your federated server to access data stored in Sybase databases. It contains two sections:

- Adding Sybase data sources to a federated server
- Tuning and troubleshooting the configuration to Sybase data sources

Adding Sybase data sources to a federated server

Configuring the federated server to access Sybase data sources involves supplying the server with information about the Sybase data sources and objects you want to access. You can configure access to Sybase data sources two ways:

- Through the DB2 Control Center
- Through the DB2 Command Center or command line processor (CLP)

The advantage of using the DB2 Control Center is that you do not have to key in each statement and command. It is the easiest way to quickly configure access to Sybase data sources. There are a few configuration tasks that you cannot accomplish through the DB2 Control Center:

- 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.

The steps in this section assume that you are using the DB2 Command Center or the command line processor (CLP) to configure access to Sybase data sources.

Prerequisites:

- A federated server and database that are setup to access Sybase data sources.
- The Sybase client software installed and configured on the federated server.
- The proper variables setup. This includes: system environmetn variables, db2dj.ini variables (UNIX only), and DB2 Profile Registry (db2set) variables.

The steps to accomplish these tasks are discussed in Setting up a federated server and database.

Restrictions:

The DBLIB wrapper is a read-only wrapper, it 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. Create the wrapper.
3. Create the server definition and set the server options.
4. Create the user mappings.
5. Test the connection to the Sybase server.
6. Create nicknames for Sybase tables and views.

These steps are explained in detail in this section. The operating system-specific differences are noted where they occur.

Step 1: Set up and test the client configuration file

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.

To set up the client configuration file use the utility that comes with the Sybase Open Client software. Set up a client configuration file on each instance in the DB2 federated server that will be used to connect to Sybase. See the Sybase documentation for more information about using this utility.

The directory that the client configuration file is created in depends on the operating system you are running on your federated server.

Table 25. Default path and name of the Sybase client configuration file.

Operating system	Default path and name
UNIX	$SYBASE/interfaces
Windows	%SYBASE%\ini\sql.ini

Once you set up the client configuration file, you must make it accessible to the DB2 federated server instance.

On UNIX federated servers:

To make the `interfaces` accessble, use one of these options:

- Copy the file to the `$HOME/sqllib` directory of the DB2 federated instance.

- Use the **ln** command to create a link from the /sqllib sub-directory 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.

On Windows federated servers:

Copy the sql.ini file to the c:\Program Files\SQLLIB directory of the DB2 federated instance. Since DB2 Relational Connect uses interfaces as the default name for the Sybase client configuration file, it is recommended that you rename the Windows sql.ini file in the c:\Program Files\SQLLIB directory to interfaces..

Note: If you choose not to rename the sql.ini file to interfaces, you must use the IFILE server option when you create the server definition (see step 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**.

Step 2: Create the wrapper

To specify the wrapper that will be used to access Sybase data sources, use the CREATE WRAPPER statement. DB2 Relational Connect includes two wrappers for Sybase, the Open Client Client-Library wrapper called CTLIB and the Open Client DB-Library wrapper called DBLIB. The following example shows the CREATE WRAPPER statement for the Open Client Client-Library wrapper:

```
CREATE WRAPPER CTLIB
```

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

Recommendation: Use the default wrapper names (CTLIB or DBLIB). When you create the wrapper using one of the default names, the federated server automatically picks up the default library name associated with that the wrapper. If the wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name you choose. If you use a name that is different than 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 26. 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.so	libdb2dblib.so
Linux	libdb2ctlib.sl	libdb2dblib.sl
Solaris Operating Environment	libdb2ctlib.so	libdb2dblib.so
Windows NT and Windows 2000	db2ctlib.dll	db2dblib.dll

Step 3: Create the server definition

In the federated database, you must define each Sybase server that you want to access. You create a server definition using CREATE SERVER statement. For example:

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

where:

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

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

NODE *'sybnode'*
: Is 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'*

Is 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.

Locating the node name

You must define the node name in the Sybase `interfaces` file (see step 1). Although the node name is specified as an option in the CREATE SERVER SQL statement, it is required for Sybase data sources. The following table contains examples of the`interfaces` file to help you locate the node name:

Table 27. Locating the node name in the Sybase interfaces file.

Operating system	Sample interfaces files
AIX	`sybase119` `query tcp ether anaconda 4100`

Solaris Operating Environment	`sybase119`
	`query tli tcp /dev/tcp`
	`\x000210040970191c0000000000000000`

Windows NT or Windows 2000	`[sybase119]` `query=TCP,anaconda 4100`

The first value is the node name; sybase119 in these examples. This is followed by the type of connection (TCP/IP) and the host name (anaconda).

Optional: Set additional server options

When you create the server definition, you can specify additional server options in the CREATE SERVER statement. There are general server options and Sybase–specific server options. The Sybase-specific options are:

IFILE

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

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 indefinately 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 may 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 indefinately for a response from the Sybase server. The Sybase Open Client uses timeout thresholds to interrupt queries and responses that run for too long a period of time.

Examples:

Using theDBLIB wrapper, set the timeout value to the number of seconds the wrapper should wait for a response from the Sybase server. This will avoid deadlocks on transactions. To include the TIMEOUT server option in the server definition on UNIX servers use:

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

To include the IFILE server option in the server definition on Windows servers, use:

```
CREATE SERVER SYBSERVER TYPE SYBASE
VERSION 12.0 WRAPPER DBLIB
OPTIONS (NODE 'sybnode', DBNAME 'sybdb',
IFILE 'C:\Sybase\OCS-12_0\ini\sql.ini')
```

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

Step 4: Create the user mappings

When you attempt to access a Sybase server, access is granted if the authorization IDs are the same between the federated database and the Sybase server. If a user's authorization ID to access the federated database differs from the user's authorization ID to access a data source, you need to define an association — a user mapping — between the two authorization IDs so that distributed requests can be sent to the data source.

Use the CREATE USER MAPPING statement to map the local user ID to the Sybase server user ID and password; for example:

```
CREATE USER MAPPING FOR maria SERVER SYBSERVER
OPTIONS (REMOTE_AUTHID 'mary', REMOTE_PASSWORD 'day2night')
```

where:

maria Is the local user ID that you are mapping to a user ID defined at the Sybase server.

SERVER *SYBSERVER*
Is the name of the Sybase server that you defined in the CREATE SERVER statement.

REMOTE_AUTHID *'mary'*

Is the user ID at the Sybase server to which you are mapping *maria*. This value is case sensitive unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

REMOTE_PASSWORD *'day2night'*

Is the password associated with *'mary'*. This value is case sensitive unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

Note: 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. The following is an example of the CREATE USER MAPPING statement which includes the **USER** special register:

```
CREATE USER MAPPING FOR USER SERVER SYBSERVER
OPTIONS (REMOTE_AUTHID 'mary', REMOTE_PASSWORD 'day2night')
```

Step 5: Test the connection to the Sybase server

Test the connection to the Sybase server to ensure that you can establish a connection, using the server definition and user mappings you defined. Open a pass-through session and issue a SELECT statement against the Sybase system tables. For example:

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

If the SELECT returns a count, then your server definition and user mapping are set up properly. If the SELECT returns an error, you may have to:

- Check the Sybase server to make sure that it is configured for incoming connections.
- Check your user mapping to make sure thatthe settings for the remote_authid and remote_password options are valid for connections to the Sybase server.
- 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 your DB2 federated variables to make sure that they are correct for working with the Sybase server. This includes the system environment variables, db2dj.ini variables, and DB2 Profile Registry (db2set) variable.
- Check your server definition and possibly drop it and create it again.
- Check your user mapping and possibly alter it or create another if necessary.

Step 6: Create the nicknames for tables and views

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 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 RUNSTATS) at the data source before you create a nickname.

For each Sybase server that you defined, assign a nickname to each table or view that you want to access on those servers. You will use these nicknames, instead of the names of the data source objects, when you query the Sybase servers. Nicknames can be up to 128 characters in length.

The federated server will fold the Sybase server, schema, and table names to uppercase unless you enclose them in double quotation marks (″). 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 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.

Repeat this step for each Sybase table or view for which you want create nicknames. 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 using the nickname. If the connection does not work, you will receive an error message.

Related concepts:

- “User mappings and user options” on page 15

- "Nicknames and data source objects" on page 16
- "Index specifications" on page 22

Related tasks:

- "Fast track to setting up your server and database" on page 39
- "Modifying server definitions" on page 203

Related reference:

- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*
- "CREATE SERVER statement" in the *SQL Reference, Volume 2*
- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*
- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*
- Appendix C, "Server options for federated systems" on page 287
- Appendix D, "User options for federated systems" on page 297
- Appendix B, "Wrapper options for federated systems" on page 285

Tuning and troubleshooting the configuration to Sybase data sources

Once you have set up the configuration to Sybase data sources, you may 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 an inordinate amount of time to access the Sybase server, you can improve the performance by setting the DB2_DJ_COMM environment variable. Setting the DB2_DJ_COMM environment variable will 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 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 28. 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.so	libdb2dblib.so
Linux	libdb2ctlib.sl	libdb2dblib.sl
Solaris Operating Environment	libdb2ctlib.so	libdb2dblib.so

2. Recycle the DB2 instance to ensure that the environment variables are set in the program. When you recycle the instance, the DB2 instance accepts the changes that you made. Issue the following commands to recycle the DB2 instance:

```
db2stop
db2start
```

Using CTLIB instead of DBLIB

CT-Library supports dynamic prepare-and-execute of statements. This allows CT-Library applications to prepare a statement once and execute it many times with different inputs. Preparing a statement once eliminates the need to recompile the statement for each input parameter change. While the DB2 application may 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 may 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 a federated server" on page 137

Related reference:

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

Chapter 9. Configuring access to Microsoft SQL Server data sources

This chapter explains how to configure your federated server to access data stored in Microsoft SQL Server databases. It contains two sections:

- Adding Microsoft SQL Server data sources to a federated server
- Tuning and troubleshooting the configuration to Microsoft SQL Server data sources

Adding Microsoft SQL Server data sources to a federated server

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 two ways:

- Through the DB2 Control Center
- Through the DB2 Command Center or command line processor (CLP)

The advantage of using the DB2 Control Center is that you do not have to key in each statement and command. It is the easiest way to quickly configure access to Microsoft SQL Server data sources. There are a few configuration tasks that you cannot accomplish through the DB2 Control Center:

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

The steps in this section assume that you are using the DB2 Command Center or the command line processor (CLP) to configure access to Microsoft SQL Server data sources.

Prerequisites:

- A federated server and database that are setup to access Microsoft SQL Server data sources.
- The Microsoft SQL Server ODBC driver installed and configured on the federated server.
- The proper variables setup. This includes: system environmetn variables, db2dj.ini variables (UNIX only), and DB2 Profile Registry (db2set) variables.

The steps to accomplish these tasks are discussed in Setting up a federated server and database.

Procedure:

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

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

These steps are explained in detail in this section. The operating system-specific differences are noted where they occur.

Step 1: Prepare the federated server and database

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

On Windows:

It is important for you to verify that the ODBC System DSN has been set to connect to the Microsoft SQL Server data source. You can check this setting in the Control Panel by:

1. Open the Control Panel through the Start menu.
2. Double-click on ODBC Data Sources to display the ODBC Data Source Administrator window.
3. Click on the System DSN tab to confirm that the System DSN you defined for the ODBC driver appears on the list. The node name for the Microsoft SQL Server data source must be defined in the System DSN.

Test the connection to ensure that the ODBC Systems DSN is able to connect to the Microsoft SQL Server data source. You can test the connection by selecting **Configure** in the ODBC Data Source Administrator window. Alternatively, you can test the connection using the Microsoft SQL Server query tool.

On UNIX:

Make sure that the odbc.ini file has been updated (or if necessary created) on the federated server. It's recommended that the odbc.ini file, or a copy of it, be place in the home directory of the DB2 instance owner. Make sure that the path to the odbc.ini is in the ODBCINI environment variable.

Test the connection to ensure that the federated server is able to connect to the Microsoft SQL Server data source. Use the DataDirect Connect ODBC **demoodbc** tool to test the connection. This tool is located in the /demo sub-directory of the Connect ODBC libraries.

Step 2: Create the wrapper

To specify the wrapper that will be used to access Microsoft SQL Server data sources, use the CREATE WRAPPER statement. DB2 Relational Connect includes two wrappers for Microsoft SQL Server.

Table 29. Supported ODBC drivers and the default wrapper names

Federated server operating system	ODBC driver	Wrapper Name
UNIX	DataDirect Connect ODBC 3.7 driver	MSSQLODBC3
Windows	ODBC 3.0 (or higher) driver	DJXMSSQL3

The following example shows the CREATE WRAPPER statement for the Windows NT and Windows 2000 wrapper:

```
CREATE WRAPPER DJXMSSQL3
```

Recommendation:IBM recommends you that use the default wrapper names (DJXMSSQL3 and MSSQLODBC3). When you create the wrapper using one of the default names, the federated server automatically picks up the default library name 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 than 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 use is:

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

where *mywrapper* is the name you give to the wrapper instead of using the default wrapper name.

The wrapper library names for Microsoft SQL Server are:

Table 30. Microsoft SQL Server wrapper library names

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

Step 3: Create the server definition

In the federated database, you must define each Microsoft SQL Server remote server that you want to access. You create a server definition using CREATE SERVER statement. For example:

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

where:

sqlserver
: Is 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*
: Is 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*
: Is the version of Microsoft SQL Server database server software that you want to access. Supported versions are 6.5 and 7.0.

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

NODE *'sqlnode'*
: Is 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'*

Is 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.

Locating 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 the`.odbc.ini` file on AIX. 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] describing each node.

Table 31. Locating the node name in the .odbc.ini file.

Operating system	Sample .odbc.ini file

AIX	`rawilson=MS SQL Server 7.0`
	`medusa=MS SQL Server 7.0`
	`[rawilson]`
	`Driver=/djxclient/mssql/merant/3.7/lib/ivmsss16.so`
	`Description=MS SQL Server Driver for AIX`
	`Address=9.112.30.39,1433`
	`[medusa]`
	`Driver=/djxclient/mssql/merant/3.7/lib/ivmsss16.so`
	`Description=MS SQL Server Driver for AIX`
	`Address=9.112.98.123,1433`

Optional: Set additional server options

When you create the server definition, you can specify additional server options in the CREATE SERVER statement. There are general server options and data source-specific server options.

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 running Windows NT or Windows 2000, the default collating sequence is not case-sensitive (for example, 'STEWART' and 'StewART' are considered equal). To guarantee correct results from federated server, set the COLLATING_SEQUENCE server option to 'I'.

Note: When you set the COLLATING_SEQUENCE server option to 'I', the federated server will not pushdown the following:

- Queries that contain ORDER BY for character columns.

- Queries that contain GROUP BY for character columns.
- Queries that contain DISTINCT for character columns.
- Queries that contain WHERE= for character columns.
- Queries that contain WHERE< for character columns
- Queries that contain WHERE> for character columns

An example of the CREATE SERVER statement with these server options is:

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

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

Step 4: Create the user mappings

When you attempt to access a Microsoft SQL Server remote server, access is granted if the authorization IDs are the same between the federated database and the Microsoft SQL Server data source remote server. If a user's authorization ID to access the federated database differs from the user's authorization ID to access a data source, you need to define an association —a user mapping— between the two authorization IDs so that distributed requests can be sent to the data source.

Use the CREATE USER MAPPING statement to map the local user ID to the Microsoft SQL Server server user ID and password; for example:

```
CREATE USER MAPPING FOR elizabeth SERVER sqlserver
OPTIONS (REMOTE_AUTHID 'liz', REMOTE_PASSWORD 'abc123')
```

where:

elizabeth
: Is the local user ID that you are mapping to a user ID defined at the Microsoft SQL Server remote server.

SERVER *sqlserver*
: Is the name of the Microsoft SQL Server remote server that you defined in the CREATE SERVER statement.

REMOTE_AUTHID *'liz'*
: Is the user ID at the Microsoft SQL Server remote server to which you are mapping *elizabeth*. This value is case sensitive unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

REMOTE_PASSWORD *'abc123'*
: Is the password associated with *'liz'*. This value is case sensitive unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

Note: 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. The following is an example of the CREATE USER MAPPING statement which includes the **USER** special register:

```
CREATE USER MAPPING FOR USER SERVER sqlserver
OPTIONS (REMOTE_AUTHID 'liz', REMOTE_PASSWORD 'abc123')
```

Step 5: Test the connection to the Microsoft SQL Server remote server

Test the connection to the Microsoft SQL Server remote server to ensure that you can establish a connection, using the server definition and user mappings you defined. Open a pass-through session and issue a SELECT statement against the Microsoft SQL Server system tables. For example:

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

If the SELECT returns a count, then your server definition and user mapping are set up properly. If the SELECT returns an error, you may have to:

- Check that the Microsoft SQL Server remote server 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 connections to the Microsoft SQL Server remote server.
- Check the ODBC drivers on the DB2 federated server to make sure that it is installed and configured correctly to connect to the Microsoft SQL Server remote server.
- Check your DB2 federated variables to make sure that they are correct for working with the Microsoft SQL Server server. This includes the system environment variables, db2dj.ini variables, and DB2 Profile Registry (db2set) variable.
- Check your server definition and possibly drop it and create it again.
- Check your user mapping and possibly alter it or create another if necessary.

Step 6: Create the nicknames for tables and views

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 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 RUNSTATS) at the data source before you create a nickname.

For each Microsoft SQL Server remote server that you defined, assign a nickname to each table or view that you want to access on those servers. You will use these nicknames, instead of the data source object names, when you query the Microsoft SQL Server data sources. Nicknames can be up to 128 characters in length.

The federated server will fold the Microsoft SQL Server server, schema, and table names to uppercase unless you enclose them in double quotation marks (″). The following example shows a CREATE NICKNAME statement:

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

where:

cust_egypt
: Is a unique nickname for the Microsoft SQL Server 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 authentication ID of the user creating the nickname.

sqlserver.customers.egypt
: Is a three-part identifier for the remote object.

  - *sqlserver* is the name 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 which you want to access.

Repeat this step for each Microsoft SQL Server table or view for which you want create nicknames. When you create the nickname, DB2 will use the connection to query the data source catalog tables (Microsoft SQL Server refers to these as system tables). This query tests your connection to the data source using the nickname. If the connection does not work, you will receive an error message.

Related concepts:

- “User mappings and user options” on page 15
- “Nicknames and data source objects” on page 16
- “Index specifications” on page 22

Related tasks:

- "Fast track to setting up your server and database" on page 39
- "Setting up the server to access Microsoft SQL Server data sources" on page 57

Related reference:

- Appendix C, "Server options for federated systems" on page 287
- Appendix D, "User options for federated systems" on page 297
- Appendix B, "Wrapper options for federated systems" on page 285

Tuning and troubleshooting the configuration to Microsoft SQL Server data sources

Once you have set up the configuration to Microsoft SQL Server data sources, you may 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 federated server accesses the Microsoft SQL Server data source.

Improving performance by setting the DB2_DJ_COMM environment variable (UNIX)

If you find that it takes an inordinate amount of time to access the Microsoft SQL Server remote server, you can improve the performance by setting the DB2_DJ_COMM environment variable. Setting the DB2_DJ_COMM environment variable will 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 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 MSSQLODBC3. The command to set the DB2_DJ_COMM environment variable is:

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

   Consult the following table for the proper library name.

Table 32. 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.so	none

Table 32. Microsoft SQL Server wrapper library names (continued)

Operating system on your federated server	MSSQLODBC3 wrapper library names	DJXMSSQL3 wrapper library names
Linux	libdb2mssql3.sl	none
Solaris Operating Environment	libdb2mssql3.so	none
Windows NT and Windows 2000	none	djxmssql3.dll

2. Recycle the DB2 instance to ensure that the environment variables are set in the program. When you recycle the instance, the DB2 instance accepts the changes that you made. Issue the following commands to recycle the DB2 instance:

```
db2stop
db2start
```

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 your system performance, therefore you should turn off tracing once you have resolved 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 federated servers, you need to 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 in the `.odbc.ini` file. For example, suppose you are using the DataDirect ODBC 3.x driver. There should be an example of the `.odbc.ini` file in the client directory. This file that contains a sample of what is needed for trace files:

```
[ODBC]
Trace=0
TraceFile=/home/user1/trace_dir/filename.xxx
TraceDll=/djxclient/mssql/merant/3.7/lib/odbctrac.so
InstallDir=/djxclient/mssql/merant/3.7
```

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

Related concepts:

- “Environment Variables and the Profile Registry” in the *Administration Guide: Implementation*

Related reference:

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

Chapter 10. Configuring access to ODBC data sources

This chapter explains how to configure your federated server to access data stored in ODBC data sources. It contains two sections:

- Adding ODBC data sources to a federated server
- Tuning and troubleshooting the configuration to ODBC data sources

Adding ODBC sources to a federated server

DB2 Relational Connect and DB2 Life Sciences Data Connect contain wrappers that support specific ODBC data sources, such as Oracle, Microsoft SQL Server, and Microsoft Excel. You will experience better performance if you use the wrappers specifically designed for those data sources.

Use the ODBC wrapper to access ODBC data sources that are not supported by DB2 Relational Connect and DB2 Life Sciences Data Connect.

The ODBC wrapper supports the Version 3.0 ODBC driver.

Configuring the federated server to access ODBC data sources involves supplying the server with information about the ODBC data sources and objects that you want to access. You can configure access to ODBC data sources two ways:

- Through the DB2 Control Center
- Through the DB2 Command Center or command line processor (CLP)

The advantage of using the DB2 Control Center is that you do not have to key in each statement and command. It is the easiest way to quickly configure access to ODBC data sources. There are a few configuration tasks that you cannot accomplish through the DB2 Control Center:

- Testing the connection to the ODBC server to validate the server definition and user mappings.
- Adding or dropping column options.

The steps in this section assume that you are using the DB2 Command Center or the command line processor (CLP) to configure access to ODBC data sources.

Prerequisites:

- A federated server and database that are setup to access ODBC data sources.

- The ODBC driver installed and configured on the federated server.
- The proper variables setup. This includes: system environmetn variables, db2dj.ini variables (UNIX only), and DB2 Profile Registry (db2set) variables.

The steps to accomplish these tasks are discussed in Setting up a federated server and database.

Restrictions:

- The ODBC wrapper is only supported on federated servers that use the Windows operating system.
- The ODBC wrapper might not function properly with data sources that do not use schemas, such as Microsoft Excel, Microsoft Access, and IBM Red Brick. There is a Microsoft Excel wrapper available with DB2 Life Sciences Data Connect.

Procedure:

To add an ODBC data source to a federated server:

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

These steps are explained in detail in this section. The operating system-specific differences are noted where they occur.

Step 1: Prepare the federated server and database

It is important for you to verify that the ODBC System DSN has been set to connect to the ODBC data source. Use the ODBC Data Source Administrator to configure the DSN. You can check this setting through the Control Panel:

1. Open the Control Panel through the Start menu.
2. Double-click on ODBC Data Sources to access the ODBC device manager.
3. Click on the System DSN tab to confirm that the System DSN 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.

Test the connection to ensure that the ODBC Systems DSN is able to connect to the ODBC data source. You can test the connection by selecting **Configure** in the ODBC Data Source Administrator window.

Step 2: Create the wrapper

To specify the wrapper that will be used to access ODBC data sources, use the CREATE WRAPPER statement. The following example shows a CREATE WRAPPER statement:

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

where:

odbc_wrapper
The name you assign to the wrapper that is being registered in the federated database.

LIBRARY *'db2rcodbc.dll'*
Specifies the library name for the ODBC wrapper. The LIBRARY name must be *'db2rcodbc.dll'*.

You need to create the ODBC wrapper only once regardless of the number of different ODBC data sources you will access. You will specify the data source location when you create the server definition. You will specify the exact data source object when you create the nickname.

Step 3: Create the server definition

In the federated database, you must define each ODBC data source server that you want to access. You create a server definition using CREATE SERVER statement. For example:

```
CREATE SERVER mssql_cust TYPE odbcserver
VERSION 3.0 WRAPPER odbc_wrapper
OPTIONS (NODE 'mssql')
```

where:

mssql_cust
Is a name that you assign to the ODBC data source server. This name must be unique. Duplicate server names are not allowed.

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

VERSION *3.0*
Is the version of ODBC driver that you want to access. The supported version is ODBC driver Version 3.0.

WRAPPER *odbc_wrapper*
Is the wrapper name you specified in the CREATE WRAPPER statement.

NODE *'mssql'*
The name of the node, the system DSN name, assigned to the ODBC data source when the DSN is defined. This value is case sensitive.

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

Step 4: Create the user mappings

When you attempt to access an ODBC data source, access is granted if the authorization IDs are the same between the federated database and the ODBC data source. If a user's authorization ID to access the federated database differs from the user's authorization ID to access a data source, you need to define an association—a user mapping—between the two authorization IDs so that distributed requests can be sent to the data source.

Use the CREATE USER MAPPING statement to map the local user ID to the ODBC data source user ID and password; for example:

```
CREATE USER MAPPING FOR elizabeth SERVER mssql_cust
OPTIONS (REMOTE_AUTHID 'liz', REMOTE_PASSWORD 'abc123')
```

where:

elizabeth
Is the local user ID that you are mapping to a user ID defined at the ODBC data source.

mssql_cust
Is the name of the ODBC data source that you defined in the CREATE SERVER statement.

'liz' Is the user ID at the ODBC data source to which you are mapping *elizabeth*. This value is case sensitive unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

'abc123'
Is the password associated with *'liz'*. This value is case sensitive unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

Note: 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. The following is an example of the CREATE USER MAPPING statement which includes the **USER** special register:

```
CREATE USER MAPPING FOR USER SERVER mssql_cust
OPTIONS (REMOTE_AUTHID 'liz', REMOTE_PASSWORD 'abc123')
```

Step 5: Test the connection to the ODBC data source

Test the connection to the ODBC data source server to ensure that you can establish a connection, using the server definition and user mappings you defined. Open a pass-through session and issue a SELECT statement against the ODBC data source system tables. For example:

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

where `schema_name` is the schema name at the remote ODBC data source, and `table_name` is the table name at the remote ODBC data source.

If the SELECT returns a count, then your server definition and user mapping are set up properly. If the SELECT returns an error, you may have to:

- Check that the ODBC remote server is started.
- Check the ODBC 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 connections to the ODBC data source.
- Check the ODBC drivers on the DB2 federated server to make sure that it is installed and configured correctly to connect to the ODBC data source server. Use the ODBC Data Source Administrator tool to check the driver.
- Check your server definition and possibly drop it and create it again.
- Check your user mapping and possibly alter it or create another if necessary.

Step 6: Create the nicknames for tables and views

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 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 RUNSTATS) at the data source before you create a nickname.

For each ODBC data source server that you defined, assign a nickname to each table or view that you want to access on those servers. You will use these nicknames, instead of the data source object names, when you query the ODBC data sources. Nicknames can be up to 128 characters in length.

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 *mssql_cust."customers"."italy"* is not allowed.

Use the CREATE NICKNAME statement to register nicknames for the ODBC data source tables and views you want to access, for example:

```
CREATE NICKNAME cust_europe FOR mssql_cust."customers"."italy"
```

cust_europe
: Is a unique nickname for the table or view. The nickname must be unique within the schema.

  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 authentication ID of the user creating the nickname.

mssql_cust."customers"."italy"
: Is a three-part identifier for the remote object.

  - *mssql_cust* is the name 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.
  - *italy* is the name of the remote table or view which you want to access.

ODBC nicknames are case sensitive. Enclose both the remote `schema_name` and `table_name` in double quotation marks (″), otherwise DB2 will fold the server, remote `schema_name`, and remote `table_name` to uppercase.

Repeat this step for each ODBC table or view for which you want to create nicknames. 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 using the nickname. If the connection does not work, you will receive an error message.

Related concepts:

- “Nicknames and data source objects” on page 16

Related tasks:

- “Fast track to setting up your server and database” on page 39
- “Setting up the server to access ODBC data sources” on page 62

Related reference:

- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*
- "CREATE SERVER statement" in the *SQL Reference, Volume 2*
- "CREATE USER MAPPING statement" in the *SQL Reference, Volume 2*
- "CREATE WRAPPER statement" in the *SQL Reference, Volume 2*
- Appendix E, "Column options for federated systems" on page 299
- Appendix C, "Server options for federated systems" on page 287
- Appendix D, "User options for federated systems" on page 297
- Appendix B, "Wrapper options for federated systems" on page 285

Tuning and troubleshooting the configuration to ODBC data sources

Once you have set up the configuration to ODBC data sources, you may 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 federated server accesses the ODBC data source.

Improving performance by setting the DB2_DJ_COMM environment variable

If you find that it takes an inordinate amount of time to access the ODBC remote server, you can improve the performance by setting the DB2_DJ_COMM environment variable. Setting the DB2_DJ_COMM environment variable will 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 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 uses Windows NT and the wrapper you are using is ODBC_WRAPPER. The command to set the DB2_DJ_COMM environment variable is:

   ```
   db2set DB2_DJ_COMM='db2rcodbc.dll'
   ```

   The DB2_DJ_COMM environment variable will be added to the Windows Registry.
2. Recycle the DB2 instance to ensure that the environment variables are set. When you recycle the instance, the DB2 instance accepts the changes that you made. Issue the following commands to recycle the DB2 instance:

   ```
   db2stop
   db2start
   ```

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 your system performance, therefore you should turn off tracing once you have resolved 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.

Related concepts:

- "Environment Variables and the Profile Registry" in the *Administration Guide: Implementation*

Related tasks:

- "Setting up the server to access ODBC data sources" on page 62

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 stored in OLE DB data sources. You can create table functions to query the OLE DB data sources. The chapter contains two sections:

- Adding ODE DB data sources to a federated server
- Registering a user-defined OLE DB external table function

Adding OLE DB data sources to a federated server

Microsoft OLE DB is a set of OLE/COM interfaces that provide applications with uniform access to data stored in diverse information sources. The OLE DB component DBMS architechture definesOLE DB consumers and OLE DB providers. An OLE DB consumer is any system or application that consumes OLE DB inerfeaces. 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). Configuring the federated server to access OLE DB data sources, involves supplying the server with information about the OLE DB providers.

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.

You can configure access to OLE DB data sources through the DB2 Command Center or command line processor (CLP).

Prerequisites:

- A federated server and database that are setup to access OLE DB data sources.
- The OLE DB 2.0 (or later) driver and OLE DB provider installed and configured on the federated server.
- The proper variables setup. This includes: system environmetn variables, db2dj.ini variables (UNIX only), and DB2 Profile Registry (db2set) variables.

The steps to accomplish these tasks are discussed in Setting up a federated server and database.

Restrictions:

The OLE DB wrapper is supported on DB2 federated servers that run Windows NT.

Procedure:

To add an OLE DB data source to a federated server:

1. Create the wrapper.
2. Create the server definition and set the server options.
3. Create the user mappings.

These steps are explained in detail in this section.

Step 1: Create the wrapper

To specify the wrapper that will be used to access OLE DB data sources, use the CREATE WRAPPER statement. The following example shows the CREATE WRAPPER statement for the OLE DB wrapper:

```
CREATE WRAPPER OLEDB
```

Recommendation: IBM recommends that you use the default wrapper name (OLEDB). When you create the wrapper using the default name, the federated server automatically picks up the default library name associated with that the wrapper. If the wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name you choose. If you use a name that is different than one of the default names, you must include the LIBRARY parameter in the CREATE WRAPPER statement. For example:

```
CREATE WRAPPER mywrapper LIBRARY 'db2oledb.dll'
```

Step 2: Create the server definition

In the federated database, you must define each OLE DB server that you want to access. You create a server definition using CREATE SERVER statement. For example:

```
CREATE SERVER Nwind WRAPPER OLEDB
OPTIONS (CONNECTSTRING 'Provider=Microsoft.Jet.OLEDB.4.0;
    Data Source=c:\msdasdk\bin\oledb\nwind.mdb',
COLLATING_SEQUENCE 'Y')
```

where:

Nwind Is a name that you assign to the OLE DB data source. This name must be unique. Duplicate server names are not allowed.

WRAPPER *OLEDB*
Is 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 needed to connect to a data source.

The string contains a series of keyword and value pairs separated by semicolons. The equal sign(=) separates each keyword and it's 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 DB2 for UNIX and Windows.

Valid values are ″Y″ (the same collating sequence is used) and 'N' (a different collating sequence is used). If the COLLATING_SEQUENCE option is not specified, it is assumed that the data source has a different collating sequence than the DB2 collating sequence.

Step 3: Create the user mappings

When you attempt to access a OLE DB data source, access is granted if the authorization IDs are the same between the federated database and the OLE DB server. If a user's authorization ID to access the federated database differs from the user's authorization ID to access a data source, you need to define an association —a user mapping— between the two authorization IDs. This will enable the federated database to send distributed requests to the data source.

Use the CREATE USER MAPPING statement to map the local user ID to the OLE DB server user ID and password; for example:

```
CREATE USER MAPPING FOR laura SERVER Nwind
OPTIONS (REMOTE_AUTHID 'lulu', REMOTE_PASSWORD '4t9ers')
```

where:

laura Is the local user ID that you are mapping to a user ID defined at the OLE DB data source.

SERVER *Nwind*

Is the name of the OLE DB server that you defined in the CREATE SERVER statement.

REMOTE_AUTHID *'lulu'*

Is the user ID at the OLE DB server to which you are mapping *laura*. This value is case-sensitive.

REMOTE_PASSWORD *'4t9ers'*
Is the password associated with *'lulu'*. This value is case-sensitive.

Note: 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. The following is an example of the CREATE USER MAPPING statement which includes the **USER** special register:

```
CREATE USER MAPPING FOR USER SERVER Nwind
OPTIONS (REMOTE_AUTHID 'lulu', REMOTE_PASSWORD '4t9ers')
```

Related concepts:

- "Object Linking and Embedding Database (OLE DB) Table Functions" in the *Application Development Guide: Building and Running Applications*

Related tasks:

- "Fast track to setting up your server and database" on page 39
- "Registering a user-defined OLE DB external table function" on page 170

Registering a user-defined OLE DB external table function

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.

Prerequisites:

A federated server and database that are set up to access OLE DB data sources. The OLE DB 2.0 (or higher) driver and OLE DB provider installed and configured on the federated server.

A federated server and database that are configured to access OLE DB data sources. Configuring the federated server involves: creating the wrapper, creating the server definition, and creating the user mappings.

Restrictions:

OLE DB table functions can be created on any operating system, but can only be executed on operating systems supported by Microsoft OLE DB.

Procedure:

To access data from an OLE DB provider, use the CREATE FUNCTION statement to register a user-defined OLE DB external table function. When

you create the function, use the server name you supplied in the CREATE SERVER statement to identify the OLE DB provider. For example:

```
CREATE FUNCTION orders () RETURNS
TABLE (orderid INTEGER, ...)
LANGUAGE OLEDB EXTERNAL NAME 'Nwind!orders'
```

where:

orders ()
Is the name you give to the function.

RETURNS TABLE *(orderid INTEGER)*
Specifies the output of the function is a table and lists the column names and data types of the output.

LANGUAGE *OLEDB*
Indicates the DB2 federated database manager will deploy a built-in generic OLE DB consumer to retrieve data from the OLE DB provider.

EXTERNAL NAME *'Nwind!orders'*
Identifies the external table and an OLE DB provider.

The parameter specified is used to establish a connection and session with a OLE DB provider and to retrieve data from a rowset.

Related tasks:

- "Adding OLE DB data sources to a federated server" on page 167
- "Fast track to setting up your server and database" on page 39

Related reference:

- Appendix C, "Server options for federated systems" on page 287

Part 3. Using, administering, and programming the federated system

Chapter 12. Working with the federated system

This chapter describes how to access and update data at the data sources. The topics in this chapter are:

- Working with nicknames
- Transaction support in a federated system
- Selecting data in a federated system
- Modifying data in a federated system

Working with nicknames

When you want to select or modify data source data, you query the nicknames using the SELECT, INSERT, UPDATE, and DELETE statements. You submit queries in DB2 SQL to the federated database. You can join data from local tables and remote data sources using a single SQL statement, as if all the data is local. For example, you can join data that is located in:

- A local DB2 for Windows table in the federated database, an Oracle table, and a Sybase view.
- A DB2 for z/OS table on one server, a DB2 for z/OS table on another server, and an Excel spreadsheet.

By processing SQL statements as if the data sources were ordinary relational tables or views within the federated database, the federated system can join relational data with data in non-relational formats.

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 objects not located in the federated database. For example:

- 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.

Procedure:

Before you query the data sources, make sure that you understand how to effectively leverage the capabilities of the federated system:

1. SQL statements you can use with nicknames
2. Accessing new data source objects
3. Accessing data sources using PASSTHRU
4. Accessing heterogeneous data through federated views

Related concepts:

- "Create the user mappings and test the connection to the data source" on page 94

Working with nicknames—details

The SQL statements you can use with nicknames

A federated system is designed to make it easy to access data, regardless of where it is actually stored. This is accomplished by creating nicknames for all the data source objects (such as tables and views) that you want to access.

For example, if the nickname DEPT is created to represent the remote table EUROPE.PERSON.DEPT, you would use the statement SELECT * FROM DEPT to query information in the remote table. You are querying the nickname instead of having to remember the underlying data source information. When you create a query, you do not have to be concerned with issues like:

- The name of the table at the data source.
- The server on which it resides.
- The type of DBMS on which the table resides, such as Informix and Oracle.
- The query language or SQL dialect that the DBMS uses.
- The data types mappings between the data source and DB2.

All the underlying metadata stored in the federated database catalog as part of the federated system setup and configuration, provide the federated server with the information it needs to process your queries.

After the federated system is set up you can use the nicknames to query the data sources, or further enhance the federated system configuration.

The following table lists the SQL statements which support the use of nicknames:

Table 33. Common SQL statements that support the use of nicknames.

SQL statement	Description	Authorization required
ALTER NICKNAME	Modifies an existing nickname by changing the local column name, the local data type, or the federated column options. The table or view at the data source is not affected.	• SYSADM or DBADM • ALTER or CONTROL privilege on the nickname. • ALTERIN privilege on the schema, if the schema name of the nickname exists • Definer of the nickname, as recorded in the DEFINER column of the catalog view for the nickname
COMMENT ON	Adds or replaces comments in the catalog descriptions of various objects, including: functions, function mappings, indexes, nicknames, servers, server options, type mappings, wrappers.	• SYSADM or DBADM • ALTER or CONTROL privilege on the object. • ALTERIN privilege on the schema • Definer of the object, as recorded in the DEFINER column of the catalog view for the object.
CREATE ALIAS	Defines an alias for a nickname.	• SYSADM or DBADM • IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name of the alias does not exist. • CREATEIN privilege on the schema, if the schema name of the alias refers to an existing schema.
CREATE INDEX	Used to create an index specification (metadata) that indicates to the query optimizer that a data source object has an index. No actual index is created, only the specification.	• SYSADM or DBADM • Or — CONTROL or INDEX privilege on the underlying data source object. • And either — IMPLICIT_SCHEMA authority on the database or CREATEIN privilege on the schema.

Table 33. Common SQL statements that support the use of nicknames. (continued)

SQL statement	Description	Authorization required
DELETE	Deletes rows from the data source object (such as a table or view) that a nickname has been created for.	• SYSADM or DBADM • DELETE privilege on the nickname and DELETE privilege on the underlying data source object. • CONTROL privilege on the underlying data source object.
DROP	Deletes an object, such as a nickname, federated view, index specification. The table, view, or index at the data source is not affected.	• SYSADM or DBADM • DROPIN privilege on the schema for the object. • CONTROL privilege on the object.
GRANT	Grants privileges on nicknames and federated views, such as ALTER, DELETE, INDEX, INSERT, SELECT, UPDATE.	• SYSADM or DBADM • WITH GRANT OPTION for each identified privilege. • CONTROL privilege on the object.
INSERT	Inserts rows into the data source object (such as a table or view) that a nickname has been created for.	• SYSADM or DBADM • INSERT privilege on the nickname and INSERT privilege on the underlying data source object. • CONTROL privilege on the underlying data source object.
LOCK TABLE	Will cause the remote object at the data source to be locked. Prevents concurrent application processes from changing a data source table that a nickname has been created for.	• SYSADM or DBADM • SELECT privilege on the underlying table. • CONTROL privilege on the underlying table.
REVOKE	Revokes privileges on nicknames and federated views, such as ALTER, DELETE, INDEX, INSERT, SELECT, UPDATE.	• SYSADM or DBADM • CONTROL privilege on the object.

Table 33. Common SQL statements that support the use of nicknames. (continued)

SQL statement	Description	Authorization required
SELECT	Selects rows from the data source object (such as a table or view) that a nickname has been created for.	• SYSADM or DBADM • SELECT privilege on the nickname and SELECT privilege on the underlying data source object. • CONTROL privilege on the underlying data source object.
UPDATE	Updates the values in specified columns in rows in the data source object (such as a table or view) that a nickname has been created for.	• SYSADM or DBADM • UPDATE privilege on the nickname and UPDATE privilege on the underlying data source object. • CONTROL privilege on the underlying data source object.

When a query is submitted to the federated database, the authorization privileges on the nickname in the query are checked. The authorization requirements of the data source object referenced by the nickname are only applied when the query is actually processed.

To select, insert, update, or delete data using a nickname, the privileges held by the authorization ID of the statement must include:

- The appropriate privilege on the nickname (for the federated database to accept the request)
- The appropriate privilege on the underlying table object (for the data source to accept the request)

For example to update a data source using a nickname, you need UPDATE privilege on the nickname and UPDATE privilege on the underlying data source object.

Related reference:

- "COMMENT statement" in the *SQL Reference, Volume 2*
- "CREATE ALIAS statement" in the *SQL Reference, Volume 2*
- "CREATE INDEX statement" in the *SQL Reference, Volume 2*
- "DELETE statement" in the *SQL Reference, Volume 2*
- "DROP statement" in the *SQL Reference, Volume 2*
- "GRANT (Database Authorities) statement" in the *SQL Reference, Volume 2*
- "INSERT statement" in the *SQL Reference, Volume 2*

- "LOCK TABLE statement" in the *SQL Reference, Volume 2*
- "REVOKE (Table, View, or Nickname Privileges) statement" in the *SQL Reference, Volume 2*
- "SELECT statement" in the *SQL Reference, Volume 2*
- "UPDATE statement" in the *SQL Reference, Volume 2*
- "ALTER NICKNAME statement" in the *SQL Reference, Volume 2*

Accessing new data source objects

Periodically, you will want to access data source objects that do not have nicknames. These might be new objects added to a data source, such as a newly created view. These might be existing objects that were not registered with the federated server when it was initially setup. In either case, these objects are new to the federated server. To access these new objects, you need to create nicknames for them using the CREATE NICKNAME statement.

Prerequisites:

The federated system needs to be configured to access the data source. A server definition for the data source server on which the object resides needs to exist in the federated database. You create a server definition using CREATE SERVER statement.

Restrictions:

You must have one of the following authorizations to issue the CREATE NICKNAME statement:

- SYSADM or DBADM
- IMPLICIT_SCHEMA authority on the federated database, if the implicit or explicit schema name of the nickname does not exist.
- CREATEIN privilege on the schema, if the schema name of the nickname exists

And the remote user ID in your user mapping must have SELECT privilege at the data source.

The following example shows a CREATE NICKNAME statement:

```
CREATE NICKNAME nickname_name FOR server_name."remote_schema"."object_name"
```

where:

nickname_name
Is a unique nickname for the data source object.

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 authentication ID of the user creating the nickname. Nicknames can be up to 128 characters in length.

server_name."remote_schema"."object_name"
Is a three-part identifier for the remote data source object.

- *server_name* is the name assigned to the data source server in the CREATE SERVER statement.
- *remote_schema* is the name of the remote schema to which the object belongs.
- *object_name* is the name of the remote object that you want to access.

Related concepts:

- "Fast track to configuring your data sources" on page 85

Accessing data sources using PASSTHRU

You can submit SQL statements directly to data sources by using a special mode called *pass-through*. You submit SQL statements in the SQL dialect used by the data source. Use a pass-through session when you want to perform an operation that is not possible with the DB2 SQL/API. For example, use a pass-through session to create a procedure, create an index, or perform queries in the native dialect of the data source.

Note: Currently, the data sources that support pass-through, only accept SQL statements in a pass-through session. In the future, it is possible that data sources will support pass-though using a data source language other than SQL.

Similarly, you can use a pass-through session to perform actions that are not supported by SQL, such as certain administrative tasks. However, you cannot use a pass-through session to perform all administrative tasks. For example, you can run the statistics utility used by the data source, but you cannot start or stop the remote database.

You can query only one data source at a time in a pass-through session. Use the SET PASSTHRU command to open a session. When you use the SET PASSTHRU RESET command it closes the pass-through session. If you use the SET PASSTHRU command instead of SET PASSTHRU RESET, the current pass-through session is closed and a new pass-through session is opened.

Use the DB2 CLP to open a pass-through session.

Pass-through sessions do not support non-relational data sources.

Related concepts:

- "Pass-through sessions" on page 11
- "Using pass-through to query data sources directly" on page 277

Accessing heterogeneous data through federated views

A federated view is a view in the federated database whose base tables are located at remote data sources. The base tables are referenced in the federated view by nicknames, instead of by the data source table names. When you query from a federated view, data is retrieved from the remote data source. The action of creating a federated database view of data source data is sometimes called "creating a view on a nickname". This is because you reference the nicknames instead of the data sources when you create the view.

These views offer a high degree of data independence for a globally integrated database, just as views defined on multiple local tables do for centralized relational database managers.

Use the CREATE FEDERATED VIEW statement to create a federated view.

You must have one of the following authorizations to issue the CREATE FEDERATED VIEW statement:

- SYSADM or DBADM
- Or for each nickname in any fullselect:
  - CONTROL or SELECT privilege on the underlying table or view
  - and at least one of the following:
    - IMPLICIT_SCHEMA authority on the federated database, if the implicit or explicit schema name of the view does not exist.
    - CREATEIN privilege on the schema, if the schema name of the view refers to an existing schema.

Privileges for the underlying objects are not considered when defining a view on a federated database nickname. Authorization requirements of the data source for the table or view referenced by the nickname are applied when the query is processed. The authorization ID of the statement may be mapped to a different remote authorization ID by a user mapping.

Federated views that are created from more than one nicknamed data source object are read-only views.

Federated views that are created from only one nicknamed data source object may or may not be read-only views. A federated view created from one non-relational data source is read only. A federated view created from a

relational data source might allow updates, depending on what is included in the CREATE FEDERATED VIEW statement.

Related concepts:

- "Create the user mappings and test the connection to the data source" on page 94

Related tasks:

- "Creating and using federated views" on page 261

Transaction support in a federated system

Before you submit transactions to the federated database, it is important that you understand the type of transactions supported in a federated system.

Single-site updates and two-phase commit:

A transaction is commonly referred to in DB2® as a *unit of work*. A unit of work is a recoverable sequence of operations within an application process. A unit of work is used by the database manager to ensure that a database is in a consistent state. Any reading from or writing to the database is done within a unit of work. A point of consistency (or commit point) is a time when all recoverable data that an application accesses is consistent with related data.

A unit of work is implicitly begun when any data in the data base is read from or written to. An application must end a unit of work by issuing either a COMMIT or a ROLLBACK statement. The COMMIT statement makes permanent all changes made within a unit of work. The ROLLBACK statement removes these changes from the database.

Changes made by the unit of work become visible to other applications after a successful COMMIT. If the application ends normally without either of these statements being explicitly issued, the unit of work is automatically committed.

Recommendation: Your applications should always explicitly commit or roll back units of work. If an application ends abnormally in the middle of a unit of work, the unit of work is automatically rolled back.

A transaction can involve one or more databases. A transaction that involves two or more databases is a distributed unit of work (DUOW). In a DUOW that involves reading from one or more databases to update another database, or in a non-distributed unit of work, each COMMIT is processed in one operation. Accordingly, the operation is called a *one-phase commit*.

In a DUOW involving updates of multiple databases, data consistency is important. The *two-phase commit* protocol is commonly used to ensure data consistency across multiple databases within a DUOW. Two-phase commit will be supported on federated systems in a future release.

Using a federated system without two-phase commit support:

This table shows the location where an update is supported by the type of transaction you want to perform. For example, if you want to perform a transaction in a PASSTHRU session, you will only be able to update remote data source objects. A PASSTHRU session cannot be used to update local data source objects.

Table 34. Federated update matrix

	Location where the update is supported	
Type of update you want to perform	**Local update**	**Remote update**
Local update	Y	N
Transparent DDL	Y	Y
Remote INSERT, UPDATE, DELETE	N	Y
PASSTHRU	N	Y

This table shows the next type of update supported based on the last update you performed. For example, suppose you just performed an INSERT on a remote data source. Your next action can be another remote INSERT, UPDATE, or DELETE operations on the same data source server. You can also open a PASSTHRU session to perform an update on that same data source server. However, you can not perform a local update since that would require you to change to another server. Before you can peform update operations on another server, you must issue a COMMIT or ROLLBACK statement.

Table 35. Federated compatability

	Next type of update permitted on the same server before you must issue a COMMIT or ROLLBACK statement			
Last type of update you performed	**Local update**	**Transparent DDL**	**Remote INSERT, UPDATE, DELETE**	**PASSTHRU**
Local update	Y	N	N	N
Transparent DDL	N	N	N	N

Table 35. Federated compatability (continued)

	Next type of update permitted on the same server before you must issue a COMMIT or ROLLBACK statement			
Last type of update you performed	**Local update**	**Transparent DDL**	**Remote INSERT, UPDATE, DELETE**	**PASSTHRU**
Remote INSERT, UPDATE, DELETE	N	N	Y	Y
PASSTHRU	N	N	Y	Y

Note: Transparent DDL is not compatible with any other operation.

If the current unit of work included an update, include a COMMIT or ROLLBACK statement at the end of the unit of work. The COMMIT or ROLLBACK must be made before proceeding with another unit of work or to another data source.

Considerations with Transparent DDL:

COMMIT or ROLLBACK statements need to be issued before and after transparent DDL transactions. Transparent DDL creates a table on a remote data source and creates a nickname in the local federated database for the remote table. Because transparent DDL is updating both local and remote objects at the same time, each transparent DDL statement has to be the only update within the transaction. If there is any update prior to the transparent DDL transaction, a COMMIT or ROLLBACK statement has to be issued before the transparent DDL transaction. Likewise, a COMMIT or ROLLBACK statement has to be issued after the transparent DDL transaction, before any other update can occur.

Considerations with PASSTHRU:

All statements sent through PASSTHRU sessions are treated as updates by the federated server. The purpose of this is to ensure data integrity. If a statement set through a PASSTHRU session is successful, it is recorded as an update regardless of the type of statement. This includes SELECT statements. If a statement is not successful, it is not recorded. Likewise, if a PASSTHRU session is empty, a statement following the empty PASSTHRU session will not be blocked.

Considerations with the auto-commit option in the DB2 CLP:

By default, the DB2 CLP will automatically commit each SQL statement executed. If you elect to turn the auto-commit command option OFF, make certain you explicitly issue COMMIT and ROLLBACK statements at the end of each transaction.

Recommendation: Set the auto-commit command option ON for distributed units of work whenever applicable. If you have set this command option OFF, you can turn it on by issuing this command:

```
UPDATE COMMAND OPTIONS USING c ON
```

INSERT, UPDATE, and DELETE privileges:

The privileges required to issue INSERT, UPDATE, and DELETE statements on nicknames are similar to the privileges required to issue these statements on tables:

- You can grant or revoke SELECT, INSERT, UPDATE, and DELETE privileges on a nickname.
- You must hold adequate privileges on the data source to perform select, insert, update, or delete operations on the underlying object.

When a query is submitted to the federated database, the authorization privileges on the nickname in the query are checked. The authorization requirements of the data source object referenced by the nickname are only applied when the query is actually processed. If you do not have SELECT privilege on the nickname, then you can not select from the object the nickname refers to. Likewise, just because you have a privilege, such as UPDATE, on the nickname does not mean you will automatically be authorized to update the object that the nickname represents. Passing the privileges checking at the federated server does not imply you will pass the privilege checking at the remote data source. Through user mappings, a federated server user ID is mapped to the data source user ID. The privilege restrictions will be enforced at the data source.

Restrictions:

The Sybase DBLIB wrapper is read-only, you can not perform INSERT, UPDATE, or DELETE operations on a nickname that uses the DBLIB wrapper.

The ODBC wrapper is read-only, you can not perform INSERT, UPDATE, or DELETE operations on a nickname that uses the ODBC wrapper.

The wrappers provided with DB2 Life Sciences Data Connect are read-only, you can not perform INSERT, UPDATE, or DELETE operations on a nickname that use these wrappers.

Update of nicknames has the following restrictions:

- A nicknamed object whose data source does not permit update cannot be updated.
- A federated view with UNION ALL statements for multiple nicknamed object is a read only view. It cannot be updated.

Referential integrity:

You can not define a constraint on a nickname. In the federated environment, DB2 does not compensate for referential integrity differences between data sources. DB2 does not interfere with referential integrity enforcement at the data sources. However, referential integrity constraints at a data source can affect nickname updates. For example, suppose an insert into a table at a data source violates a referential integrity constraint at that data source. DB2 maps the resulting error to a DB2 error. Referential integrity between data sources is the responsibility of the applications.

LOBs:

There are three types of LOBs: character large objects (CLOBs), double-byte character large objects (DBCLOBs), and binary large objects (BLOBs).

Using DB2 for UNIX® and Windows® Version 8, you can perform read operations against LOBs located in all the relational data sources. Additionally, you can perform write operations against LOBs located in Oracle (Version 7.3 or higher) data sources using the NET8 wrapper. Non-relational data sources do not support LOBs.

Application savepoint:

In order to protect statement level atomicity for insert, update, or delete against a nickname, enhancement is made in Federated System to guard against potential data inconsistency. Application savepoints at the data sources are incorporated in the global design. If a data source does not support application savepoints, Federated System is unable to ensure statement level atomicity at runtime in the event of an error. A new SQL error code, SQLCODE -20190, is returned to the users when Federated System detects potential exposure of data inconsistency on any insert, update, or delete operations against nicknames residing in this data source. To open up insert, update, delete against nicknames on such data source, user may turn off the blocking logic via Alter Server command to set server option 'iud_app_svpt_enforce' to 'N'.

Some data sources, such as Informix, do not support application save points. If you are accessing data sources that do not support cursor application save

points, you need to change your server definitions. Add the IUD_APP_SVPT_ENFORCE server option and set the option to 'N'. This will enable you to update the data source using nicknames. Use the ALTER SERVER statement to add this option to the server definition.

Triggers:

You cannot define a trigger on a nickname.

Related concepts:

- "X/Open distributed transaction processing model" in the *Administration Guide: Planning*
- "Federated LOB support" on page 265

Selecting data in a federated system

Use the SELECT statement to select data from data sources.

Restrictions:

To select data using a nickname, the privileges held by the authorization ID of the statement must include SELECT privilege on the nickname (for the federated database to accept the request), and SELECT privilege on the underlying table object (for the data source to accept the request).

Procedure:

Some of the types of distributed requests used with a federated system are requests that query:

- A single remote data source.
- A local data source and a remote data source.
- Multiple remote data sources.

The federated database is a local data source. Tables and view in the federated database are local objects. You do not create nicknames for these objects, you use the actual object name in your SELECT statement. Remote data sources include: another DB2 for UNIX and Windows database instance on the federated server, another DB2 for UNIX and Windows database instance on another server, and data sources other than DB2 for UNIX and Windows. Th following are examples of SELECT statements. Refer to the *DB2 SQL Reference* for the full syntax and functionality of Select statement.

Suppose that a federated server is configured to access a DB2 for OS/390 data source, a DB2 for iSeries data source, and an Oracle data source. Stored in each data source is a table that contains sales information.

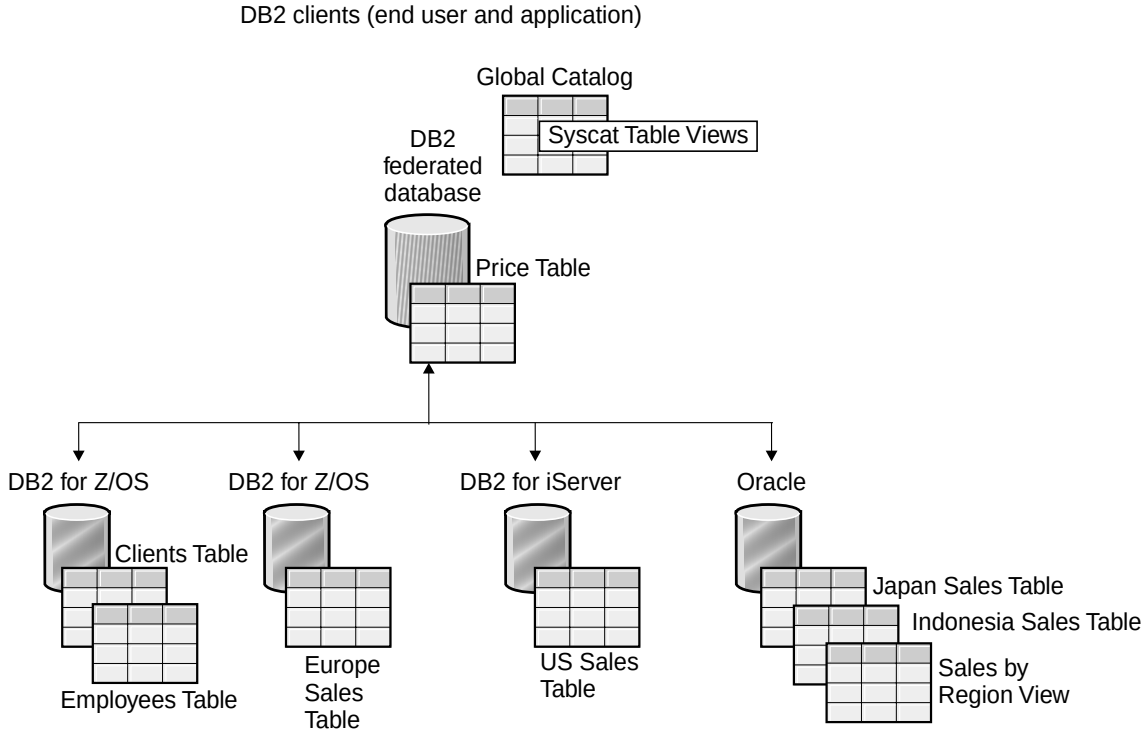

Figure 4. Sample federated system with DB2 and Oracle data sources

These tables include columns that record the customer number (CUST_NO), the quantity ordered (QUANTITY), and the product number ordered (PROD_NO). Additionally, you have a local table in the federated database which contains price information. This table includes columns that record the product number (PROD_NO) and the current price (PRICE).

The nicknames for the remote data source objects are stored in the SYSCAT tables.

Data source information

Data source object name	Type of object	Location
PRICES	Local table	DB2 federated database
EUROPE_SALES	Remote table	DB2 for z/OS and OS/390 database
US_SALES	Remote table	DB2 for iSeries database
JAPAN_SALES	Remote table	Oracle database
SALES_BY_REGION	Remote view	Oracle database

SYSCAT Tables

TABNAME	TYPE
PRICES	T
FED_PRICES	N
Z_EU_SALES	N
iS_US_SALES	N
ORA_JAPANSALES	N
ORA_REGIONSALES	N
.....	

Figure 5. Tables and nicknames for sample queries

The following examples show how you can query these data sources.

Querying a single data source:

Z_EU_SALES contains the products ordered by your European customers. It also includes the quantity ordered at each sale. The query uses a SELECT statement with an ORDER BY clause to list the sales in Europe sorted by customer number:

```
SELECT CUST_NO, PROD_NO, QUANTITY
FROM Z_EU_SALES
ORDER BY CUST_NO
```

Joining a local data source and a remote data source:

PRICES is a table that resides in the federated database. It contains the price list for the products you sell. You want to select the prices from this local table that correspond to the products listed in Z_EU_SALES. You also want to sort the result set by the customer number.

```
SELECT sales.CUST_NO, sales.PROD_NO, sales.QUANTITY,
FROM Z_EU_SALES sales, PRICES
WHERE sales.PROD_NO=PRICE.PROD_NO
ORDER BY sales.CUST_NO
```

Querying multiple remote data sources:

Suppose that you want to gather all the sales information from each region, and order the result set by product number.

```
WITH GLOBAL_SALES (Customer, Product, Quantity) AS
(SELECT CUST_NO, PROD_NO, QUANTITY FROM Z_EU_SALES
UNION ALL
SELECT CUST.NO,PROD.NO, QUANTITY FROM iS_US_SALES
UNION ALL
```

```
SELECT CUST.NO,PROD.NO, QUANTITY FROM ORA_JAPANSALES)
SELECT Customer, Product, Quantity
FROM GLOBAL_SALES
ORDER BY Product
```

Suppose that you have a view at the Oracle data source which lists the sales for Japan and Indonesia. The nickname for this view is ORA_SALESREGION. You want to combine this information with the sales from the United States and display the product prices next to each sale.

```
SELECT us_jpn_ind.CUST_NO, us_jpn_ind.PROD_NO,
 us_jpn_ind.QUANTITY, us_jpn_ind.QUANTITY*PRICES.PRICE
AS SALEPRICE FROM
(SELECT CUST_NO, PROD_NO, QUANTITY
FROM ORA_SALESREGION
UNION ALL
SELECT CUST_NO, PROD_NO, QUANTITY
FROM iS_US_SALES us ) us_jpn_ind,PRICES
WHERE us_jpn_ind.PROD_NO = PRICES.PROD_NO
ORDER BY SALEPRICE DESC
```

Related reference:

- "SELECT statement" in the *SQL Reference, Volume 2*

Modifying data in a federated system

With a federated system, you can perform INSERT, UPDATE, and DELETE operations on nicknamed objects. The following sections include examples for performing these operations.

Inserting data into data source objects

Use the INSERT statement to insert data into data sources.

There are two types of data source objects: local and remote. In a federated system, local data source objects are object that reside in the federated database. You do not create nicknames for these objects; you use the actual object name in your INSERT statement. Remote data source objects are any objects that do not reside in the federated database, including objects that reside on the federated server.

Prerequisites:

To insert using a nickname, the privileges held by the authorization ID of the statement must include INSERT privilege on the nickname (for the federated database to accept the request), and INSERT privilege on the underlying table object (for the data source to accept the request).

Restrictions:

INSERT is not available through the ODBC wrapper, the DBLIB wrapper, or the wrappers that are provided from DB2 Life Sciences Data Connect.

Procedure:

Suppose you have an Informix table that has been created as follows:

```
CREATE TABLE infx_table (c1 INTEGER, c2 VARCHAR(20))
```

You can use the following SQL to configure the federated server to access this table:

```
CREATE WRAPPER informix

CREATE SERVER infx_server TYPE informix
VERSION 9.3 WRAPPER informix
OPTIONS(ADD NODE 'inf93', ADD DBNAME 'inf_db',
ADD IUD_APP_SVPT_ENFORCE 'N')

CREATE USER MAPPING FOR USER SERVER infx_server
OPTIONS(ADD REMOTE_AUTHID 'infx_authid', ADD REMOTE_PASSWORD 'infx_pswd')

CREATE NICKNAME infx_table_nn FOR infx_server."infx_authid"."infx_table
```

You can issue insert, update, and delete statements using the *infx_table_nn* nickname. For example:

```
INSERT INTO infx_table_nn VALUES(1,'Walter')
```

Related tasks:

- "Selecting data in a federated system" on page 188
- "Updating data in data source objects" on page 192
- "Deleting data from data source objects" on page 193

Related reference:

- "INSERT statement" in the *SQL Reference, Volume 2*

Updating data in data source objects

Use the UPDATE statement to change data in data sources.

There are two types of data source objects: local and remote. In a federated system, local data source objects are object that reside in the federated database. You do not create nicknames for these objects; you use the actual object name in your UPDATE statement. Remote data source objects are any objects that do not reside in the federated database, including objects that reside on the federated server.

Prerequisites:

To update from a nickname, the privileges held by the authorization ID of the statement must include UPDATE privilege on the nickname (for the federated database to accept the request), and UPDATE privilege on the underlying table object (for the data source to accept the request).

Restrictions:

UPDATE is not available through the ODBC wrapper, the DBLIB wrapper, or the wrappers that are provided from DB2 Life Sciences Data Connect.

Procedure:

Suppose you have an Informix table that has been created as follows:

```
CREATE TABLE infx_table (c1 INTEGER, c2 VARCHAR(20))
```

You can use the following SQL to configure the federated server to access this table:

```
CREATE WRAPPER informix

CREATE SERVER infx_server TYPE informix
VERSION 9.3 WRAPPER informix
OPTIONS(ADD NODE 'inf93', ADD DBNAME 'inf_db',
ADD IUD_APP_SVPT_ENFORCE 'N')

CREATE USER MAPPING FOR USER SERVER infx_server
OPTIONS(ADD REMOTE_AUTHID 'infx_authid', ADD REMOTE_PASSWORD 'infx_pswd')

CREATE NICKNAME infx_table_nn FOR infx_server."infx_authid"."infx_table
```

You can issue insert, update, and delete statements using the *infx_table_nn* nickname. For example:

```
UPDATE infx_table_nn SET c2='Bill' WHERE c1=2
```

Related tasks:

- "Selecting data in a federated system" on page 188
- "Inserting data into data source objects" on page 191
- "Deleting data from data source objects" on page 193

Related reference:

- "UPDATE statement" in the *SQL Reference, Volume 2*

Deleting data from data source objects

Use the DELETE statement to delete data from data sources.

There are two types of data source objects: local and remote. In a federated system, local data source objects are object that reside in the federated database. You do not create nicknames for these objects; you use the actual

object name in your UPDATE statement. Remote data source objects are any objects that do not reside in the federated database, including objects that reside on the federated server.

Prerequisites:

To delete from a nickname, the privileges held by the authorization ID of the statement must include DELETE privilege on the nickname (for the federated database to accept the request), and DELETE privilege on the underlying table object (for the data source to accept the request).

Restrictions:

DELETE is not available through the ODBC wrapper, the DBLIB wrapper, or the wrappers that are provided from DB2 Life Sciences Data Connect.

Procedure:

Suppose you have an Informix table that has been created as follows:

```
CREATE TABLE infx_table (c1 INTEGER, c2 VARCHAR(20))
```

You can use the following SQL to configure the federated server to access this table:

```
CREATE WRAPPER informix

CREATE SERVER infx_server TYPE informix
VERSION 9.3 WRAPPER informix
OPTIONS(ADD NODE 'inf93', ADD DBNAME 'inf_db',
ADD IUD_APP_SVPT_ENFORCE 'N')

CREATE USER MAPPING FOR USER SERVER infx_server
OPTIONS(ADD REMOTE_AUTHID 'infx_authid', ADD REMOTE_PASSWORD 'infx_pswd')

CREATE NICKNAME infx_table_nn FOR infx_server."infx_authid"."infx_table
```

You can issue insert, update, and delete statements using the *infx_table_nn* nickname. For example:

```
DELETE FROM infx_table_nn WHERE c1=3
```

Related tasks:

- “Selecting data in a federated system” on page 188
- “Inserting data into data source objects” on page 191
- “Updating data in data source objects” on page 192

Related reference:

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

Chapter 13. Modifying the federated system

Periodically, you will need to make adjustments to your federated system. For example, you might need to add a column option to a nickname to improve performance. You might need to make the federated database aware of a new index that has been added to a data source object.

This chapter describes how to:

- Modify wrappers
- Modify nicknames
- Modify server definitions
- Create and modify user-defined data type mappings
- Create index specifications for data source objects
- Create and modify user-defined function mappings
- Create and modify remote tables using transparent DDL

Modifying wrappers

Wrappers are mechanisms by which the federated server interacts with data sources. Typically, you create one wrapper for each type of data source you want to access.

Once you create the wrapper, you can change or delete it.

- If you did not explicitly set the DB2_FENCED wrapper option to 'N', you can alter the wrapper to add this option.
- Suppose that you create the wrong wrapper. You can drop the wrapper and create a new one.
- When you no longer need access to a data source, you can drop the wrapper.

Related tasks:

- "Altering a wrapper" on page 196
- "Dropping a wrapper" on page 196

Related reference:

- Appendix B, "Wrapper options for federated systems" on page 285

Modifying wrappers-details

Altering a wrapper

If you did not explicitly set the DB2_FENCED wrapper option to 'N', you can alter the wrapper to add this option. If you have scripts or applications that you use for DDL statements, consider adding this option. Even though the current default setting for DB2_FENCED is 'N', it is possible that IBM will change the default setting in the future. When the default changes, any wrappers created without this option will adhere to the new default. If you explicitly set the DB2_FENCED wrapper to 'N', you can ensure that the behavior of the wrapper will not change when you run the scripts or applications.

Prerequisites:

You must have SYSADM or DBADM authority to issue the ALTER WRAPPER statement.

Restrictions:

The only value supported in DB2 for UNIX and Windows Version 8 for the DB2_FENCED wrapper option is 'N'.

Procedure:

To change a wrapper, use the ALTER WRAPPER statement. Suppose that you have an Informix wrapper and want to apply the DB2_FENCED option to the wrapper. The statement you use is:

```
ALTER WRAPPER INFORMIX OPTIONS (SET DB2_FENCED N')
```

Related tasks:

- "Wrappers : Federated Systems help" in the *Help: Federated Systems*

Related reference:

- "ALTER WRAPPER statement" in the *SQL Reference, Volume 2*
- Appendix B, "Wrapper options for federated systems" on page 285

Dropping a wrapper

There are several reasons why you might want to drop a wrapper.

Suppose that you create the wrong wrapper. You can drop the wrapper and create a new one. For example, sometimes a data source has 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 you are using on your federated server. Suppose that you want to access two Oracle tables and one Oracle view. 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. The implications of dropping a wrapper are discussed in that topic.

Another reason to drop a wrapper is that you no longer need access to the data source that the wrapper is associated with. For example, suppose that your organization has a requirement to access client information in both Sybase and Microsoft SQL server databases. You create one wrapper for the Informix data source and one wrapper for the Microsoft SQL Server data source. Later your organization decides to migrate all of the information from Microsoft SQL Server to Informix. You no longer need the Microsoft SQL Server wrapper and can drop it.

Caution: There are implications when you drop a wrapper, other objects are impacted:

- All server definitions, user-defined functions mappings, and user-defined data type mappings that are dependent on the wrapper are 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 dropped.
- Any federated views dependent on these nicknames are marked inoperative.
- All applications dependent on the dropped objects and inoperative views are invalidated.

Prerequisites:

To issue the DROP WRAPPER statement, you must have one of the following authorizations:

- SYSADM or DBADM authority.
- DROPIN privilege on the schema for the wrapper.
- Definer of the wrapper as recorded in the DEFINER column of the SYSCAT.WRAPPERS catalog view.

Procedure:

To drop a wrapper, use the DROP statement. For example, to drop the Microsoft SQL Server *MSSQLODBC3* wrapper, the statement you use is:

```
DROP WRAPPER MSSQLODBC3
```

Related concepts:

- "Create the wrapper" on page 88

Related reference:

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

Modifying nicknames

A *nickname* is an identifier used to reference a data source object. You create a nickname for each table, view, or other object at the data source that you want to access.

Once you create the nickname, there are several reasons why you might want to alter or drop it. For example:

- You can alter the nickname to change the local column names.
- You can alter the nickname to set a data type mapping that applies to only one specific data source object.
- You can alter the nickname to add a column option that might improve performance.
- If the underlying data source object structure or content has changed dramatically, you might decide to drop the nickname and re-create it so that the metadata about the object is updated in the global catalog.
- You must drop a nickname and re-create it if you want to change the name of a nickname.
- You can drop the nickname when you no longer need access to the underlying data source object.

Related tasks:

- "Altering a nickname" on page 199
- "Dropping a nickname" on page 202

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*
- "ALTER NICKNAME statement" in the *SQL Reference, Volume 2*
- Appendix E, "Column options for federated systems" on page 299

Modifying nicknames-details

Altering a nickname

You change a nickname to modify the federated database's representation of a data source object. There are several reason why you might want to alter a nickname.

- To alter the local column names for a nickname
- To alter the data type mapping for a nickname column
- To alter the nickname column options

Use the ALTER NICKNAME statement to modify a nickname.

Prerequisites:

The privileges held by the authorization ID of the statement must include at least one of the following:

- SYSADM or DBADM authority
- ALTER privilege on the nickname specified in the statement
- CONTROL privilege on the nickname specified in the statement
- ALTERIN privilege on the schema, if the schema name of the nickname exists
- Definer of the nickname as recorded in the DEFINER column of the catalog view for the nickname

Restrictions:

- The ALTER NICKNAME statement cannot be used to register a new data source index in the federated database. Use the CREATE INDEX statement with the SPECIFICATION ONLY clause to create a index specification.
- The ALTER NICKNAME statement cannot be used to change the name of the nickname. To change the name of the nickname, you must drop the nickname and create the nickname again using the new name.
- If a view has been created on a nickname, the ALTER NICKNAME statement cannot be used to change the local names or data types for the columns in the table or view that the nickname references.
- The federated_column_options clause must be specified last if you also need to specify the LOCAL NAME parameter, the LOCAL TYPE parameter, or both in the ALTER NICKNAME statement.
- The ALTER TABLE statement cannot be used to modify a nickname.

Altering nickname column names

When you create a nickname, the column names associated with the underlying data source object are stored locally, in the federated database.

Suppose that you have a nickname *Z_EMPLOYEES* for a DB2 for z/OS table that includes a column with the name of *EMPNO*. To make it clear what data is stored in that column, you can change the local column name. You can alter the nickname so that the local column name that users work with is *Employee_Number* instead of *EMPNO*.

```
ALTER NICKNAME Z_EMPLOYEES ALTER COLUMN EMPNO LOCAL NAME 'Employee_Number'
```

Altering nickname column data type mappings

Suppose that you have the nickname ORASALES for an Oracle table called SALES. The table contains a column that is the Oracle DATE data type. The default type mapping for the Oracle DATE data type is to the DB2 TIMESTAMP data type. However you only want to display the date value when you retrieve data from this column. You can alter the nickname for the SALES table to change the mapping to the DB2 DATE data type.

```
ALTER NICKNAME ORASALES ALTER COLUMN ORDER_DATE LOCAL TYPE DATE
```

Using the ALTER NICKNAME statement to change a data type mapping, only sets the mapping for the specific data source object when it is accessed through that nickname. You can have another nickname for the same object that uses the default data type mapping.

Note: When the local specification of a column's data type is changed, the database manager invalidates any statistics (HIGH2KEY, LOW2KEY, and so on) gathered for that column.

Altering nickname column options

You specify column information in the CREATE NICKNAME or ALTER NICKNAME statements using parameters called *column options*. You can specify any of these values in either upper- or lowercase. 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 operations. This assists the SQL Compiler in reaching global optimization.

You can ADD, SET, or DROP column options in the ALTER NICKNAME statements

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. DB2 treats the object that a nickname references as if it is a table. As a result, you can set column options for any data source object that you create a nickname for. The ALTER NICKNAME statement can be used to add or change column options for nicknames. There are two column options: NUMERIC_STRING and VARCHAR_NO_TRAILING_BLANKS.

NUMERIC_STRING: This column option applies to character type columns (CHAR and VARCHAR). 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 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.

Suppose that you have the nickname ORA_INDSALES for an Oracle table called INDONESIA_SALES. The table contains the column POSTAL_CODE with the data type of VARCHAR. Originally the column contained only numeric characters, and the NUMERIC_STRING column option was set to 'Y'. However, the column now contains a mixture of numeric and non-numeric character. To change the NUMERIC_STRING column option to 'N':

```
ALTER NICKNAME ORA_INDSALES ALTER COLUMN POSTAL_CODE
OPTION (SET NUMERIC_STRING 'N')
```

VARCHAR_NO_TRAILING_BLANKS: Unlike the server option with the same name, the VARCHAR_NO_TRAILING_BLANKS column option can be used to identity specific columns that contain no trailing blanks. The SQL Compiler pushdown analysis step will then take this information into account when checking all operations performed on columns which have this setting.

Suppose that you have the nickname ORA_INDSALES for an Oracle table called INDONESIA_SALES. The table contains the column NAME with the data type of VARCHAR. The NAME column does not have trailing blanks. To add the VARCHAR_NO_TRAILING_BLANKS option to the nickname:

```
ALTER NICKNAME ORA_INDSALES ALTER COLUMN NAME
OPTION (ADD VARCHAR_NO_TRAILING_BLANKS 'Y')
```

Related concepts:

- "Create nicknames for each data source object" on page 96
- "Pushdown analysis" on page 233

Related tasks:

- "Dropping a nickname" on page 202
- "Global optimization" on page 246
- "Nicknames : Federated Systems help" in the *Help: Federated Systems*
- "Filtering tables and views for creating nicknames : Federated Systems help" in the *Help: Federated Systems*

- "Changing options for a single nickname: Federated Systems help" in the *Help: Federated Systems*
- "Filtering tables for creating nicknames: Federated Systems help" in the *Help: Federated Systems*
- "Changing options for all nicknames : Federated Systems help" in the *Help: Federated Systems*

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*
- "ALTER NICKNAME statement" in the *SQL Reference, Volume 2*
- Appendix E, "Column options for federated systems" on page 299

Dropping a nickname

Dropping a nickname deletes the nickname from the global catalog. Additionally, any objects that are directly or indirectly dependent on that nickname are either deleted or made inoperative. For example, federated views are defined using nicknames. If a nickname is used to define a view, and the nickname is later dropped, the view is made inoperative. Any plans that are dependent upon that view become invalid.

When a nickname is dropped, the data source object that the nickname references is not affected.

Use the DROP statement to delete a nickname.

Prerequisites:

The nickname must be listed in the catalog.

The privileges that must be held by the authorization ID of the DROP statement when dropping nicknames must include one of the following:

- SYSADM or DBADM authority
- DROPIN privilege on the schema for the nickname
- Definer of the nickname as recorded in the DEFINER column of the catalog view for the nickname
- CONTROL privilege on the nickname

Procedure:

There are several reasons why you might need to drop a nickname:

- If the underlying data source object structure or content has changed dramatically, you might decide to drop the nickname. You can then re-create the nickname so that the metadata about the object is updated in the global catalog.
- If you want to change the name of a nickname, you must drop a nickname and re-create the nickname using the new name.
- If you no longer need access to the underlying data source object, you can drop the nickname.

To drop a nickname, the syntax is:

```
DROP NICKNAME nickname
```

where *nickname* identifies the nickname to be dropped.

The nickname is deleted from the federated database. Additionally:

- All information about the columns and indexes associated with the nickname is deleted from the global catalog.
- Any index specifications that are dependent on the nickname are dropped.
- Any views dependent on the nickname are marked inoperative.
- Any packages depending on the dropped index specifications or inoperative views are invalidated.
-

Related concepts:

- "Create nicknames for each data source object" on page 96

Related tasks:

- "Altering a nickname" on page 199

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*

Modifying server definitions

A server definition is a set of metadata that the federated server uses to connect to a data source.

Use the ALTER SERVER statement to modify a server definition. Some of the information within a server definition is stored as server options. When you modify a server definition, 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 are specified as parameters in the CREATE SERVER and ALTER SERVER statements.

Use the DROP statement to delete a server definition.

Related tasks:

- "Altering server definitions" on page 204
- "Dropping a server definition" on page 207

Related reference:

- Appendix C, "Server options for federated systems" on page 287

Modifying server definitions-details

Altering server definitions

You change an existing server definition in the federated database catalog to:

- Modify the definition of a specific data source.
- Modify the definition of a specific type or version of data source.
- Make changes in the configuration through server options.

You can alter a server definition using the DB2 Control Center or through the DB2 Command Line Processor.

To alter a server definition using the DB2 Control Center, open the Alter Server Options window.

To alter a server definition from the DB2 Command Line Prompt, use the ALTER SERVER statement.

Note: In the ALTER SERVER statement, the word SERVER and the parameter names that start with server- refer only to data sources in a federated system. They do not refer to the federated server in such a system, or to DRDA application servers.

Prerequisites:

The authorization ID of the statement must include either SYSADM or DBADM authority on the federated database.

Restrictions:

You cannot use the ALTER SERVER statement to:

- Modify the DBNAME or NODE server options.
- Change the wrapper that is associated with a server mapping. For example to change from DBLIB to CTLIB.

Modifying the definition of a specific data source

Suppose that you have a server definition for a Microsoft SQL Server Version 6.5 data source. The name you assigned the server in the CREATE SERVER statement is SQLSVR_ASIA. If the Microsoft SQL Server server is upgraded to Version 7.0, the statement to alter the server definition is:

```
ALTER SERVER SQLSVR_ASIA VERSION 7
```

Suppose that you have a server definition for a Sybase data source that uses the DBLIB wrapper. The name you assigned the server in the CREATE SERVER statement is SYBSERVER. To take advantage of the more powerful capabilities of the CTLIB wrapper, you decide to upgrade the server. To do this, you must first create the wrapper, using the CREATE WRAPPER statement. Then you can change the wrapper used in the server definition. The statements to create the wrapper and alter the server definition are:

```
CREATE WRAPPER CTLIB OPTIONS
ALTER SERVER SYBSERVER WRAPPER CTLIB
```

Modifying the definition of a specific type or version of data source

Suppose that you have five Informix servers that you created server definitions for. If they have all been upgraded from Informix Version 7 to Version 9, you can modify all five definitions at the same time.

```
ALTER SERVER TYPE informix VERSION 9
```

Suppose that you have a server definition for an Oracle Version 7 data source that uses the SQLNET wrapper. The name assigned to the server in the CREATE SERVER statement is ORASERVER. This server has been upgraded to Oracle Version 9 and will use the NET8 wrapper. To change the version and wrapper in the server definition, the statement is:

```
ALTER SERVER ORASERVER VERSION 9 WRAPPER NET8
```

Changing the server configuration through server options

Server options are set to values that persist over successive connections to the data source. These values are stored in the global catalog. There are general server options and data type–specific server options. Examples of server options include:

- If you defined an Informix server using the server name of INFMX01 and you now want to change the CPU_RATIO option to 5.0, the statement to alter the server definition is:

  ```
  ALTER SERVER INFMX01 OPTIONS (SET CPU_RATIO '5.0')
  ```

- If you defined an Oracle server using the server name of ORCL99 and you now want to add the FOLD_ID and FOLD_PW options, the statement to alter the server definition is:

  ```
  ALTER SERVER ORCL99 OPTIONS (ADD FOLD_ID'U', FOLD_PW 'U')
  ```

- If you want to set the timeout value to the number of seconds the DBLIB wrapper should wait for a response from the Sybase server, use the TIMEOUT server option. The statement to alter the server definition is:
  ```
  ALTER SERVER SYBSERVER OPTIONS (ADD TIMEOUT '60')
  ```

Server option overrides: Suppose a server option is set to one value for a type of data source, for example the PLAN_HINTS server option is set to 'Y' for server type ORACLE. However, the PLAN_HINTS server option is set to 'N' for a specific data source named PURNELL. The setting for the specific data source overrides the setting for the server type. This configuration causes PLAN_HINTS to be enabled at all Oracle data sources except PURNELL.

Modifying the DBNAME and NODE server options: The name for the data source on the RDBMS is set in the NODE server option. Some data sources have multiple databases on each instance. For these data source, the name of the database which the federated server connects to is set in the DBNAME server option. These options are set when you initially create the server definition in the CREATE SERVER statement.

To modify the DBNAME or NODE server options, you must drop the server definition and re-create it. Alternatively you can create a new one server definition, using a different name, while keeping the original server definition active to ensure the new definition has the same values as the original definition. You would then drop the original server definition.

Changing server options temporarily: To set a server option value temporarily, use the SET SERVER OPTION statement. This statement overrides the value for the duration of a single connection to the federated database. The overriding value does not get stored in the global catalog.

Related concepts:

- "Server characteristics affecting pushdown opportunities" on page 235
- "Server characteristics affecting global optimization" on page 247

Related tasks:

- "Modifying server definitions" on page 203
- "Dropping a server definition" on page 207

- "Servers : Federated Systems help" in the *Help: Federated Systems*
- "Selecting server options: Federated Systems help" in the *Help: Federated Systems*
- "Adding a new server option: Federated Systems help" in the *Help: Federated Systems*
- "Viewing server options: Federated Systems help" in the *Help: Federated Systems*

- "Adding a server option: Federated Systems help" in the *Help: Federated Systems*
- "Altering the values for server options: Federated Systems help" in the *Help: Federated Systems*
- "Dropping server options: Federated Systems help" in the *Help: Federated Systems*

Related reference:

- "ALTER SERVER statement" in the *SQL Reference, Volume 2*
- "SET SERVER OPTION statement" in the *SQL Reference, Volume 2*
- Appendix C, "Server options for federated systems" on page 287

Dropping a server definition

Dropping a server definition deletes the definition from the global catalog. Additionally, any objects that are directly or indirectly dependent on that server definition are either deleted or made inoperative.

When a server definition is dropped, the data source object that the server definition references is not affected.

Use the DROP statement to delete a server definition.

Prerequisites:

You must have SYSADM or DBADM authority to drop a server definition.

Procedure:

When you no longer need to access a data source server, drop the server definition from the federated database. You can drop a server definition using the DB2 Control Center or using DROP statement from the DB2 command line processor.

To drop a server definition, the syntax is:

```
DROP SERVER servername
```

where *servername* identifies the server definition to be dropped.

If you defined an Informix server that uses the server name of INFMX01, the statement to drop the server definition is:

```
DROP SERVER INFMX01
```

The server definition is deleted from the federated database. Additionally:

- All nicknames for data source objects (such as tables and views) residing at the data source are dropped.
- Any index specifications dependent on these nicknames are dropped.
- Any user-defined function mappings, user-defined type mappings, and user mappings that are dependent on the dropped server definition are dropped.
- All packages dependent on the dropped server definition, function mappings, nicknames, and index specifications are invalidated.

Related concepts:

- "Supply the server definition" on page 91

Related tasks:

- "Modifying server definitions" on page 203
- "Altering server definitions" on page 204

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*

Modifying default data type mappings

When you create a nickname for a data source table, 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.

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. The other type of mapping is a *reverse type mapping*, which are used with transparent DDL. In a reverse type mapping, the mapping is from a local type to a comparable remote type.

DB2 uses data type mappings to determine what DB2-supported data type should be defined for the column of 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 SYSSTAT.TYPEMAPPING 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 the column names and the data types of those columns. Only nicknames created after a mapping is changed, reflect the new type mapping. Nicknames created before the mapping is changed relfect the default data type mapping. To update existing nicknames, you will have to either alter each nickname to reflect the new mapping, or drop and re-create the nickname.

Forward type mapping:

A forward type mapping is the mapping from a local DB2 type to a a remote (data source) data type. Forward type mappings are used when a query you submit to the federated database includes a nicknames for that data source.

The syntax for the CREATE TYPE MAPPING statement has been updated in DB2 for UNIX and Windows, Version 8. There are two acceptable formats for creating a forward type mapping.

```
CREATE TYPE MAPPING type_mapping_name FROM LOCAL TYPE local_data_type
TO remote_server REMOTE TYPE data_source_data_type
```

```
CREATE TYPE MAPPING type_mapping_name TO remote_server
REMOTE TYPE data_source_data_type FROM LOCAL TYPE local_data_type
```

Both a TO and a FROM keyword must be present in the CREATE TYPE MAPPING statement. If the type has a short and long form (for example, CHAR and CHARACTER), the short form should be specified.

Reverse type mapping:

A reverse type mapping is only used with transparent DDL. As part of transparent DDL, you want to establish as mapping from a data source data type to a DB2 type. Reverse type mappings are only used when you create a remote table and a nickname for that remote table using transparent DDL.

The syntax for the CREATE TYPE MAPPING statement has been updated in DB2 for UNIX and Windows, Version 8. There are two acceptable formats for creating a forward type mapping.

```
CREATE TYPE MAPPING type_mapping_name FROM remote_server
REMOTE TYPE data_source_data_type TO LOCAL TYPE local_data_type
```

```
CREATE TYPE MAPPING type_mapping_name TO LOCAL TYPE local_data_type
FROM remote_server REMOTE TYPE data_source_data_type
```

Both a TO and a FROM keyword must be present in the CREATE TYPE MAPPING statement. If the type has a short and long form (for example, CHAR and CHARACTER), the short form should be specified

Unsupported data types:

Federated servers to not support mappings for all DB2 data types. The *local_data_type* cannot be:

- LONG VARCHAR
- LONG VARGRAPHIC
- DATALINK
- a user-defined data type

The *data_source_data-type* cannot be a user-defined type.

However, there are alternatives you can use:

- User-defined types. You cannot create a user-defined type mapping for these data types. However, you create a nickname for view at the data source that is identical to the table that contains the user-defined data types. The view must 'cast' the user-defined type column to the built-in, or system, type. The drawback with this alternative, is that views have no statistics or indexes.
- LONG VARCHAR. A nickname can be created for a remote table that contains LONG VARCHAR columns. However, the results will be mapped to a local DB2 data type that is not LONG VARCHAR.

Related concepts:

- "Define alternative data type mappings to the federated database" on page 101
- "Data type mappings" on page 18

Related tasks:

- "Change a type mapping for all data source objects located on a specific server" on page 211
- "Change a type mapping for a specific data source object" on page 212

- "Change a type mapping for a specific data source type" on page 214
- "Change a type mapping for a specific data source type and version" on page 215

Related reference:

- Appendix H, "Default forward data type mappings" on page 307
- Appendix I, "Default reverse data type mappings" on page 323

Modifying default data type mappings-details

Change a type mapping for all data source objects located on a specific server

You can specific a type mapping for all objects located on a specific server.

Prerequisites:

The privileges held by the authorization ID of the statement must have SYSADM or DBADM authority.

Restrictions:

The *local_data_type* cannot be LONG VARCHAR, LONG VARGRAPHIC, DATALINK, or a user-defined data type. The *data_source_data-type* cannot be a user-defined type.

Procedure:

Suppose you want to access three Oracle tables located on the same server. You have defined this server to the federate database as ORA2SERVER in a server definition. Each table contains a column with a data type of DATE, which records time stamp information. The Oracle DATE data type is mapped by default to the local DB2 data type TIMESTAMP. However, when you query these columns you only want the result set to display time information. You can create a type mapping which overrides the default mapping for any object located on this Oracle server.

If you create nicknames for the three tables without changing the default type mapping, TIMESTAMP would be defined locally for these columns. Federated queries which included these columns would display time stamps in the result set.

Instead, create a data type mapping for ORA2SERVER before you create the nicknames for these tables. The DB2 TIME data type, instead of the DB2 TIMESTAMP data type, is defined locally for the Oracle columns. If new

tables which contain DATE columns are added to this Oracle server, nicknames created for those tables will also map the Oracle DATE data type to the DB2 TIME data type.

If you create the nicknames before you create the type mapping, you have use the ALTER NICKNAME statement to change the type mapping. You will have to modify each nickname separately to change the data type mapping.

To map the Oracle DATE data type to the DB2 TIME data type for the ORA2SERVER, the syntax is:

```
CREATE TYPE MAPPING ORA2_DATE FROM LOCAL TYPE TIME
TO SERVER ORA2SERVER REMOTE TYPE DATE
```

Related concepts:

- "Define alternative data type mappings to the federated database" on page 101

Related tasks:

- "Altering a nickname" on page 199
- "Change a type mapping for a specific data source object" on page 212
- "Change a type mapping for a specific data source type" on page 214
- "Change a type mapping for a specific data source type and version" on page 215

Related reference:

- "CREATE TYPE MAPPING statement" in the *SQL Reference, Volume 2*
- Appendix H, "Default forward data type mappings" on page 307

Change a type mapping for a specific data source object

To change the data type mapping for a column of a specific data source object, use the ALTER NICKNAME statement instead of the CREATE TYPE MAPPING statement.

Prerequisites:

The privileges held by the authorization ID of the statement must have SYSADM or DBADM authority.

Restrictions:

The *local_data_type* cannot be LONG VARCHAR, LONG VARGRAPHIC, DATALINK, or a user-defined data type. The *data_source_data-type* cannot be a user-defined type.

Procedure:

You can change the local type in a data type mapping for a specific table. For example, Oracle data type NUMBER(32,3) maps by default to the DB2 data type DOUBLE, a floating decimal data type. Suppose that in an Oracle table for employee information, a column BONUS was defined with a data type of NUMBER(32,3). Because of the mapping, a query which includes the BONUS column could return values that look like this:

```
5.0000000000000E+002
1.0000000000000E+003
```

where +002 signifies that the decimal point should be moved two places to the right, and +003 signifies that the decimal point should be moved three places to the right.

To have queries which include the BONUS column return values that look like dollar amounts, you could define a different mapping for this particular table. Change the default mapping for the Oracle NUMBER(32,3) type from the DB2 DOUBLE to the DB2 DECIMAL type. Use a precision and scale that reflect the format of actual bonuses. For example, if you knew that the dollar portion of the bonuses would not exceed six figures, you could map NUMBER(32,3) to DECIMAL(8,2). Under the constraint of this new mapping, a query including the BONUS column would return values like this:

```
 500.00
1000.00
```

The ALTER NICKNAME statement would be:

```
ALTER NICKNAME ORASALES ALTER COLUMN BONUS LOCAL TYPE DECIMAL(8,2)
```

Related concepts:

- “Define alternative data type mappings to the federated database” on page 101
- “Data type mappings” on page 18

Related tasks:

- “Altering a nickname” on page 199
- “Modifying default data type mappings” on page 208
- “Change a type mapping for all data source objects located on a specific server” on page 211
- “Change a type mapping for a specific data source type” on page 214
- “Change a type mapping for a specific data source type and version” on page 215

Related reference:

- "ALTER NICKNAME statement" in the *SQL Reference, Volume 2*
- "CREATE TYPE MAPPING statement" in the *SQL Reference, Volume 2*
- Appendix H, "Default forward data type mappings" on page 307

Change a type mapping for a specific data source type

You can specific a type mapping for all objects of a specific data source type. For example you can create a data type mapping for all Oracle data source objects, regardless of which Oracle server the object are located on.

Prerequisites:

The privileges held by the authorization ID of the statement must have SYSADM or DBADM authority.

Restrictions:

The *local_data_type* cannot be LONG VARCHAR, LONG VARGRAPHIC, DATALINK, or a user-defined data type. The *data_source_data-type* cannot be a user-defined type.

Procedure:

For example, Oracle data type NUMBER (23,3) maps by default to the DB2 data type DOUBLE, a floating decimal data type. Suppose that for Oracle tables and views you want any column that uses the Oracle data type NUMBER (23,2) to map to DB2 DECIMAL (8,2). The CREATE TYPE MAPPING statement would be:

```
CREATE TYPE MAPPING ORA_DEC FROM LOCAL TYPE SYSIBM.DECIMAL(8,2)
TO SERVER TYPE ORACLE REMOTE TYPE NUMBER (23,3)
```

Related concepts:

- "Define alternative data type mappings to the federated database" on page 101
- "Data type mappings" on page 18

Related tasks:

- "Change a type mapping for all data source objects located on a specific server" on page 211
- "Change a type mapping for a specific data source object" on page 212
- "Change a type mapping for a specific data source type and version" on page 215

Related reference:

- "CREATE TYPE MAPPING statement" in the *SQL Reference, Volume 2*
- Appendix H, "Default forward data type mappings" on page 307

Change a type mapping for a specific data source type and version

You can specific a type mapping for all objects of a specific data source type. For example you can create a data type mapping for all Oracle data source objects which use a Oracle Version 8 server.

Prerequisites:

The privileges held by the authorization ID of the statement must have SYSADM or DBADM authority.

Restrictions:

The *local_data_type* cannot be LONG VARCHAR, LONG VARGRAPHIC, DATALINK, or a user-defined data type. The *data_source_data-type* cannot be a user-defined type.

Procedure:

For example, Oracle data type NUMBER (23,3) maps by default to the DB2 data type DOUBLE, a floating decimal data type. Suppose that for Oracle tables and views you want any column that uses the Oracle data type NUMBER (23,2) to map to DB2 DECIMAL (8,2). The CREATE TYPE MAPPING statement would be:

```
CREATE TYPE MAPPING ORA_DEC FROM LOCAL TYPE SYSIBM.DECIMAL(8,2)
TO SERVER TYPE ORACLE VERSION 8 REMOTE TYPE NUMBER (23,3)
```

Related tasks:

- "Modifying default data type mappings" on page 208
- "Change a type mapping for all data source objects located on a specific server" on page 211
- "Change a type mapping for a specific data source object" on page 212
- "Change a type mapping for a specific data source type" on page 214

Related reference:

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

Creating index specifications for data source objects

When a nickname is created for a data source table, the federated server supplies the global catalog with information about any indexes that the data source table has. The optimizer uses this information to expedite the processing of distributed requests. This information is a set of metadata and called an *index specification*.

The federated server does not create an index specification if:

- A nickname is created for a table that has no index.
- A nickname is created for a data source object that does not contain indexes such as a view, Informix synonym, table-structured file, Documentum Docbase file, Excel spreadsheet, BLAST algorithm, or XML tagged file.
- The remote index is on a column of more than 255 bytes, or contains a total key length in excess of 1024 bytes.
- The remote index is on a LOB column.

In these circumstances the federated server does not store index specifications for the data source objects. However, you can supply the necessary index information to the global catalog using the CREATE INDEX statement.

Prerequisites:

The privileges held by the authorization ID of the statement must include at least one of the following:

- SYSADM or DBADM authority
- One of: CONTROL privilege on the object or INDEX privilege on the object. And one of: IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name of the index does not exist, or CREATEIN privilege on the schema, if the schema name of the index refers to an existing schema.

Restrictions:

There are some restrictions when creating an index specification on a nickname.

- If the bind option DYNAMICRULES BIND applies, the statement cannot be dynamically prepared. Also, you cannot use the INCLUDE, CLUSTER, PCTFREE, MINPCTUSED, DISALLOW REVERSE SCANS, and ALLOW REVERSE SCANS parameters in the CREATE INDEX statement.
- UNIQUE should be specified only if the data for the index key contains unique values for every row of the data source table. The uniqueness will not be checked.

- The sum of the stored lengths of the specified columns must not be greater than 1024.
- No LOB column, DATALINK column, or distinct type column based on a LOB or DATALINK may be used as part of an index. This restriction is enforced even if the length attribute of the column is small enough to fit within the 1024-byte limit.

CREATE INDEX syntax:

The CREATE INDEX statement can be embedded in an application program or issued through dynamic SQL statements from the Control Center or the command line.

When used with nicknames, the CREATE INDEX statement creates an index specification in the federated global catalog; it does not create an index on the data source table.

Use this syntax to create an index specification:

```
CREATE INDEX index_name ON nickname
(column_name) SPECIFICATION ONLY
```

```
CREATE UNIQUE INDEX index_name ON nickname
(column_name DESC) SPECIFICATION ONLY
```

For an index specification, *column_name* is the name by which the federated server references a column of a data source table.

Related concepts:

- "Index specifications" on page 22

Related tasks:

- "Creating index specifications on tables that acquire new indexes" on page 218
- "Creating index specifications on views" on page 219
- "Creating index specifications on Informix synonyms" on page 221
- "Nickname characteristics affecting global optimization" on page 249
- "Creating an index: Control Center help" in the *Help: Control Center*

Related reference:

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

Creating index specifications for data source objects-details

Creating index specifications on tables that acquire new indexes

There are several situations in which a table acquires a new index:

- You create a nickname for a table that does not have an index, but acquires an index later.
- You create a nickname for a table that has an index, but acquires another index later.

In these situations, you should create an index specification for the table so that the SQL Complier can use this information when processing queries that reference the table.

Prerequisites:

The privileges held by the authorization ID of the statement must include at least one of the following:

- SYSADM or DBADM authority
- One of: CONTROL privilege on the object or INDEX privilege on the object. And one of: IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name of the index does not exist, or CREATEIN privilege on the schema, if the schema name of the index refers to an existing schema.

Restrictions:

There are some restrictions when creating an index on a nickname.

- If the bind option DYNAMICRULES BIND applies, the statement cannot be dynamically prepared. Also, you cannot use the INCLUDE, CLUSTER, PCTFREE, MINPCTUSED, DISALLOW REVERSE SCANS, and ALLOW REVERSE SCANS parameters in the CREATE INDEX statement.
- UNIQUE should be specified only if the data for the index key contains unique values for every row of the data source table. The uniqueness will not be checked.
- The sum of the stored lengths of the specified columns must not be greater than 1024.
- No LOB column, DATALINK column, or distinct type column based on a LOB or DATALINK may be used as part of an index. This restriction is enforced even if the length attribute of the column is small enough to fit within the 1024-byte limit.

A table that has no index, later acquires an index:

Suppose that you create the nickname *employee* a data source table called CURRENT_EMP, which has no indexes. Sometime after this nickname is created, an index was defined on CURRENT_EMP using the WORKDEPT and JOB columns for the index key. Provide the new index information to the global catalog. The CREATE INDEX statement you create will reference the nickname for the table and contain information about the index of the data source table.

To create an index specification that describes this index, the syntax would be:

```
CREATE UNIQUE INDEX job_by_dept ON employee
(WORKDEPT, JOB) SPECIFICATION ONLY
```

where *job_by_dept* is the index name.

A table acquires a new index:

Suppose that you create the nickname *jp_sales* for a table called JAPAN_SALES. A new index is later added to the table in addition to the ones it had when the nickname was created. The new index uses the MARKUP column for the index key. Provide the new index information to the global catalog. The CREATE INDEX statement you create will reference the nickname for the table and contain information about the index of the data source table.

To create an index specification that describes this index, the syntax would be:

```
CREATE UNIQUE INDEX jp_markup ON jp_sales (MARKUP) SPECIFICATION ONLY
```

where *jp_markup* is the index name.

Related concepts:

- "Index specifications" on page 22

Related tasks:

- "Creating index specifications for data source objects" on page 216
- "Creating index specifications on views" on page 219
- "Creating index specifications on Informix synonyms" on page 221
- "Nickname characteristics affecting global optimization" on page 249

Related reference:

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

Creating index specifications on views

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. Create

an index specification for the view so that the SQL Complier can use this information when processing queries that reference the view.

Prerequisites:

The privileges held by the authorization ID of the statement must include at least one of the following:

- SYSADM or DBADM authority
- One of: CONTROL privilege on the object or INDEX privilege on the object. And one of: IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name of the index does not exist, or CREATEIN privilege on the schema, if the schema name of the index refers to an existing schema.

Restrictions:

There are some restrictions when creating an index on a nickname.

- If the bind option DYNAMICRULES BIND applies, the statement cannot be dynamically prepared. Also, you cannot use the INCLUDE, CLUSTER, PCTFREE, MINPCTUSED, DISALLOW REVERSE SCANS, and ALLOW REVERSE SCANS parameters in the CREATE INDEX statement.
- UNIQUE should be specified only if the data for the index key contains unique values for every row of the data source table. The uniqueness will not be checked.
- The sum of the stored lengths of the specified columns must not be greater than 1024.
- No LOB column, DATALINK column, or distinct type column based on a LOB or DATALINK may be used as part of an index. This restriction is enforced even if the length attribute of the column is small enough to fit within the 1024-byte limit.

A nickname is created on a view:

Suppose that you create the nickname *jp_sales2002* for a view called JAPAN_SALES2002. The underlying table for this view is the JAPAN_SALES table which contains several indexes: REGION, AMOUNT, SALES_REP. The CREATE INDEX statement you create will reference the nickname for the view and contain information about the index of the underlying table for the view.

When creating an index specification for a view, make certain the column or columns that the table index is based on is part of the view. If you want to create index specifications for all indexes on the underlying table, each index

specification will have to be created separately. For example, to create an index specification that describes the REGION index, the syntax would be:

```
CREATE UNIQUE INDEX jp_2002_region ON jp_sales2002
(REGION) SPECIFICATION ONLY
```

where *jp_2002_region* is the index name, and *jp_sales2002* is the nickname for the view JAPAN_SALES2002.

Related concepts:

- "Index specifications" on page 22

Related tasks:

- "Creating index specifications for data source objects" on page 216
- "Creating index specifications on tables that acquire new indexes" on page 218
- "Creating index specifications on Informix synonyms" on page 221
- "Nickname characteristics affecting global optimization" on page 249

Related reference:

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

Creating index specifications on Informix synonyms

In Informix, you can create a synonym for a table or view. While the DB2 federated server allows you to create nicknames for Informix synonyms, the action that the federated server takes depends on whether the synonym is based on a table or a view:

- Suppose that a nickname is created for a synonym, and the synonym is based on an Informix table. If the federated server determines that the table the synonym refers to has an index, then an index specification is created for the synonym. If the table that the synonym refers to does not have an index, then no index specification is created for the synonym. However you can create an index specification manually, using the CREATE INDEX statement.
- Suppose the a nickname is created for a synonym, and the synonym is based on an Informix view. The federated server can not determine which underlying table or tables the view is based on. Therefore no index specification is created for the synonym. However you can create an index specification manually using the CREATE INDEX statement.

Prerequisites:

The privileges held by the authorization ID of the statement must include at least one of the following:

- SYSADM or DBADM authority
- One of: CONTROL privilege on the object or INDEX privilege on the object. And one of: IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name of the index does not exist, or CREATEIN privilege on the schema, if the schema name of the index refers to an existing schema.

Restrictions:

There are some restrictions when creating an index on a nickname.

- If the bind option DYNAMICRULES BIND applies, the statement cannot be dynamically prepared. Also, you cannot use the INCLUDE, CLUSTER, PCTFREE, MINPCTUSED, DISALLOW REVERSE SCANS, and ALLOW REVERSE SCANS parameters in the CREATE INDEX statement.
- UNIQUE should be specified only if the data for the index key contains unique values for every row of the data source table. The uniqueness will not be checked.
- The sum of the stored lengths of the specified columns must not be greater than 1024.
- No LOB column, DATALINK column, or distinct type column based on a LOB or DATALINK may be used as part of an index. This restriction is enforced even if the length attribute of the column is small enough to fit within the 1024-byte limit.

A nickname is created on an Informix synonym that is based on a table:

When the synonym is based on an Informix table that does not contain an index, you can create an index specification for the synonym to tell the optimizer which column or columns to search on to find data quickly. The statement you create will specify the nickname for the synonym, and you will supply information about the column or columns in the table that the synonym is based on. Suppose that you create the nickname *contracts* for a synonym called SALES_CONTRACTS and the table that this synonym is based on is called table which contains several indexes: REGION, AMOUNT, SALES_REP. The CREATE INDEX statement you create will reference the nickname for the view and contain information about the index of the underlying table for the view.

A nickname is created on an Informix synonym that is based on a view:

Suppose that you create the nickname *jp_sales2002* for a view called JAPAN_SALES2002. The underlying table for this view is the JAPAN_SALES table which contains several indexes: REGION, AMOUNT, SALES_REP. The

CREATE INDEX statement you create will reference the nickname for the view and contain information about the index of the underlying table for the view.

When creating an index specification for a view, make certain that the column or columns the table index is based on, is part of the view. If you want to create index specifications for all indexes on the underlying table, each index specification will have to be created separately.

To create an index specification that describes REGION index, the syntax would be: `CREATE UNIQUE INDEX jp_2002_region ON jp_sales2002 (REGION) SPECIFICATION ONLY` where *jp_2002_region* is the index name and *jp_sales2002* is the nickname for the view JAPAN_SALES2002.

Related concepts:

- "Index specifications" on page 22

Related tasks:

- "Creating index specifications for data source objects" on page 216
- "Creating index specifications on tables that acquire new indexes" on page 218
- "Creating index specifications on views" on page 219
- "Nickname characteristics affecting global optimization" on page 249

Related reference:

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

Creating and modifying function mappings

Create a mapping between a federated database function or function template, and a data source function. The mapping can associate the federated database function or template with a function at either a specified data source or a range of data sources. For example, you can create a function mapping for all data sources of a particular type and version. When you create a function mapping, it disables the default mapping between a federated database function and a data source function

You can override the default mappings to:

- Change a function mapping for all data source objects located on a specific server
- Change a function mapping for a specific data source type
- Provide function mapping statistical information to the optimizer
- Disable default function mappings

Use the CREATE FUNCTION MAPPING statement to create a function or function template.

Prerequisites:

The authorization ID of the statement must have SYSADM or DBADM authority.

Restrictions:

- For a function or function template that has any input parameters, the *data_type* specifies the data type of such a parameter. The data type cannot be LONG VARCHAR, LONG VARGRAPHIC, DATALINK, a large object (LOB) type, or a user-defined type.
- If the data source function has input parameters:
  - The DB2 counterpart function must have the same number of input parameters that the data source function has.
  - The data types of the input parameters for the DB2 counterpart function must be compatible with the corresponding data types of the input parameters for data source function.
- If the data source function has no input parameters:
  - The DB2 counterpart function cannot have any input parameters.

For the federated server to recognize a data source function, the function must be mapped against an existing DB2 function. 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. The default function mappings from DB2 for UNIX and Windows functions to DB2 for z/OS functions are in the DRDA wrapper. The default function mappings from DB2 for UNIX and Windows functions to Sybase functions are in the CTLIB and DBLIB wrappers, and so forth.

For some non-relational data sources, the wrappers do not contain the default function mappings. The 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.

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. If a DB2 counterpart function does not exist, you must create one on the DB2 federated server. :

The DB2 counterpart function can be either a complete function or a function template.

A *function template* is a DB2 function that you create to invoke a function on a data source. The federated server recognizes a data source function when there is a mapping between the data source function and a counterpart function at the federated database. You can create a function template to act as the counterpart when no counterpart exists.

However, unlike a regular function, a function template has no executable code. After you create a function template, you must then create the function mapping between the template and the data source function. You create a function template with the CREATE FUNCTION statement, using the AS TEMPLATE parameter. You create a function mapping by using the CREATE FUNCTION MAPPING statement. When the federated server receives queries which specify the function template, the federated server will invoke the data source function.

Note: When you create a function mapping, it is possible that the return values from a function evaluated at the data source will be different than the return values from a compatible function evaluated at the DB2 federated database. DB2 will use the function mapping, but this might result in a SQL syntax error or unexpected results.

Change a function mapping for all data source objects located on a specific server:

Suppose that you want to map a function template called BONUS to a UDF, also called BONUS, that is used at an Oracle data source called ORACLE1.

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN2 FOR BONUS()
 SERVER ORACLE1 OPTIONS (REMOTE_NAME 'BONUS')
```

Change a function mapping for a specific data source type:

Suppose that you want to map a function template to a UDF that all Oracle data sources can access. The template is called STATS and belongs to a schema called NOVA. The Oracle UDF is called STATISTICS and belongs to a schema called STAR.

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN1 FOR NOVA.STATS ( DOUBLE, DOUBLE )
 SERVER TYPE ORACLE OPTIONS (REMOTE_NAME 'STAR.STATISTICS')
```

Provide function mapping statistical information to the optimizer:

Suppose that you want to map the local function UCASE(CHAR) to a UDF that is used at an Oracle data source called ORACLE2. You want to include the estimated number of instructions per invocation of the Oracle UDF. The syntax is:

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN4 FOR SYSFUN.UCASE(CHAR)
 SERVER ORACLE2 OPTIONS (REMOTE_NAME 'UPPERCASE', 99 INSTS_PER_INVOC '1000')
```

Disabling default function mappings:

Default function mappings can be rendered inoperable by disabling them (they cannot be dropped). To disable a default function mapping, the CREATE FUNCTION MAPPING statement specifies the name of the DB2 function and sets the DISABLE option to 'Y'.

Assume that there is a default function mapping between the DB2 WEEK function and a similar function on Oracle data sources. When a query that requests Oracle data and that references WEEK is processed, either function might be invoked. The function invoked depends on which function is estimated by the optimizer to require less overhead. Suppose that you want to determine how performance would be affected if only the WEEK function is invoked for such queries. To ensure that WEEK is invoked each time, you must disable the mapping. The syntax is:

```
CREATE FUNCTION MAPPING FOR SYSFUN.WEEK(INT)
TYPE ORACLE OPTIONS (DISABLE 'Y')
```

Related concepts:

- "Function mappings and function templates" on page 20
- "Function mappings options" on page 22

Related reference:

- "CREATE FUNCTION MAPPING statement" in the *SQL Reference, Volume 2*
- Appendix F, "Function mapping options for federated systems" on page 301

Creating and modifying remote tables using transparent DDL

Transparent DDL is a capability of DB2 federation that assists database administrators in creating and modifying tables that reside on remote data sources without needing to use PASSTHRU sessions. Transparent DDL allows you to affect table definitions.

The SQL statements you use with transparent DDL are CREATE TABLE, ALTER TABLE, and DROP TABLE.

A transparent DDL CREATE TABLE statement creates a remote table at the data source and a nickname for that table at the federated server, in the same transaction. It will map the DB2 data types you specify to the remote data types using the default reverse type mappings. 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. If you are using the Oracle SQLNET wrapper and specify the DB2 CLOB data type, it will be mapped to the Oracle LONG data type.

The advantage of using transparent DDL, is that a DB2 database administrator can use procedures they are familiar with to create both local and remote tables. The database administrator can either use the DB2 Control Center or DDL statements in the DB2 command line processor (CLP) to create the tables. This avoids the need to learn the different DDL syntax required for each data source.

Recommendation: Use the remote table wizard in the DB2 Control Center to create remote tables.

Prerequisites:

Before you can create remote tables on a data source, you need to setup the configuration to that data source:

- The wrapper needs to be created for that data source type.
- The server definition needs to be created for the server where the remote table will be located.
- The user mappings need to be created between DB2 and the data source server.

The privileges held by the authorization ID of the transparent DDL statements must include at least one of the following:

- SYSADM or DBADM authority
- CREATETAB authority on the database and USE privilege on the table space as well as one of:
  - IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name of the table does not exist
  - CREATEIN privilege on the schema, if the schema name of the table refers to an existing schema

Restrictions:

Transparent DDL has some limitations:

- You can only create base tables on the remote data source. You cannot specify tablespaces.
- You can specify basic column information in the table definition, but you will not be able to specify table options or column options. For example, the LOB options (LOGGED and COMPACT) are not supported.
- You cannot specify a comment on a column.
- You cannot generate column contents.
- You can specify a primary key, but you cannot specify a foreign key or check constraints.
- You cannot create indexes for the table.
- Data capture is not supported.
- You cannot drop a column from a remote table created using transparent DDL.
- You cannot modify the parameters of existing columns, such as the data type or length.
- You cannot modify or drop tables that were created at the remote data source.

See the *DB2 SQL Reference* for the complete syntax for the CREATE TABLE, ALTER TABLE, and DROP TABLE statements.

Creating new remote tables using transparent DDL

To create a remote table using transparent DDL, you can use either the DB2 Control Center wizard or the CREATE TABLE statement. Use the remote table wizard in the DB2 Control Center to avoid specifying a parameter or option that is not supported. Through the wizard you can specify columns by selecting from a list of predefined columns, or by specifying the attributes for a new column.

Suppose you want to create the table EMPLOYEE on an Oracle server. Using the CLP to create the table, the syntax is:

```
CREATE TABLE EMPLOYEE
   EMP_NO       SMALLINT NOT NULL,
   NAME         VARCHAR(9),
   DEPT         SMALLINT,
   REGION       VARCHAR(12)
   JOB          CHAR(5),
   HIREDATE     DATE,
   SALARY       DECIMAL(7,2),
   PRIMARY KEY (EMP_NO)
   OPTIONS (REMOTE_SERVER 'ORASERVER',
   REMOTE_TABNAME 'EMPLOY', REMOTE_SCHEMA 'J15USER1')
```

Altering remote tables that were created transparent DDL

The ALTER TABLE statement is used to modify tables created using transparent DDL. You cannot modify tables that were created at the remote data source. Using the ALTER TABLE statement you can:

- Create new columns
- Modify the table primary key.

However, transparent DDL does impose some limitations on the modifications you can make with the ALTER TABLE statement:

- You cannot drop a column from a remote table.
- You cannot rearrange the existing columns in the table. You can specify only the placement of new columns you add.

To alter a remote table using transparent DDL, you can use either the DB2 Control Center or the ALTER TABLE statement. Use the DB2 Control Center to avoid specifying a parameter or option that is not supported.

Suppose you want to change the primary key on a remote table EMPLOYEE on an Oracle server that you created using transparent DDL. Using the CLP to modify the table, the syntax is:

```
ALTER TABLE EMPLOYEE
   PRIMARY KEY (EMP_NO, REGION)
```

Suppose you want to add the columns ORDER_DATE and SHIP_DATE to the remote table SPALTEN that was created using transparent DDL. Using the CLP to create the table, the syntax is:

```
ALTER TABLE SPALTEN ADD COLUMN
  ORDER_DATE    DATE
  SHIP_DATE    DATE
```

Dropping remote tables that were created transparent DDL

To drop a remote table using transparent DDL, you can use either the DB2 Control Center or the DROP TABLE statement. Dropping a remote table that was created using transparent DDL also drops the corresponding nickname for that table, and invalidates plans on that nickname.

To drop the table SPALTEN, the syntax is:

```
DROP TABLE SPALTEN
```

Related concepts:

- "Fast track to configuring your data sources" on page 85

Related tasks:

- "Altering a remote table : Federated Systems help" in the *Help: Federated Systems*

- "Altering the columns in a remote table : Federated Systems help" in the *Help: Federated Systems*
- "Creating a remote table : Federated Systems help" in the *Help: Federated Systems*

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "CREATE TABLE statement" in the *SQL Reference, Volume 2*
- "DROP statement" in the *SQL Reference, Volume 2*
- Appendix I, "Default reverse data type mappings" on page 323

Chapter 14. Tuning and performance issues with a federated system

Performance problems can originate at either federated database, the data sources, or both. A bottleneck at either the federated database or the data sources can degrade performance. Isolating problems involves tuning the federated database and data sources for maximum performance. It may require tuning queries, applications, configuration parameters, and network usage to solve these problems.

Tuning query processing

To obtain data from data sources clients (users and applications) submit queries in DB2® SQL to the federated database. The DB2 SQL Compiler then consults information in the global catalog and the data source wrapper to help it process the query. This includes information about connecting to the data source, server attributes, mappings, index information, and processing statistics.

As part of the SQL Compiler process, the *query optimizer* analyzes a query. The Compiler develops alternative strategies, called *access plans*, for processing the query. The access plans might call for the query to be:

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

DB2 evaluates the access plans primarily on the basis of information about the data source capabilities and the data. The wrapper and the global catalog contain this information. DB2 decomposes the query into segments that are called *query fragments*. Typically it is more efficient to *pushdown* a query fragment to a data source, if the data source can process the fragment. However, the query optimizer takes into account other factors such as:

- The amount of data that needs to be processed.
- The processing speed of the data source.
- The amount of data that the fragment will return.
- The communication bandwidth.

Note: Pushdown analysis is only performed on relational data sources. Pushdown analysis does not determine how a query can be pushed down for non-relational data sources.

The following figure illustrates the steps performed by the SQL Compiler when it processes a query.

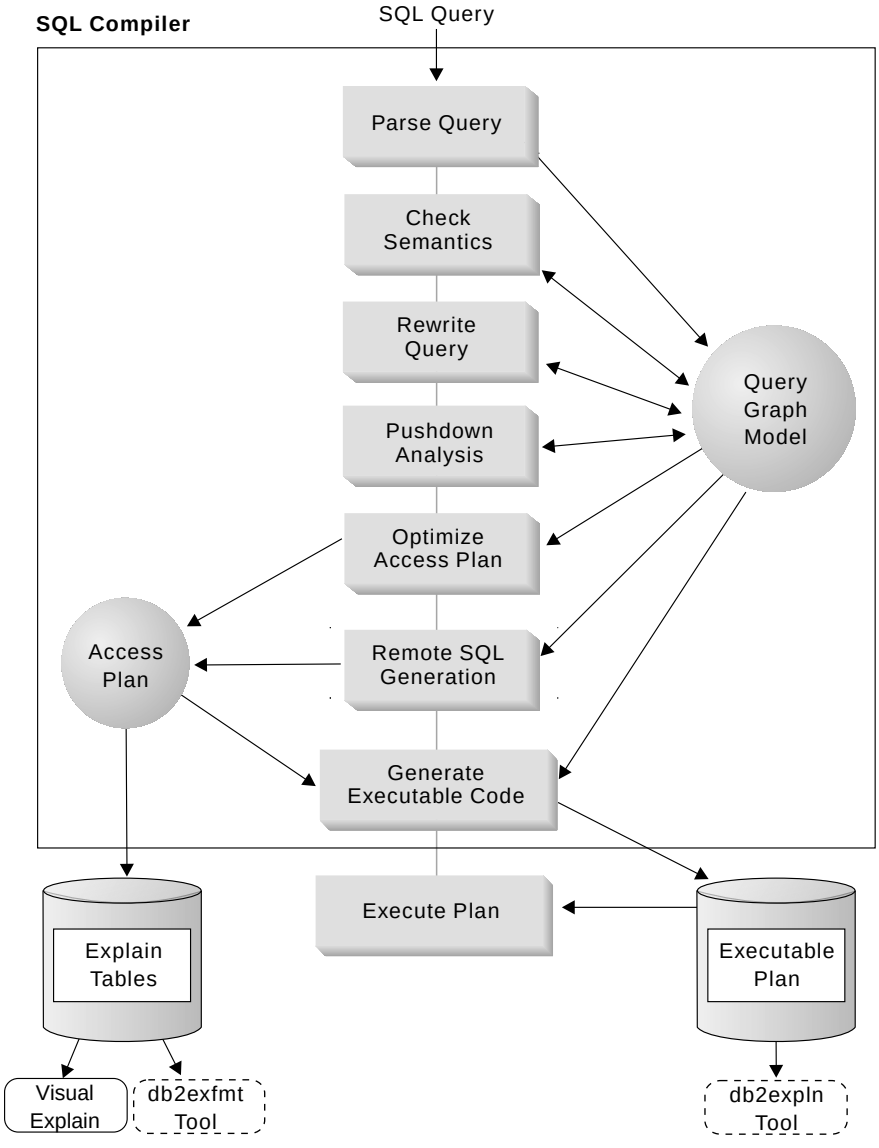

Figure 6. SQL Compiler query analysis flowchart

The query optimizer generates local and remote access plans for processing a query fragment, based on resource cost. DB2 then chooses the plan it believes will process the query with the least resource cost.

If any of the fragments are to be processed by data sources, DB2 submits these fragments to the data sources. After the data sources process the fragments, the results are retrieved and returned to DB2. If DB2 performed any part of the processing, it combines its results with the results retrieved from the data source. DB2 then returns all results to the client.

The major task of pushdown analysis is to recommend to the optimizer whether an operation can be remotely evaluated. It is just a recommendation. The optimizer might choose to not perform an operation directly on a remote data source because it is less cost-effective. A secondary task of pushdown analysis is to attempt to transform the query into a form that can be better optimized by both the DB2 optimizer and remote query optimizers.

The final access plan selected by the optimizer can consist of operations at the remote data source. For those operations that will be performed by each data source, remote SQL generation creates an efficient SQL statement based on the data source SQL dialect. This helps produce an optimal plan for the query for all data sources, and is called *global optimization*.

For non-relational sources, the wrappers are responsible for their own statement generation.

Related concepts:

- "Pushdown analysis" on page 233

Related tasks:

- "Global optimization" on page 246

Pushdown analysis

Pushdown analysis is performed on relational data sources. Pushdown analysis tells the query optimizer if a remote data source can perform an operation. An *operation* can be a function, such as relational operator, system or user functions, or an SQL operator (GROUP BY, ORDER BY, and so on).

Functions that cannot be pushed-down, can significantly impact query performance. Consider the effect of forcing a selective predicate to be evaluated locally instead of at the remote data source. This approach could require the federated server to retrieve the entire table from the remote data source, and then filter the table locally against the predicate. If your network is constrained—and the table is large—query performance could suffer.

Operators that are not pushed-down can also significantly impact query performance. For example, having a GROUP BY operator aggregate remote

data locally could, once again, require the federated server to retrieve the entire table from the remote data source.

For example, suppose that the nickname EMP references the table EMPLOYEE. This table has 10,000 rows. One column contains the last names of employees, and one column contains the salary for each employee. The following query is sent to the federated server to compute the number of employees with a last name that starts with 'B' who are paid higher than 50,000.

```
SELECT LASTNAME, COUNT (*) FROM EMP
 WHERE LASTNAME = 'B' AND SALARY > 50000
 GROUP BY LASTNAME;
```

When the DB2® SQL Compiler receives this statement, it considers several possibilities:

- The collating sequences are the same. It is likely that the query predicate will be pushed-down to the data source. It is usually more efficient to filter and group results at the data source instead of copying the entire table to the federated server and performing the operations locally. Pushdown analysis determines if operations can be performed at the data source. Since the collating sequences are the same, the predicate and the GROUP BY operation can take place at the data source.
- The collating sequences are the same, and the query optimizer knows that the federated server is very fast. It is possible that the query optimizer will decide that performing the GROUP BY operation locally is the best (least cost) approach. The predicate will be pushed-down to the data source for evaluation. This is an example of pushdown analysis combined with global optimization.
- The collating sequences are not the same. Pushdown analysis will determine that the entire WHERE clause cannot be evaluated at the data source. However, the query optimizer might decide it is more efficient to pushdown the SALARY > 50000 portion of the predicate. The range comparison must still be done at the federated database. This is another example of pushdown analysis combined with global optimization.

The SQL Compiler will consider the available access plans, and then choose the plan that is the most efficient.

In general, the goal is to ensure that the query optimizer considers pushing down the functions and operators to the data sources for evaluation. Many factors can affect whether a function or an SQL operator is evaluated at a remote data source. The key factors which influence the query optimizer are: server characteristics, nickname characteristics, and query characteristics.

Related concepts:

Pushdown analysis-details

Server characteristics affecting pushdown opportunities

The following sections contain data source-specific factors that can affect pushdown opportunities. In general, these factors exist because you use the DB2® SQL dialect to submit queries and the DB2 dialect may offer more functionality than the data source SQL dialect. The DB2 federated server can compensate for the lack of function at a data server, but doing so may require that the operation take place at the federated server.

SQL differences

- SQL capabilities. Each data source supports a variation of the SQL dialect and different levels of functionality. For example, consider the GROUP BY list. Most data sources support the GROUP BY operator. However some data sources have restrictions on the number of items on the GROUP BY list, or restrictions on whether an expression is allowed on the GROUP BY list. If there is a restriction at the remote data source, the federated server might have to perform the GROUP BY operation locally.
- SQL restrictions. Each data source can have different SQL restrictions. For example, some data sources require parameter markers to bind in values to remote SQL statements. Therefore, parameter marker restrictions must be checked to ensure that each data source can support such a bind mechanism. If the federated server cannot determine a good method to bind in a value for a function, this function must be evaluated locally.
- SQL limitations. The federated server might allow the use of larger integers than the remote data sources. However, limit-exceeding values cannot be embedded in statements that are sent to the data sources. Therefore, the function or operator that operates on this constant must be evaluated locally.
- Server specifics. Several factors fall into this category. One example is sorting NULL values (highest, or lowest, depending on the ordering). For example, if the NULL value is sorted at a data source differently from the federated serve, ORDER BY operations on a nullable expression cannot be remotely evaluated.

Collating sequence

If you set the COLLATING_SEQUENCE server option to "Y", you are telling the federated database that the data source collating sequence matches the DB2 collating sequence. This setting allows the optimizer to consider order

dependent processing at a data source, which can improve performance. Set the COLLATING_SEQUENCE server option value for a data source to 'Y' when one of the following is true:

1. All data source character data is upper case
2. All data source character data is lower case
3. All data source character data is composed of numbers0 through 9 only, with no other characters such as blanks

If the data source collating sequence is not the same as the federated database collating sequence, you can receive incorrect results. For example, if your plan uses merge joins, the optimizer will push down ordering operations to the data sources as much as possible. If the data source collating sequence is not the same, the join results may not have a correct result set. Set the COLLATING_SEQUENCE server option to "N" if you are not sure that the collating sequence at the data source is identical to the DB2 collating sequence.

Alternatively, you can configure a federated database to use the same collating sequence that a data source uses. You then set the COLLATING_SEQUENCE server option to 'Y'. This allows the optimizer to consider ″pushing-down″ character range comparison predicates.

To determine if a data source and DB2 have the same collating sequence, consider the following factors:

- National language support

  The collating sequence is related to the language supported on a server. Compare the DB2 NLS information for your operating system to the data source NLS information.
- Data source characteristics

  Some data sources are created using case-insensitive collating sequences, which can yield different results from DB2 in order-dependent operations.
- Customization

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

When a query from a federated server requires sorting, the place where the sorting is processed depends on several factors. If the federated database's collating sequence is the same as the data source collating sequence, the sort can take place at the data source. If the collating sequences are the same, the query optimizer can decide which is the most efficient way to complete the query—a local sort or a remote sort. Likewise, if a query requires a comparison of character data, this comparison can also be performed at the data source.

Numeric comparisons, in general, can be performed at either location even if the collating sequence is different. You may get incorrect results, however, if the weighting of null characters is different between the federated database and the data source. Likewise, for comparison statements, be careful if you are submitting statement to a case-insensitive data source. The weights assigned to the characters ″I″ and ″i″ in a case-insensitive data source are the same. For example, in a case-insensitive data source with an English code page, **STEWART**, **SteWArT**, , and **stewart** would all be considered equal. The DB2 federated database, by default, is case-sensitive and would assign different weights to the characters.

If the collating sequences of the federated database and the data source differ, the federated server retrieves the data to the federated database, so that it can do the sorting and comparison locally. The reason is that users expect to see the query results ordered according to the collating sequence defined for the federated server; by ordering the data locally, the federated server ensures that this expectation is fulfilled.

If your query contains an equal sign, it is possible to push-down that portion of the query even if the collating sequences are different (set to 'N'). For example, the predicate C1 = 'A' could be pushed-down to a data source. Of course, such queries cannot be pushed-down when the collating sequence at the data source is case-insensitive. When a data source is case-insensitive, the results from C1= 'A' and C1 = 'a' are the same, which is not acceptable in a case-sensitive environment (DB2).

Administrators can create federated databases with a particular collating sequence that matches the data source collating sequence. This approach may speed performance if all data sources use the same collating sequence or if most or all column functions are directed against data sources that use the same collating sequence.

Retrieving data for local sorts and comparisons usually decreases performance. Therefore, consider configuring the federated database to use the same collating sequences that your data sources use. That way, performance might increase, because the federated server can allow sorts and comparisons to take place at data sources. For example, in DB2 for z/OS™ and OS/390, sorts defined by ORDER BY clauses are implemented by a collating sequence based on an EBCDIC code page. If you want to use the federated server to retrieve DB2 for z/OS and OS/390® data sorted in accordance with ORDER BY clauses, it is advisable to configure the federated database so that is uses a predefined collating sequence based on the EBCDIC code page.

If the collating sequences at the federated database and the data source differ, and you need to see the data ordered in the data source's sequence, you can submit your query in a pass-through session, or define the query in a data source view.

Federated server options

The previously listed factors that affect pushdown opportunities are characteristics of the database servers, and you can not change them. The following server options can be set by you, and in some cases can affect query performance:

- COLLATING_SEQUENCE. If a data source has a collating sequence that differs from the DB2 for UNIX® and Windows® collating sequence, any operation depending on the DB2 collating sequence cannot be remotely evaluated at a data source. An example is executing MAX column functions against a nickname character column at a data source with a different collating sequence. Because results might differ if the MAX function is evaluated at the remote data source, the federated database will perform the aggregate operation and the MAX function locally.
- VARCHAR_NO_TRAILING_BLANKS. This option is for varying-length character strings that contain no trailing blanks. Some data sources, such as Oracle, have non-blank-padded comparison semantics that return the same results as the DB2 for UNIX and Windows comparison semantics. If you are certain that all VARCHAR and VARCHAR2 columns at a data source contain no trailing blanks, consider creating this server option for a data source. Ensure that you consider all objects that can potentially have nicknames, including views.

Type and function mapping factors

The default data type mappings are built into the data source wrappers. These mappings are designed so that sufficient buffer space is given to each data source data type to avoid runtime buffer overflow. You can customize the type mapping for a specific data source to suit specific applications. For example, if you are accessing an Oracle data source column with a DATE data type it will be mapped by default to the DB2 for UNIX and Windows TIMESTAMP data type. You can change the local data type to the DB2 for UNIX and Windows DATE data type. This change bypasses the use of a SCALAR function to extract a subset of the total data stored in the TIMESTAMP data type.

The default function mappings are also built into the data source wrappers. The federated database will compensate for functions that are not supported by a data source. There are three cases where function compensation will occur:

- A function simply does not exist at the data source. Some of the SYSFUN functions, for example, do not exist on DB2 for z/OS and OS/390 data sources, and thus require local compensation.

- A function exists at the data source; however, the characteristics of the operand violate function restrictions. An example is the IS NULL relational operator. Most data sources support it, but some have restrictions such as only allowing a column name on the left hand side of the IS NULL operator.
- A function, if evaluated remotely, may return a different result. An example is the '>' (greater than) operator. For those data sources with different collating sequences, the greater than operator may return different results than if it is evaluated locally by DB2 for UNIX and Windows.

Related concepts:

- "Collating Sequences" in the *Application Development Guide: Programming Client Applications*
- "Tuning query processing" on page 231
- "Pushdown analysis" on page 233
- "Nickname characteristics affecting pushdown opportunities" on page 239
- "Query characteristics affecting pushdown opportunities" on page 241

Related tasks:

- "Creating and modifying function mappings" on page 223
- "Modifying default data type mappings" on page 208
- "Accessing data sources using PASSTHRU" on page 181

Related reference:

- Appendix C, "Server options for federated systems" on page 287

Nickname characteristics affecting pushdown opportunities

There are several nickname-specific factors that can affect pushdown opportunities. The local data type of a nickname column can affect the number of possibilities in a joining sequence evaluated by the optimizer. Nicknames can be flagged with a column option to indicate the columns contain no trailing blanks. This gives the SQL Compiler the opportunity to generate a more efficient form of a predicate for the SQL statement sent to the data sources.

Local data type of a nickname column

Ensure that the local data type of a column does not prevent a predicate from being evaluated at the data source. The default data type mappings are provided to avoid any possible overflow. However, a joining predicate between two columns of different lengths might not be considered at the data source whose joining column is shorter, depending on how DB2® binds in the longer column. This situation can affect the number of possibilities in a joining sequence evaluated by the optimizer. For example, Oracle data source

columns created using the INTEGER or INT data type are given the type NUMBER(38). A nickname column for this Oracle data type will be given the local data type FLOAT because the range of a DB2 integer is from 2**31 to (-2**31)-1, which is roughly equal to NUMBER(9). In this case, joins between a DB2 integer column and an Oracle integer column cannot take place at the DB2 data source (shorter joining column). However, if the domain of this Oracle integer column can be accommodated by the DB2 INTEGER data type, change its local data type with the ALTER NICKNAME statement so that the join can take place at the DB2 data source.

Federated 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. DB2 treats the object that a nickname references as if it is a table. As a result, you can set column options for any data source object that you create a nickname for. The ALTER NICKNAME statement can be used to add or change column options for nicknames. There are two column options:

- NUMERIC_STRING. This column option applies to character type columns (CHAR and VARCHAR). 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 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.
- VARCHAR_NO_TRAILING_BLANKS. Unlike the server option with the same name, this column option can be used to identity specific Oracle columns that contain no trailing blanks. The SQL Compiler pushdown analysis step will then take this information into account when checking all operations performed on columns which have this setting. Based on the VARCHAR_NO_TRAILING_BLANKS setting, the SQL Compiler may generate a different but equivalent form of a predicate used in the remote SQL statement sent to the data source. You might see a different predicate being evaluated against the data source, but the net result should be equivalent.

Related concepts:

- "Tuning query processing" on page 231
- "Pushdown analysis" on page 233
- "Server characteristics affecting pushdown opportunities" on page 235

- "Query characteristics affecting pushdown opportunities" on page 241

Related tasks:

- "Altering a nickname" on page 199

Related reference:

- Appendix E, "Column options for federated systems" on page 299

Query characteristics affecting pushdown opportunities

A query can reference a SQL operator that involves nicknames from multiple data sources. When the federated server combines the results from two referenced data sources by using one operator, the operation must take place at the federated server. An example of this is a set operator, like UNION. The operator cannot be evaluated at a remote data source directly.

Related concepts:

- "Server characteristics affecting pushdown opportunities" on page 235
- "Nickname characteristics affecting pushdown opportunities" on page 239

Pushdown analysis decisions

Rewriting your SQL statements can provide additional pushdown opportunities when the federated server processes queries. To help determine the optimal SQL rewrites, the following sections introduce several tools you can use to determine where a query is evaluated for pushdown, list common questions (and suggested areas to investigate) associated with query analysis, and address data source upgrade issues.

Analyzing where a query is evaluated

Detailed query optimizer information is kept in explain tables separate from the actual access plan itself. This information allows for in-depth analysis of an access plan. The explain tables are accessible on all supported operating systems, and contain information for both static and dynamic SQL statements. You can access the explain tables using SQL statements. This allows for easy manipulation of the output, for comparison among different queries, or for comparisons of the same query over time.

Procedure:

There are three ways to get information from the explain tables.

- Explain table format tool. Use the **db2exfmt** tool to present the information from the explain tables in a predefined format.

- Visual Explain. Use the **db2dd** or the **db2vexp** command to start Visual Explain. Use this tool to view the query access plan graph. In case of a SELECT query involving a nickname, you will always see a SHIP operator. A statement may or may not be associated with the SHIP. If the operators below SHIP are being processed by the remote data source, then there will be a remote SQL statement associated with SHIP. If the operators below SHIP are being processed by the federated server, then the remote SQL statement will not be associated with SHIP. In the case of INSERT, UPDATE or DELETE operations involving a nickname, you may or may not see a SHIP. If the INSERT, UPDATE or DELETE statement is completely evaluated by the remote data source, then you will not see a SHIP in the access plan. In this case, the remote SQL statement will be found in the RETURN operator (the top operator).
- SQL Explain. Use the **db2expln** or the **dynexpln** command to start SQL Explain. Use this tool to view the access plan strategy as text. SQL explain does not provide a graphical user interface. When the query optimizer generates a remote plan, it takes into consideration different factors regarding the capability of remote data source. The optimizer expects that the remote data source will use a plan similar to the plan it generates. But the remote data source is free to choose and use any plan it sees fit. The SQL Explain tool does not display the remote part of the access plans.

Related concepts:

- "Explain tools" in the *Administration Guide: Performance*
- "SQL explain tools" in the *Administration Guide: Performance*
- "db2expln syntax and parameters" in the *Administration Guide: Performance*
- "dynexpln" in the *Administration Guide: Performance*
- "Description of db2expln and dynexpln output" in the *Administration Guide: Performance*
- "Tuning query processing" on page 231
- "Pushdown analysis" on page 233
- "Understanding access plan optimization decisions" on page 253
- "Data source upgrades and customization" on page 245

Related tasks:

- "Global optimization" on page 246

Understanding access plan evalutation decisions

This section lists typical access plan analysis questions, and areas you can investigate to increase pushdown opportunities.

Why isn't this predicate being evaluated remotely?

This question arises when a predicate is very selective and thus could be used to filter rows and reduce network traffic. Remote predicate evaluation also affects whether a join between two tables of the same data source can be evaluated remotely.

Areas to examine include:

- Subquery predicates. Does this predicate contain a subquery that pertains to another data source? Does this predicate contain a subquery involving an SQL operator that is not supported by this data source? Not all data sources support set operators in a predicate
- Predicate functions. Does this predicate contain a function that cannot be evaluated by this remote data source? Relational operators are classified as functions.
- Predicate bind requirements. Does this predicate, if remotely evaluated, require bind-in of some value? If so, would it violate SQL restrictions at this data source?
- Global optimization. The optimizer may have decided that local processing is more cost-effective.

Why isn't the GROUP BY operator evaluated remotely?

There are several areas you can check:

- Is the input to the GROUP BY operator evaluated remotely? If the answer is no, examine the input.
- Does the data source have any restrictions on this operator? Examples include:
  - Limited number of GROUP BY items
  - Limited byte counts of combined GROUP BY items
  - Column specification only on the GROUP BY list
- Does the data source support this SQL operator?
- Global optimization. The optimizer may have decided that local processing is more cost-effective.

Why isn't the SET operator evaluated remotely?

There are several areas you can check:

- Are both of its operands completely evaluated at the same remote data source? If the answer is no and it should be yes, examine each operand.
- Does the data source have any restrictions on this SET operator? For example, are large objects or long fields valid input for this specific SET operator?

Why isn't the ORDER BY operation evaluated remotely?

Consider:

- Is the input to the ORDER BY operation evaluated remotely? If the answer is no, examine the input.
- Does the ORDER BY clause contain a character expression? If yes, does the remote data source have a different collating sequence than the federated server collating sequence?
- Does the data source have any restrictions on this operator? For example, is there a limited number of ORDER BY items? Does the data source restrict column specification to the ORDER BY list?

Why is a remote INSERT with a fullselect statement not completely evaluated remotely?

Consider:

- Could the subselect be completely evaluated on the remote data source? If no, examine the subselect.
- Does the subselect contain a set operator? If yes, does this data source support set operators as input to an INSERT?
- Does the subselect reference the target table? If yes, does this data source allow this syntax?

Why is a remote INSERT with VALUES clause statement not completely evaluated remotely?

Consider:

- Can the VALUES clause be completely evaluated at the remote data source? In other words, does an expression contain a function not supported by the remote data source?
- Does the expression involve a scalar subquery? Is that syntax supported?
- Does the expression reference the target table? Is that syntax supported?

Why is a remote, searched UPDATE statement not completely evaluated remotely?

Consider:

- Can the SET clause be completely evaluated at the remote data source? In other words, does an update expression contain a function not supported by the remote data source?
- Does the SET clause involve a scalar subquery? Does the data source allow this syntax?
- Can the search condition be completely evaluated at the remote data source? If the answer is no, examine the search condition instead.
- Does the search condition or SET clause reference the target table? Does the data source allow this syntax?
- Does the search condition or SET clause reference the target table with correlation? Does the data source allow this syntax?

Why is a positioned UPDATE statement not completely evaluated remotely?

This happens when DB2® chooses to evaluate the update expression locally before sending the UPDATE statement to the data source. This approach should not significantly affect performance.

- Can the SET clause be completely evaluated at the remote data source? In other words, does an update expression contain a function not supported by the remote data source?
- Does the SET clause involve a scalar subquery? Does the data source allow this syntax?

Why is a remote, searched DELETE statement not completely evaluated remotely?

Consider:

- Can the search condition be completely evaluated at the remote data source? If the answer is no, examine the search condition instead.
- Does the search condition reference the target table? Does the data source allow this syntax?
- Does the search condition reference the target table with correlation? Does the data source allow this syntax?

Related concepts:

- “Pushdown analysis” on page 233
- “Analyzing where a query is evaluated” on page 241

Data source upgrades and customization

The DB2® SQL Compiler relies on information that is stored in the global catalog to provide it with the SQL capabilities of the data sources. This information periodically needs to be updated. The SQL capabilities of the data sources might change in new versions of the data sources. When data sources are upgraded or customized, update the global catalog information so that the SQL Compiler is using the most current information.

Use DB2 SQL DDL statements, such as CREATE FUNCTION MAPPING and CREATE SERVER OPTION, to update the catalog.

Related concepts:

- “Tuning query processing” on page 231
- “Pushdown analysis” on page 233
- “Server characteristics affecting pushdown opportunities” on page 235
- “Nickname characteristics affecting pushdown opportunities” on page 239
- “Query characteristics affecting pushdown opportunities” on page 241
- “Analyzing where a query is evaluated” on page 241

- "Understanding access plan evalutation decisions" on page 242

Related tasks:

- "Modifying server definitions" on page 203
- "Creating and modifying function mappings" on page 223
- "Global optimization" on page 246

Global optimization

The SQL Compiler has two phases which help to produce an optimal access strategy for evaluating a query referencing a remote data source. These phases are: remote SQL generation and global optimization. For a query submitted to the federated database, the access strategy might involve breaking down the original query into a set of query fragments and then combining the results.

Using the output of the pushdown analysis phase as a recommendation, the query optimizer decides where each operation will be evaluated. An operation might be evaluated locally at the DB2 federated server or remotely at the data source. The decision is based on the output of the sophisticated fixed cost model used by the optimizer. This model determines:

- The cost to evaluate the operation.
- The cost to transmit the data or messages between the DB2 federated server and the data sources.

The goal is to produce an optimized query. An optimized query, is a query with an access plan that optimizes the query operations of all data sources, globally across the federated system. *Global optimization* is reached when an access plan with the least cost is selected.

The DB2 SQL Compiler has an optimizer knowledge base that contains data about native data sources. The optimizer does not generate remote access plans that cannot be generated by specific DBMSs. In other words, optimizer avoids generating plans that optimizers at remote data sources cannot understand or accept.

Many factors can affect the output from global optimization and thus affect query performance. The key factors are: server characteristics and nickname characteristics.

Related concepts:

- "Tuning query processing" on page 231
- "Pushdown analysis" on page 233
- "Server characteristics affecting global optimization" on page 247

Related tasks:

- "Nickname characteristics affecting global optimization" on page 249

Global optimization-details

Server characteristics affecting global optimization

You provide the query optimizer with information about the data source server characteristics through the server option settings. The server option settings are part of the data source server definition. You can set server options in the CREATE SERVER statement, when you initially establish the server definition. Use the ALTER SERVER statement to add server options to an existing server definition. The server option settings are stored in the federated database global catalog.

These options are separated into three categories: location options (such as the data source machine name), security options (such as authentication information), and performance options (such as the CPU ratio).

The performance options help the optimizer determine if evaluation operations can be done at data sources. The server options affecting performance that might require your tuning are:

- CPU_RATIO
- IO_RATIO
- COMM_RATE
- COLLATING_SEQUENCE
- PLAN_HINTS

Relative ratio of CPU speed

This value indicates how much faster or slower the data source CPU speed is compared with the DB2® CPU. A low ratio indicates that the data source workstation CPU is faster than the DB2 workstation CPU. For low ratios, the optimizer will consider pushing-down operations that are CPU-intensive to the data source.

Relative ratio of I/O speed

This value indicates how much faster or slower the data source I/O speed is compared with the federated server I/O speed. A low ratio indicates that the data source workstation I/O speed is faster than the DB2 workstation I/O speed. For low ratios, the query optimizer will consider pushing-down I/O-intensive operations to the data source.

Communication rate between the federated server and the data source

A low communication rate indicates slow network communication between the federated server and the data source. Lower communication rates

encourage the query optimizer to reduce the number of messages sent to or from this data source. If the COMM_RATE server option is set to a very small number, the optimizer produces a query requiring minimal network traffic.

Data source collating sequence

Your choice of collating sequence might affect performance of the federated database. Use the COLLATING_SEQUENCE server option to indicate if a data source collating sequence matches the local DB2 federated database collating sequence. DB2 can push down order-dependent processing involving character data to the data source. If a data source collating sequence does not match the federated database collating sequence, the optimizer considers data retrieved from this data source as unordered. DB2 will retrieve the relevant data and do all order-dependent processing on character data locally (which can slow performance). Collating sequence is discussed in the topic *Server characteristics affecting pushdown opportunities.*

Remote plan hints

Use the PLAN_HINTS server option to indicate if plan hints are supported at the data source. 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.

If plan hints are enabled, the query sent to the data source contains additional information. For example, a statement sent to an Oracle optimizer with plan hints could look like this:

```
SELECT /*+ INDEX (table1, tlindex)*/
 col1
 FROM table1
```

The plan hint is the string `/*+ INDEX (table1, tlindex)*/`

Related concepts:

- “Fast track to configuring your data sources” on page 85
- “Server characteristics affecting pushdown opportunities” on page 235
- “Analyzing global optimization” on page 252
- “Understanding access plan optimization decisions” on page 253

Related tasks:

- “Global optimization” on page 246
- “Nickname characteristics affecting global optimization” on page 249

Related reference:

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

- Appendix C, “Server options for federated systems” on page 287

Nickname characteristics affecting global optimization

There are several nickname-specific factors that can affect global optimization, including the index information and the global catalog statistics.

It is important that the index information and global catalog statistical data available to the SQL Compiler is current.

Index specifications

The SQL Compiler uses index information to optimize queries. The index information for a table is initially acquired when the nickname is created for that table. Index information is not gathered for nicknames on objects that do not have indexes such as views, synonyms, or non-relational data source objects.

If a nicknamed object does not have an index, you can create an index specification for it. Index specifications build an index definition in the global catalog. The index specification is not an actual index. Use the CREATE INDEX statement with the SPECIFICATION ONLY clause to create an index specification. The syntax for creating an index specification on a nickname is similar to the syntax for creating an index on a local table.

Consider creating index specifications when:

- A nickname is created for a table that has no index.
- You create a nickname for a data source object that does not contain indexes such as a view or a synonym.
- The remote index is on a column of more than 255 bytes, or contains a total key length in excess of 1024 bytes.
- The remote index is on a LOB column.
- You want to encourage the query optimizer to use a specific nickname as the inner table of a nested loop join. You can create an index on the joining column if none exists.

Consider your needs before issuing CREATE INDEX statements against a nickname for a view.

- If the view is a simple SELECT on a table with an index, creating indexes on the nickname (locally) that match the indexes on the table at the data source can significantly improve query performance.
- If indexes are created locally over views that are not simple SELECT statements (for example, a view created by joining two tables), query performance may suffer.

Suppose that an index is created over a view that is a join of two tables. The optimizer may choose that view as the inner element in a nested loop join. The query will have poor performance because the join will be evaluated several times. An alternative is to create nicknames for each of the tables referenced in the data source view and create a federated view that references both nicknames.

Global catalog 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 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 RUNSTATS) at the data source before you create a nickname.

Catalog statistics describes the overall size of tables and views, and the range of values in associated columns. The information includes the:

- Number of rows in a nickname object
- Number of pages that a nickname occupies
- Highest/lowest values of a column

Note: In this version of DB2, the RUNSTATS utility can not be used to update nickname statistics.

While the federated database can retrieve the statistical data held at a data source, it cannot automatically detect updates to existing statistical data at data sources. Furthermore, federated database has no mechanism for handling object definition or structural changes to objects at the data sources (such as when a column is added to a table).

If the statistical data or structural data for an object has changed, you have two choices for updating the statistics.

- Run the equivalent of RUNSTATS at the data source. Then, drop the current nickname. Re-create the nickname. Use this approach if structural information has changed.
- Manually update the statistics in the SYSSTAT.TABLES catalog view. This approach requires fewer steps but will not work if structural information has changed.

Updating row changes:

If a large number of rows at the data source were added or deleted, the federated database will not be aware of these changes. However you may notice slower performance since the optimizer is making decisions based on nickname information that is no longer accurate. Update the statistics for the nickname so the optimizer has accurate statistics when it develops access plans for processing queries against the data source.

Updating column changes:

If columns at the data source are added, deleted, or altered, you may notice incorrect results or receive an error message. Suppose you have the nickname *EUROSALES* which refers to the *europe* table in a Sybase database. If a new column called *CZECH* is added to the table, the federated database will not be aware of the *CZECH* column. Queries which reference that column will result in an error message.

When there are column changes to a data source object, there are several steps you need to take to update the statistics for that object in the federated database catalog:

1. Run the utility on the data source that is equivalent to DB2 RUNSTATS. This will update the statistics stored in the data source catalog.
2. Drop the current nickname for the data source object using the DROP NICKNAME statement.
3. Re-create the nickname using the CREATE NICKNAME statement.

The nickname will now have updated statistical information consistent with the data source object schema.

Related concepts:

- "Nickname characteristics affecting pushdown opportunities" on page 239
- "Server characteristics affecting global optimization" on page 247

Related tasks:

- "Dropping a nickname" on page 202
- "Creating index specifications for data source objects" on page 216

Global optimization decisions

The following sections introduce several tools you can use for analyzing query optimization, and presents common questions (and suggested areas to investigate) associated with query optimization.

Analyzing global optimization

Detailed query optimizer information is kept in explain tables separate from the actual access plan itself. This information allows for in-depth analysis of an access plan. The explain tables are accessible on all supported operating systems, and contain information for both static and dynamic SQL statements. You can access the explain tables using SQL statements. This allows for easy manipulation of the output, for comparison among different queries, or for comparisons of the same query over time.

Procedure:

There are three ways to get global access plan information from the explain tables.

- Explain table format tool. Use the **db2exfmt** tool to present the information from the explain tables in a predefined format.
- Visual Explain. Use the **db2dd** or the **db2vexp** command to start Visual Explain. Use this tool to view the query access plan graph. In case of a SELECT query involving a nickname, you will always see a SHIP operator. A statement may or may not be associated with the SHIP. If the operators below SHIP are being processed by the remote data source, then there will be a remote SQL statement associated with SHIP. If the operators below SHIP are being processed by the federated server, then the remote SQL statement will not be associated with SHIP. In the case of INSERT, UPDATE or DELETE operations involving a nickname, you may or may not see a SHIP. If the INSERT, UPDATE or DELETE statement is completely evaluated by the remote data source, then you will not see a SHIP in the access plan. In this case, the remote SQL statement will be found in the RETURN operator (the top operator).
- SQL Explain. Use the **db2expln** or the **dynexpln** command to start SQL Explain. Use this tool to view the access plan strategy as text. SQL explain does not provide a graphical user interface. When the query optimizer generates a remote plan, it takes into consideration different factors regarding the capability of remote data source. The optimizer expects that the remote data source will use a plan similar to the plan it generates. But the remote data source is free to choose and use any plan it sees fit. The SQL Explain tool does not display the remote part of the access plans.

Related concepts:

- “Explain tools” in the *Administration Guide: Performance*
- “SQL explain tools” in the *Administration Guide: Performance*
- “Understanding access plan optimization decisions” on page 253

Understanding access plan optimization decisions

This section lists typical optimization questions, and areas you can investigate to improve performance.

Why isn't a join between two nicknames of the same data source being evaluated remotely?

Areas to examine include:

- Join operations. Can the data source support them?
- Join predicates. Can the join predicate be evaluated at the remote data source? If the answer is no, examine the join predicate.
- Number of rows in the join result. You can determine the number of rows with Visual Explain. Does the join produce a much larger set of rows than the two nicknames combined? Do the numbers make sense? If the answer is no, consider updating the nickname statistics with the RUNSTATS utility.

Why isn't the GROUP BY operator being evaluated remotely?

Areas to examine include:

- Operator syntax. Verify that the operator can be evaluated at the remote data source.
- Number of rows. Check the estimated number of rows in the GROUP BY operator input and output using Visual Explain. Are these two numbers very close? If the answer is yes, the optimizer considers it more efficient to evaluate this GROUP BY locally. Also, do these two numbers make sense? If the answer is no, consider updating the nickname statistics using RUNSTATS.

Why is the statement not being completely evaluated remotely?

The optimizer performs cost-based optimization. Even if pushdown analysis indicates that every operator can be evaluated at the remote data source, the optimizer still relies on its cost estimate to generate a globally optimal plan. There are a great many factors that can contribute to that plan. Suppose that the remote data source can process every operation in the original query. However, its CPU speed is much slower than the CPU speed of the federated server. It may turn out to be more beneficial to perform the operations at the DB2® federated server instead. If results are not satisfactory, verify the server statistics in the SYSSTAT.SERVEROPTIONS catalog table.

Why does a plan generated by the optimizer and completely evaluated remotely, have much worse performance than the original query executed directly at the remote data source?

Areas to examine include:

- The remote SQL statement generated by the DB2 query optimizer. Ensure that it is identical to the original query. Check for predicate ordering changes. A good query optimizer should not be sensitive to the predicate ordering of a query. Unfortunately, not all DBMS optimizers are identical. It

is likely that the optimizer at the remote data source may generate a different plan based on the input predicate ordering. If this is true, this is a problem inherent in the remote optimizer. Consider either modifying the predicate ordering on the input to DB2 or contacting the service organization of the remote data source for assistance.

Also, check for predicate replacements. A good query optimizer should not be sensitive to equivalent predicate replacements. It is possible that the optimizer at the remote data source may generate a different plan based on the input predicate. For example, some optimizers cannot generate transitive closure statements for predicates.

- The number of returned rows. You can get this number from Visual Explain. If the query returns a large number of rows, network traffic is a potential bottleneck.
- Additional functions. Does the remote SQL statement contain additional functions than the original query? Some of the extra functions may be generated to convert data types. Ensure that they are necessary.

Related concepts:

- “Pushdown analysis” on page 233
- “Understanding access plan evalutation decisions” on page 242
- “Analyzing global optimization” on page 252

Related tasks:

- “Global optimization” on page 246

Chapter 15. Application programming issues for federated systems

Although developing applications for a federated system is similar to the way you develop applications for any other DB2 database, there are some unique aspects when working with nicknames. This chapter discusses issues that programmers need to consider when developing applications for federated systems. For detailed information on application programming, see:

- *IBM DB2 Universal Database Application Development Guide: Building and Running Applications Version 8*
- *IBM DB2 Universal Database Application Development Guide: Programming Client Applications Version 8*

the

How client applications interact with data sources

To client applications, the data sources in a federated system appear as a single collective database. To obtain data from data sources, applications submit queries in DB2® SQL to the federated database. DB2 then distributes the queries to the appropriate data sources, and either returns this data to the applications or performs the requested action. The federated database can join data from local tables and remote data sources, as if all the data is local. For example, you can join data that is located in a local DB2 for Windows® table, an Informix™ table, and a Sybase view in a single SQL statement. By processing SQL statements as if the data sources were ordinary relational tables or views within the federated database, the federated system can join relational data with data in non-relational formats.

In a federated system, you access data sources through nicknames. A *nickname* is an identifier that an application uses to reference a data source object, such as a table or view. To write to a data source—for example, to update a data source table—an application can use DB2 SQL (with nicknames). Alternatively, applications can use the SQL dialect of the data source (without nicknames) in a special session called *pass-through* to access the data sources directly.

Applications that use DB2 SQL and nicknames, can access any data types that DB2 recognizes, except for BIGINT.

The federated database system catalog contains information about the objects in the federated database and information about objects at the data sources. Since the catalog contains information about the entire federated system, it is called a *global catalog*.

Related concepts:

- "Using pass-through to query data sources directly" on page 277

Related tasks:

- "Referencing data source objects by nicknames in SQL statements" on page 256
- "Performing operations on data source objects" on page 257
- "Cataloging information about data source objects" on page 258
- "Invoking stored procedure nicknames" on page 260
- "Defining column options on nicknames" on page 260
- "Accessing data sources using PASSTHRU" on page 181

Working with nicknames in your applications

When you create a nickname for a data source object, DB2 interprets and interacts with that object as if it were a DB2 table. When writing applications for federated systems, you need to understand:

- How to reference data source objects by nickname
- How to perform operations on nicknamed objects
- How to supply the global catalog with information about the nicknamed objects
- How to invoke stored procedures through nicknames
- How to define column options on nicknames
- How to use views referenced by nicknames

Referencing data source objects by nicknames in SQL statements

With a federated system, you do not need to identify the data source server, schema, and object in your SQL statements. Instead, use the nicknames defined for data source objects in your SQL statements to query data source objects.

Prerequisites:

Data source objects must have nicknames registered in the federated database before you can include them in your queries.

Using nicknames in SELECT, INSERT, UPDATE, and DELETE statements:

Suppose that you define the nickname NFXDEPT to represent a table in an Informix database called NFX1.PERSON.DEPT, where:

- NFX1 is the name assigned to the server in the server definition.
- PERSON is the data source schema.
- DEPT is the data source table name.

The statement SELECT * FROM NFXDEPT is allowed from the federated server. However, the statement SELECT * FROM NFX1.PERSON.DEPT is not allowed (except in a pass-through session). The federated server does not have NFX1.PERSON.DEPT registered as a nickname.

Using nicknames in the CREATE TABLE statement:

Suppose you create a local summary table and reference a nickname in the summary_table_definition clause of the CREATE TABLE statement. You must also specify the DEFINITION ONLY keywords in this clause. For example, the CREATE TABLE statement would be:

```
CREATE TABLE table_name LIKE nickname DEFINITION ONLY
```

For a complete list of the SQL statements that you can use with a federated system, see Quick Reference - federated SQL statements.

Related concepts:

- "Using pass-through to query data sources directly" on page 277

Related tasks:

- "Pass-though considerations and restrictions" on page 278
- "Using pass-through with Oracle data sources" on page 278

Performing operations on data source objects

There are other SQL statements that may require you to use nicknames in the syntax.

Prerequisites:

Data source objects must have nicknames registered in the federated database before you can include them in your SQL statements.

Using nicknames in the COMMENT ON statement:

The COMMENT ON statement adds or replaces comments in the federated database global catalog. The COMMENT ON statement is valid against a nickname and columns that are defined on a nickname. This statement does not update data source catalogs.

Using nicknames in the GRANT and REVOKE statements:

The GRANT and REVOKE statements are valid against a nickname for certain privileges and for all users and groups. However, DB2 does not issue a corresponding GRANT or REVOKE against the object on the data source that the nickname references.

For example, suppose that user JON creates a nickname for an Oracle table that had no index. The nickname is ORAREM1. Later, the Oracle DBA defines an index for this table. User EILEEN now wants the DB2 federated database to know that this index exists, so that the query optimizer can devise strategies to access the table more efficiently. EILEEN can inform the federated database that a new index exists, by creating an index specification for ORAREM1.

The information about the index is stored in the SYSSTAT.INDEXES catalog view. Use the GRANT statement to give EILEEN the index privilege on this nickname, so that she can create the index specification.

```
GRANT INDEX ON NICKNAME ORAREM1 TO USER EILEEN
```

To revoke user EILEEN's privileges to create an index specification on nickname ORAREM1, use the REVOKE statement:

```
REVOKE INDEX ON ORAREM1 FROM USER EILEEN
```

Related concepts:

- "Create nicknames for each data source object" on page 96

Related tasks:

- "Creating index specifications for data source objects" on page 216

Related reference:

- "COMMENT statement" in the *SQL Reference, Volume 2*
- "GRANT (Table, View, or Nickname Privileges) statement" in the *SQL Reference, Volume 2*
- "REVOKE (Table, View, or Nickname Privileges) statement" in the *SQL Reference, Volume 2*

Cataloging information about data source objects

When a nickname is created for a data source object, the federated database global catalog is updated with information about that object. DB2 query optimize uses this information to plan how to retrieve data from the object. It is important to make sure that the data source information is current. The federated database does not automatically detect changes to data source objects.

Updating catalog statistics:

The information stored in the global catalog about a data source object, depends on the type of object. For database tables and views, the name of the object, the column names and attributes, are stored in the global catalog.

In the case of a table, the information also includes:

- Statistics. For example, the number of rows and the number of pages on which the rows exist. Ensure that DB2 obtains the latest statistics. Run the data source equivalent of the RUNSTATS command against the table before you create the nickname.
- Index descriptions. If the table has no indexes, you can supply the catalog with metadata that an index definition typically contains. For example, you can inform the catalog which column or columns in the table have unique values, and whether any rows are unique. You can generate this metadata, which is collectively called an *index specification*, by issuing the CREATE INDEX statement against the nickname for the table. The CREATE INDEX statement produces only an index specification; it does not create an actual index.

To determine what data source information that is stored in the global catalog, query the SYSCAT.TABLES and SYSCAT.COLUMNS catalog views . To determine what data source index information stored in the catalog, or what a particular index specification contains, query the SYSCAT.INDEXES catalog view.

Change applications to reference the SYSSTAT view instead of the SYSCAT view:

As noted in the DB2 for 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.

Related concepts:

- “Catalog statistics” in the *Administration Guide: Performance*
- “Catalog statistics tables” in the *Administration Guide: Performance*
- “The federated database” on page 7

Related tasks:

- “Creating index specifications for data source objects” on page 216
- “Nickname characteristics affecting global optimization” on page 249

Related reference:

- "SYSCAT.COLUMNS catalog view" in the *SQL Reference, Volume 1*
- "SYSCAT.INDEXES catalog view" in the *SQL Reference, Volume 1*
- "SYSCAT.TABLES catalog view" in the *SQL Reference, Volume 1*
- Appendix A, "Views in the global catalog table containing federated information" on page 281

Invoking stored procedure nicknames

If you are migrating applications from DataJoiner which invoke stored procedures through nicknames, you will have to modify your applications. DB2 for UNIX and Windows does not currently support this feature. The ability to invoke a stored procedure nickname will be added in a future release.

Defining column options on nicknames

Setting column options might improve the performance of your applications. Column options are parameters in the CREATE NICKNAME and ALTER NICKNAME statements. You can specify column options when you initially create a nickname or by modifying an existing nickname.

The information you provide through the column options is stored in the global catalog.

There are two column options you can set:

- *numeric_string*
- *varchar_no_trailing_blanks*

Numeric_string column option

By default, the query optimizer assumes that numeric string columns contain trailing blanks. If a data source numeric string column contains only numeric digits, and no other characters, including blanks, then set the *numeric_string* column option to 'Y'. This will allow queries against this column to be optimized on sorting operations and comparison operations. For example:

```
ALTER NICKNAME nickname OPTIONS (SET numeric_string 'Y')
```

Varchar_no_trailing_blanks column option

By default, the query optimizer assumes that Oracle columns using the VARCHAR data type contain trailing blanks. On this assumption, it develops an access strategy that involves modifying queries so that the values returned from these columns are the ones that the user expects. If a VARCHAR column has no trailing blanks, set the *varchar_no_trailing_blanks* column option to 'Y'. The optimizer can develop a more efficient access strategy for queries which include the VARCHAR data type.

For example:

```
ALTER NICKNAME nickname OPTIONS (SET varchar_no_trailing_blanks 'Y')
```

Related reference:

- Appendix E, “Column options for federated systems” on page 299

Creating and using federated views

A view in the federated database whose base tables are located at remote data sources is called a *federated view*. The base tables are referenced in the federated view by nicknames, instead of by the data source table names.

The advantages of using federated views are similar to the advantages of using views defined on multiple local tables in a centralized relational database manager:

- Views provide an integrated representation of the data.
- You can exclude table columns which contain confidential or sensitive data from a view.

Restrictions:

Federated views that are created from multiple nicknamed data source objects are read-only views and cannot be updated.

Federated views that are created from only one nicknamed data source object may or may not be read-only views.

- A federated view created from a single non-relational data source is read-only.
- A federated view created from a single relational data source might allow updates, depending on what is included in the CREATE FEDERATED VIEW statement.

Procedure:

You create a federated view of data source objects that have nicknames. The action of creating a federated database view of data source data is sometimes called “creating a view on a nickname”. This phrase reflects the fact that for the federated view to be created, the CREATE VIEW statement fullselect must reference the nickname of each data source table and view that the federated view is to contain.

Creating a federated view which merges similar data from several data tables:

Suppose you have customer data on three separate servers, one in Europe, one in Asia, and one in South America. The Europe customer data is in a Oracle table. The nickname for that table is ORA_EU_CUST. The Asia customer data is in a Sybase table. The nickname for that table is SYB_AS_CUST. The South America customer data resides in an Informix table. The nickname for that table is INFMX_SA_CUST. Each table has columns containing the customer number (CUST_NO), the customer name (CUST_NAME), the product number (PROD_NO), and the quantity ordered (QUANTITY). The syntax to create a view from these three nicknames that merges this customer data is:

```
CREATE FEDERATED VIEW FV1
AS SELECT * FROM ORA_EU_CUST
UNION
SELECT * FROM SYB_AS_CUST
UNION
SELECT * FROM INFMX_SA_CUST
```

Joining data to create a federated view:

Suppose you have customer data on one server and sales data on another server. The customer data is in a Oracle table. The nickname for that table is ORA_EU_CUST. The sales data is in a Sybase table. The nickname for that table is SYB_SALES. You want to match up the customer information with the purchases those customers have made. Each table has a column containing the customer number (CUST_NO). The syntax to create a view from these two nicknames that joins this data is:

```
CREATE FEDERATED VIEW FV4
AS SELECT A.CUST_NO, A.CUST_NAME, B.PROD_NO, B.QUANTITY
FROM ORA_EU_CUST A, SYB_SALES B
WHERE A.CUST_NO=B.CUST_NO
```

When the view is created, the person defining the view must have SELECT priviledge on the nicknames. Authorization requirements of the data source for the table or view referenced by the nickname are applied when the query is processed. The authorization ID of the statement may be mapped to a different remote authorization ID through a user mapping.

The presence of a nickname in the the CREATE FEDERATED VIEW statement changes the authorization model used for the view when the view is created and when the view is subsequently referenced in a query.

When a federated view is referenced in a query, the nicknames used to create the view are queried. This means that the underlying data sources are queried. The authorization ID that issues the query (or the remote authorization ID to which it maps) must have the necessary privileges to access the data source table or view.

Related tasks:

- "Accessing heterogeneous data through federated views" on page 182

Related reference:

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

Using isolation levels to maintain data integrity

An isolation level associated with an application process defines the degree of isolation of that application process from other concurrently executing application processes. The isolation level is specified as an attribute of a package that applies to a the application processes that use the package. Isolation levels are used when you prep or bind an application.

Locking occurs at the base table row. The database manager, however, can replace multiple row locks with a single table lock. This is called lock escalation. An application process is guaranteed that at least the minimum requested lock level.

You can maintain data integrity for a data source table by requesting that the table rows be locked at a specific isolation level. For example, to ensure that you have sole access to a row, you would specify the repeatable read (RR) isolation level for that row. The federated server maps the isolation level you request to a corresponding one at the data source. The isolation levels are:

CS Cursor stability

RR Repeatable read

RS Read stability

UR Uncommitted read

Procedure:

The following table lists the isolation levels that you can request on the supported data sources.

Table 36. Comparable isolation levels between the federated server and supported data sources.

DB2 federated server	CS	RR	RS	UR
Informix	Default	Transaction read-only	Transaction read-only	Same as cursor stability

Table 36. Comparable isolation levels between the federated server and supported data sources. (continued)

Oracle	Default	Transaction read-only	Transaction read-only	Same as cursor stability
Sybase	Default	Transaction read-only	Transaction read-only	Same as cursor stability
Microsoft SQL Server	Default	Transaction read-only	Transaction read-only	Same as cursor stability
ODBC	Default	Transaction read-only	Transaction read-only	Same as cursor stability
OLE DB	Default	Transaction read-only	Transaction read-only	Same as cursor stability

Related concepts:

- "Isolation levels" in the *SQL Reference, Volume 1*
- "Applications in Host or iSeries Environments" in the *Application Development Guide: Programming Client Applications*
- "How client applications interact with data sources" on page 255

Overriding the default data type mappings

When a nickname is created for a data source object, DB2® populates the global catalog with information about the table.

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. The other type of mapping is a *reverse type mapping*, which are used infrequently. In a reverse type mapping, the mapping is from a local type to a comparable remote type.

DB2 uses data type mappings to determine what DB2-supported data type should be defined for the column of a data source object. Default data type mappings are built into the data source wrappers.

Your applications might require data type mappings that are different than the default mappings. You can override the default mappings using the CREATE TYPE MAPPING statement.

Restrictions:

DB2 federated servers do not support mappings for these local data types: LONG VARCHAR, LONG VARGRAPHIC, DATALINK, and user-defined types.

Procedure:

Use the CREATE TYPE MAPPING statement to define new forward data type mappings. For example:

```
CREATE TYPE MAPPING type_mapping_name FROM SERVER remote_server_name
TO LOCAL TYPE local_data_type REMOTE TYPE data_source_data_type
```

Note: This syntax is new for the CREATE TYPE MAPPING statement in DB2 for UNIX® and Windows, Version 8.

Mappings you create are stored in the federated database global catalog SYSSTAT.TYPEMAPPING view.

See *Default forward data type mappings* for a list of the built-in mappings.

Related tasks:

- "Modifying default data type mappings" on page 208

Related reference:

- "CREATE TYPE MAPPING statement" in the *SQL Reference, Volume 2*
- Appendix H, "Default forward data type mappings" on page 307

Federated LOB support

With a federated database system, you can access and manipulate large objects (LOBs) at remote data sources. Because LOBs can be very large, transferring LOBs from a remote data source can be time-consuming. The DB2® federated database attempts to minimize transferring LOB data from the data sources, and also attempts to deliver requested LOB data directly from the data source to the requesting application without materializing the LOB at DB2.

DB2 Federated System supports select operations on LOBs at Informix, Microsoft® SQL Server, and Sybase data sources. For example:

```
SELECT empname, picture FROM nfmx_emp_table WHERE empno = '01192345'
```

where *picture* represents a LOB column and *nfmx_emp_table* represents a nickname referencing an Informix™ table containing employee data.

DB2 Federated System supports select, insert, update, and delete operations on LOBs at Oracle data sources (Version 7.3 or higher).

DB2 for UNIX® and Windows® Version 8 supports the follow read and write operations:

Table 37. Read and write support for LOBs

Data source	Type of operations
DB2 family	read only
Informix	read only
Microsoft SQL Server	read only
Oracle (NET8 wrapper)	read and write
Oracle (SQLNET wrapper)	read only
ODBC	read only
Sybase	read only

Related concepts:

- "How applications can use LOB locators" on page 266
- "Restrictions on LOBs" on page 267
- "Mappings between LOB and non-LOB data types" on page 267

Federated LOB support—details

How applications can use LOB locators

Applications can request LOB locators for LOBs that are stored in remote data sources. A *LOB locator* is a 4-byte value stored in a host variable. An application can use the LOB locator to refer to a LOB value (or LOB expression) held in the database system. Using a LOB locator, an application can manipulate the LOB value as if the LOB value was stored in a regular host variable. When you use LOB locators, there is no need to transport the LOB value from the data source server to the application (and possibly back again).

DB2® can retrieve LOBs from remote data sources, store them at the federated server, and then issue a LOB locator against the stored LOB. LOB locators are released when:

- Applications issue ″FREE LOCATOR″ SQL statements.
- Applications issue COMMIT statements.
- The DB2 federated instance is restarted.

Related concepts:

- “Large Object Locators” in the *Application Development Guide: Programming Server Applications*
- “Federated LOB support” on page 265
- “Restrictions on LOBs” on page 267
- “Mappings between LOB and non-LOB data types” on page 267

Restrictions on LOBs

The federated database is unable to bind remote LOBs to a file reference variable.

LOBs are not supported in pass-through sessions.

Related concepts:

- “Federated LOB support” on page 265
- “How applications can use LOB locators” on page 266
- “Mappings between LOB and non-LOB data types” on page 267

Mappings between LOB and non-LOB data types

There are a few situations in which you need to map a DB2® LOB data type to a non-LOB data type at a data source.

Procedure:

When you need to create a mapping between a column at a data source and its DB2 LOB type counterpart, use the CREATE TYPE MAPPING statement. For example suppose that you need to create a mapping for the Oracle LONG data type to the DB2 CLOB data type, for every Oracle data source object identified in your federated database. You would create a type mapping such as:

```
CREATE TYPE MAPPING my_oracle_lob FROM sysibm.clob TO SERVER
TYPE oracle TYPE LONG
```

where:

my_oracle_lob
: Is the name of the type mapping.

sysibm.clob
: Is the DB2 CLOB data type you are mapping to.

SERVER TYPE *oracle*
Is the type of server you are connecting to.

TYPE *LONG*
Is the Oracle data type counterpart.

Note: This syntax is new for the CREATE TYPE MAPPING statement in DB2 for UNIX® and Windows, Version 8.

Mappings you create are stored in the federated database global catalog SYSSTAT.TYPEMAPPING view.

Related concepts:

- "Federated LOB support" on page 265
- "How applications can use LOB locators" on page 266
- "Restrictions on LOBs" on page 267

Related reference:

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

Using distributed requests to query data sources

Queries submitted to the federated database can request results from a single data source; but typically they request results that are from multiple data sources. Because a typical query is distributed to multiple data sources, it is called a *distributed request*. In general, a distributed request uses one or more of three SQL conventions to specify where data is to be retrieved from: subqueries, set operators, and join subselects.

Suppose that you have a federated server configured to access a DB2 for OS/390 data source, a DB2 for iSeries data source, and an Oracle data source. Stored in each data source is a table that contains employee information. The federated server references these tables by nicknames that point to where the tables reside.

UDB390_EMPLOYEES
Nickname for a table on a DB2 for OS/390 data source that contains employee information.

iSERIES_EMPLOYEES
Nickname for a table on a DB2 for iSeries data source that contains employee information.

ORA_EMPLOYEES
Nickname for a table on an Oracle data source that contains employee information.

ORA_REGIONS
: Nickname for a table on an Oracle data source that contains information about the regions that the employees live in.

The following examples illustrate the three SQL conventions used with distributed requests, using the nicknames defined for each of the tables.

Example: A distributed request with a subquery:

iSERIES_EMPLOYEES contains the phone numbers of employees who live in Asia. It also contains the region codes associated with these phone numbers, but it does not list the regions that the codes represent. ORA_REGIONS lists both codes and regions. The following query uses a subquery to find the region code for China. Then it uses the region code to return a list of those employees in iSERIES_EMPLOYEES who have a phone number in China.

```
SELECT name, telephone FROM db2admin.iSERIES_employees
WHERE region_code IN (SELECT region_code FROM dbadmin.ora_regions
WHERE region_name = 'CHINA')
```

Example: A distributed request with set operators:

The federated server supports three set operators: UNION, EXCEPT, and INTERSECT.

- Use the UNION set operator to combine the rows that satisfy any of two or more SELECT statements.
- Use the EXCEPT set operator to retrieve those rows that satisfy the first SELECT statement but not the second.
- Use the INTERSECT set operator to retrieve those rows that satisfy both SELECT statements.

All three set operators can use the ALL operand to indicate that duplicate rows are not be removed from the result. This eliminates the need for an extra sort.

The following query retrieves all employee names and region codes that are present in both iSERIES_EMPLOYEES and UDB390_EMPLOYEES, even though each table resides in a different data source.

```
SELECT name, region_code
   FROM as400_employees
INTERSECT
SELECT name, region_code
   FROM udb390_employees
```

Example: A distributed request for a join:

A relational join produces a result set that contains a combination of columns retrieved from two or more tables. You should specify conditions to limit the size of the rows in the result set.

The query below combines employee names and their corresponding region names by comparing the region codes listed in two tables. Each table resides in a different data source.

```
SELECT t1.name, t2.region_name
FROM dbadmin.iSERIES_employees.t1, dbadmin.ora_regions.t2
WHERE t1.region_code = t2.region_code
```

Using server options to optimize distributed requests

In a federated system, use parameters called *server options* to supply the global catalog with information that applies to a data source as a whole, or to control how DB2 interacts with a data source. For example, you can:

- Catalog the instance identifier by assigning the identifier as a value to the **node** server option.
- Use the **varchar_no_trailing_blanks** server option to inform the optimizer that every VARCHAR column residing on the data source server is free of trailing blanks.
- Set the **plan_hints** server option to a value that enables DB2 to provide Oracle data sources with statement fragments, called *plan hints*, that help Oracle optimizers do their job. Specifically, plan hints can help an optimizer to decide which index to use in accessing a table, and which table join sequence to use in retrieving data for a result set.

Typically, the database administrator sets server options for a federated system. However, a programmer can make good use of the options that help to optimize queries. For example, suppose that for data sources ORACLE1 and ORACLE2, the plan_hints server option is set to its default, ‘n’ (no, do not furnish this data source with plan hints). Also suppose that you write a distributed request for data from ORACLE1 and ORACLE2. You expect that plan hints would help the optimizers at these data sources improve their strategies for accessing this data. You could override the default with a setting of ‘y’ (yes, furnish the plan hints) while your application is connected to the federated database. When the connection is terminated, the setting would automatically revert to ‘n’.

Use the SET SERVER OPTION statement to set or change server options. To ensure that the setting takes effect, specify the SET SERVER OPTION statement immediately following the CONNECT statement. The server option is set for the duration of a connection to the federated database. In addition, it is advisable to prepare the statement dynamically.

Related reference:

- "SET SERVER OPTION statement" in the *SQL Reference, Volume 2*
- Appendix C, "Server options for federated systems" on page 287

Invoking user-defined functions in applications

Application developers often need to create their own suite of functions specific to their application or domain. They can use user-defined scalar functions for this purpose.

For example, a retail store could define a PRICE data type for tracking the cost of items that it sells. This store might also want to define a SALES_TAX function. This function would take a given price value as input, compute the applicable sales tax, and return this data to the requesting user or application.

These functions can operate over all database types, including large object types and distinct types. UDFs allow queries to contain powerful computation and search predicates to filter irrelevant data close to the source of the data, thereby reducing response time. The SQL optimizer treats UDFs exactly like built-in functions such as SUBSTR and LENGTH. Applications can be developed using different application language environments, such as C, C++, COBOL, and FORTRAN, while sharing a set of SQL UDFs.

UDFs can not only manipulate data but also perform actions. For example, you might enable a UDF to send an electronic message or to update a flat file.

In DB2, UDFs can include:

- Functions that you define from scratch.
- Functions in the SYSFUN schema. Examples include mathematical functions such as SIN, COS, and TAN; scientific functions such as RADIANS, LOG10, and POWER; and general purpose functions such as LEFT, DIFFERENCE, and UCASE.

Related concepts:

- "User-defined functions" in the *SQL Reference, Volume 1*

Invoking user-defined functions in applications—details

Enabling the federated database to access functions at data source

You can use a federated database in connection with UDFs when:

- You want to directly invoke an UDF at a data source. You can do this in a pass-through session.

- You want a federated database to access a *remote function*—a UDF at a data source or a built-in function that resides at a data source.

Before you can use a federated database to invoke a remote function, federated database must associate the function with a function specification stored in the global catalog. The signature in this specification must correspond to the signature of the function that you want to invoke. A *signature* is the combination of a function's name and input parameters. Signatures *correspond* if:

- They contain the same names and the same number of parameters, and
- If the data type of each parameter in one signature is the same as (or can be converted to) the data type of the corresponding parameter in the other signature.

There are two conditions under which the federated database can associate a function specification at its database with a remote function:

- If the federated database contains a function whose signature corresponds to that of the signature of the remote function.
- If the federated database doesn't contain a function with the requisite signature, you can define to the database a function template that contains this signature. Then you map the function template to the function that you want to invoke.

To define a function template to the federated database, use the CREATE FUNCTION statement. To map a function or a function template at the federated database to a remote function, use the CREATE FUNCTION MAPPING statement. The settings for function mappings are stored in the SYSCAT.FUNCMAPPINGS catalog view.

DB2 can invoke a data source function that it does not recognize, if you create a mapping between the data source function and a DB2 function that is stored in the federated database. Examples of functions that DB2 does not recognize are:

- A user-defined function.
- A new built-in function that was added to the data source after the nickname was created for the data source.

.

The DB2 function can be an existing function or function template, or a new function or function template that you create. To create a mapping between functions, the data source function and the DB2 function it is mapped to should have:

- the same number of input parameters

- compatible data types

Suppose you want to map a user-defined function named US_DOLLAR at an Oracle data source called ORACLE2, to a DB2 user-defined function that you create. You decide to name the DB2 user-defined function DOLLAR and to name this function mapping ORACLE_DOLLAR. The two SQL statements would be:

```
CREATE FUNCTION DOLLAR () RETURNS DECIMAL(9,2)
CREATE FUNCTION MAPPING ORACLE_DOLLAR FOR DOLLAR SERVER ORACLE2
OPTIONS (REMOTE_NAME 'US_DOLLAR')
```

After the mapping is created, you submit distributed requests that reference the DB2 function. For example, if you mapped a DB2 user-defined function called DOLLAR to an Oracle user-defined function called US_DOLLAR, your request would specify DOLLAR rather than US_DOLLAR.

Related concepts:

- "Function mappings and function templates" on page 20
- "Invoking user-defined functions in applications" on page 271

Related tasks:

- "Creating and modifying function mappings" on page 223
- "Specifying function overhead through mapping options" on page 273
- "Specifying function names in a function mapping" on page 275
- "Discontinuing function mappings" on page 275

Related reference:

- "CREATE FUNCTION statement" in the *SQL Reference, Volume 2*
- "CREATE FUNCTION MAPPING statement" in the *SQL Reference, Volume 2*
- "CREATE FUNCTION (Sourced or Template) statement" in the *SQL Reference, Volume 2*
- Appendix F, "Function mapping options for federated systems" on page 301

Specifying function overhead through mapping options

When you create a function mapping, you can provide DB2 with important information about the potential cost, or overhead, of executing a data source function at the data source. This information helps the DB2 query optimizer determine the best strategy for executing the query. When a distributed request is processed, the optimizer evaluates multiple access strategies and estimates the overhead to invoke the DB2 function and the data source function. Which ever strategy that is expected to cost the least amount of overhead is the one that is used.

You include estimated statistics on the overhead that would be consumed when the data source function is invoked in the CREATE FUNCTION MAPPING statement . For example, the statement can specify the estimated number of instructions that would be required to invoke the data source function. It can specify the estimated number of I/Os that would be expended for each byte of the argument set that is passed to this function. These estimates are stored in the global catalog, and appear in the SYSCAT.FUNCMAPOPTIONS view. When a DB2 function is used in the mapping — instead of a data source function or a DB2 function template — the global catalog contains estimates of overhead that would be consumed when the DB2 function is invoked. You can see these estimates in the SYSCAT.FUNCTIONS view.

To specify estimated statistics in the CREATE FUNCTION MAPPING statement, use the function mapping options. describes the options you can set for data source functions and their default values.

Table 38. Function mapping options and their settings

Option	Valid settings	Default setting
disable	Disable a default mapping. Valid values are 'Y' and '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'
insts_per_argbyte	Estimated number of instructions processed for each byte of the argument set that is passed to the data source function.	'0'
insts_per_invoc	Estimated number of instructions processed per invocation of the data source function.	'450'
ios_per_argbyte	Estimated number of I/Os expended for each byte of the argument set that is passed to the data source function.	'0'
ios_per_invoc	Estimated number of I/Os per invocation of a data source function.	'0'
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

If the estimates of consumed overhead change, you can record the change in the global catalog. To record new estimates for the data source function, first drop or disable the function mapping. Then recreate the mapping with the CREATE FUNCTION MAPPING statement, specifying the new estimates in the statement. When you run the statement, the new estimates will be added to the SYSCAT.FUNCTIONS catalog view. To record changed estimates for the DB2 function, update the SYSSTAT.FUNCTIONS catalog view directly.

Specifying function names in a function mapping

How you code the CREATE FUNCTION MAPPING statement depends on part on whether the names of the objects that you are mapping together are the same or different. If you are creating a mapping between two functions (or a function template and a function) that have the same name, you must assign this name to the *function_name* parameter.

But if the names differ, then:

- Assign the name of the federated database function or function template to the *function_name* parameter.
- Specify a function mapping option called “remote_name” and assign the name of the data source function to this option. The name must have fewer than 255 characters.

Suppose you want to map a user-defined function named UPPERCASE at an Oracle data source called ORACLE2 to the DB2 function UCASE(CHAR). You also want to include the estimated number of instructions per invocation of the UPPERCASE function. You decide to name this function mapping ORACLE_UPPER. The syntax would be:

```
CREATE FUNCTION MAPPING ORACLE_UPPER FOR SYSFUN.UCASE(CHAR)
SERVER ORACLE2 OPTIONS
(REMOTE_NAME 'UPPERCASE', INSTS_PER_INVOC '1000')
```

Discontinuing function mappings

If you want to discontinue using a function mapping, follow these guidelines:

- User-defined function mappings are listed in the SYSCAT.FUNCMAPPINGS catalog view. To discontinue a user-defined function mapping, you have to delete the mapping. You do this with the DROP FUNCTION MAPPING statement.
- Default mappings are not listed in the SYSCAT.FUNCMAPPINGS catalog view. To discontinue a default function mapping, you disable the mapping. You do this in the CREATE FUNCTION MAPPING statement by setting the “disable” option to ‘y’ (yes, disable this function mapping). The default is ‘n’.

Related concepts:

- "Define alternative function mappings to the federated database" on page 103

Related tasks:

- "Creating and modifying function mappings" on page 223
- "Enabling the federated database to access functions at data source" on page 271
- "Specifying function overhead through mapping options" on page 273
- "Specifying function names in a function mapping" on page 275

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*
- "CREATE FUNCTION MAPPING statement" in the *SQL Reference, Volume 2*
- Appendix A, "Views in the global catalog table containing federated information" on page 281

Enabling the federated database to recognize data source user-defined data types (UDTs)

A UDT is a distinct user-defined data type that shares its internal representation with an existing type, but is considered to be a separate and incompatible type for semantic purposes. For example, a user might want to define a PICTURE type, a TEXT type, and an AUDIO type, all of which have quite different semantics, but which all use the predefined data type binary large object (BLOB) for their internal representation.

One of the benefits of UDTs is strong typing. Strong typing guarantees that only functions and operations defined on the distinct type can be applied to the type. For example, the system does not allow you to directly compare a PICTURE type with an AUDIO type even though they share the same underlying type. If you did want to make such a comparison, you need to first convert values of one type to values of the other.

User-defined types, like built-in types, can be used for columns of tables as well as parameters of functions. For example, a user can define a data type such as ANGLE (which varies between 1 and 360) and a set of UDFs to act on it, such as SINE, COSINE and TANGENT.

In some cases, the definition of a table, view, or function at a data source might include a UDT that DB2 does not recognize. So that DB2 can recognize the UDT (and consequently access the table, view, or function), you must map the UDT to a corresponding one at the federated database. If the federated

database does not contain a corresponding UDT, you can create one with the DB2 CREATE DISTINCT TYPE statement. To create the mapping, use the DB2 CREATE TYPE MAPPING statement.

Using pass-though sessions within applications

Pass-through sessions let applications communicate directly with a server using the server's native client access method and native SQL dialect.

Using pass-through to query data sources directly

Pass-through sessions are useful when:

- Applications must create objects at the data source or perform INSERT, UPDATE, or DELETE operations
- DB2® does not support a unique data source operation

When referencing objects in a pass-through session, use the true name of the object, not the nickname.

Use the SET PASSTHRU statement to start a pass-through session and access a server directly. This statement must be issued dynamically. An example of this statement is:

```
SET PASSTHRU ORACLE1
```

which opens a pass-through session to the data source ORACLE1.

If a static statement is submitted in a pass-through session, it is sent to the federated server for processing. If you want to submit an SQL statement to a data source for processing, you must prepare it dynamically in the pass-through session and have it executed while the session is still open.

- To submit a SELECT statement, use the PREPARE statement with it, and then use the OPEN, FETCH, and CLOSE statements to access the results of your query.
- For a supported statement other than SELECT, you have two options. You can use the PREPARE statement to prepare the supported statement, and then the EXECUTE statement to execute it. Alternatively, you can use the EXECUTE IMMEDIATE statement to prepare and execute the statement.

If you issue the COMMIT or ROLLBACK command during a pass-through session, this command will complete the current unit of work (UOW), but does not end the pass-through session.

Pass-though considerations and restrictions

There are a number of considerations and restrictions to keep in mind when you use pass-through. Some of them are of a general nature; others apply only to specific data sources. The following information applies to all data sources:

- Statements prepared within a pass-through session must be executed within the same pass-through session. Statements prepared within a pass-through session, but executed outside of the same pass-through session, will fail and result in a SQLSTATE 56098 error.
- An application can have several SET PASSTHRU statements to different data sources. Although the application might issue multiple SET PASSTHRU statements, only the last session is active. You cannot pass through to more than one data source at a time. When a new SET PASSTHRU statement is invoked, it terminates the previous SET PASSTHRU statement.
- If multiple pass-through sessions are used in an application, be sure to issue a COMMIT before you open another pass through session. This will conclude the unit of work for the current session.
- Host variables defined in SQL statements within a pass-through session must take the form :H*n* where H is uppercase and *n* is a unique whole number. The values of *n* must be numbered consecutively beginning with zero.
- Pass-through does not support stored procedure calls.
- Pass-through does not support the SELECT INTO statement.

Related concepts:

- "Pass-through sessions" on page 11
- "Using pass-through to query data sources directly" on page 277

Related tasks:

- "Using pass-through with Oracle data sources" on page 278
- "Accessing data sources using PASSTHRU" on page 181

Related reference:

- "SET PASSTHRU statement" in the *SQL Reference, Volume 2*

Using pass-through with Oracle data sources

When a remote client issues a SELECT statement from a command line processor (CLP) in pass-through mode and the client code is a SDK prior to DB2 Universal Database Version 5, the SELECT will elicit an SQLCODE -30090 with reason code 11. To avoid this error, remote clients must use an SDK that is at Version 5 or greater.

Any DDL statement issued against an Oracle server is performed at parse time and is not subject to transaction semantics. The operation, when complete, is automatically committed by Oracle. If a rollback occurs, the DDL is not rolled back.

When you issue a SELECT statement from raw data types, use the RAWTOHEX function to receive the hexadecimal values. When you perform an INSERT into raw data types, provide the hexadecimal representation.

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 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 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 39. 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.FUNCTIONS	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 39. 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 40. 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 40. 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. Wrapper options for federated systems

Wrapper options are used to configure the wrapper or to define how DB2 uses the wrapper. Currently, there is only one wrapper option, DB2_FENCED. The DB2_FENCED wrapper option indicates if the wrapper is fenced or trusted by DB2. A fenced wrapper operates under some restrictions.

If you did not explicitly set the DB2_FENCED wrapper option to 'N', you can alter the wrapper to include this option. If you have scripts or applications that you use for DDL statements, consider using this option. Even though the current default setting for DB2_FENCED is 'N', it is possible that IBM will change the default setting in the future. When the default changes, any wrappers created without this option will adhere to the new default. If you explicitly set the DB2_FENCED wrapper to 'N', you can ensure that the behavior of the wrapper will not change when you run the scripts or applications.

Table 41. Wrapper options and their settings

Option	Valid settings	Default setting
DB2_FENCED	Indicates if the wrapper is fenced or trusted by DB2. 'N' The tasks performed by the wrapper are not restricted.	'N'

Related concepts:

- "Create the wrapper" on page 88
- "Wrappers and wrapper modules" on page 12
- "Modifying wrappers" on page 195

Appendix C. 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 data source specific, and are noted in the following table. Life Sciences data sources have additional, very specific server options.

The common federated server options are:

- Compatibility options. COLLATING_SEQUENCE, IGNORE_UDT
- Data integrity options. IUD_APP_SVPT_ENFORCE
- Location options. CONNECTSTRING, DBNAME, IFILE
- Security options. FOLD_ID, FOLD_PW, PASSWORD
- Performance options. COMM_RATE, CPU_RATIO, IO_RATIO, LOGIN_TIMEOUT, PACKET_SIZE, PLAN_HINTS, PUSHDOWN, TIMEOUT, VARCHAR_NO_TRAILING_BLANKS

Table 42. Server options and their settings

Option	Valid settings	Default setting	Applies to
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 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'	DB2 for iSeries DB2 for z/OS and OS/390 DB2 for UNIX and Windows Informix, MS SQL Server, ODBC, Oracle, Sybase

Table 42. Server options and their settings (continued)

Option	Valid settings	Default setting	Applies to
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 2147483648. Values may be expressed as whole numbers only, for example 12.	'2'	DB2 for iSeries DB2 for z/OS and OS/390 DB2 for UNIX and Windows Informix, MS SQL Server, ODBC, Oracle, Sybase
CONNECTSTRING	Specifies initialization properties needed to connect to an OLE DB provider.	None	OLE DB
CPU_RATIO	Indicates how much faster or slower a data source's CPU runs than the federated server's CPU. 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.	'1.0'	DB2 for iSeries DB2 for z/OS and OS/390 DB2 for UNIX and Windows Informix, MS SQL Server, ODBC, Oracle, Sybase

Table 42. Server options and their settings (continued)

Option	Valid settings	Default setting	Applies to
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.	DB2 for iSeries DB2 for z/OS and OS/390 DB2 for UNIX and Windows Informix, MS SQL Server, ODBC, Sybase
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.	DB2 for iSeries DB2 for z/OS and OS/390 DB2 for UNIX and Windows Informix, MS SQL Server, ODBC, Oracle, Sybase

Table 42. Server options and their settings (continued)

Option	Valid settings	Default setting	Applies to
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.	DB2 for iSeries DB2 for z/OS and OS/390 DB2 for UNIX and Windows Informix, MS SQL Server, ODBC, Oracle, Sybase
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.	Sybase

Table 42. Server options and their settings (continued)

Option	Valid settings	Default setting	Applies to
IGNORE_UDT	Specifies whether the federated server should determine the built-in type that underlies a UDT without strong typing. Applies only to data sources accessed through the CTLIB and DBLIB protocols. Valid values are: 'Y' Ignore the fact that UDTs are user-defined and determine what built-in types under lie them. 'N' Do not ignore user-defined specifications of UDTs. When DB2 creates nicknames, it looks for and catalogs information about the objects (tables, views, stored procedures) that the nicknames point to. As it looks for the information, it might find that some objects have data types that it doesn't recognize (that is, data types that don't map to counterparts at the federated database). Such unrecognizable types can include: • New built-in types • UDTs with strong typing • UDTs without strong typing. These are built-in types that the user has simply renamed. These types are supported only by certain data sources, such as Sybase and Microsoft SQL Server. When the federated server data types that it doesn't recognize, it returns the error message, SQL3324N. However, it can make an exception to this practice. For data sources accessible through the CTLIB or DBLIB protocols, you can set the IGNORE_UDT server option so that when the federated database encounters an unrecognizable UDT without strong typing, the federated database determines what the UDT's underlying built-in type is. Then, if the federated database recognizes this built-in type, the federated database returns information about the built-in type to the catalog. To have the federated database determine the underlying built-in types of UDTs that do not have strong typing, set IGNORE_UDT to 'Y'.	'N'	Sybase

Table 42. Server options and their settings (continued)

Option	Valid settings	Default setting	Applies to
IO_RATIO	Denotes how much faster or slower a data source's I/O system runs than the federated server's 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.	'1.0'	DB2 for iSeries DB2 for z/OS and OS/390 DB2 for UNIX and Windows Informix, MS SQL Server, ODBC, Oracle, Sybase
IUD_APP_SVPT_ENFORCE	Specifies whether DB2 federated system should enforce detecting or building of application savepoint statements. 'Y' The federated server will not allow INSERT, UPDATE, or DELETE statements on nicknames if the data source does not support application savepoint statements. A SQL error code (SQL20190) will be generated when DB2 cannot perform atomic INSERT, UPDATE, or DELETE. 'N' The federated server will allow INSERT, UPDATE, or DELETE statements on nicknames.	'Y'	DB2 for iSeries DB2 for z/OS and OS/390 DB2 for UNIX and Windows Informix, MS SQL Server, ODBC, Oracle, Sybase
LOGIN_TIMOUT	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'	Sybase
NODE	Name by which a data source is defined as an instance to its RDBMS.	None.	Informix, MS SQL Server, Oracle, Sybase

Table 42. Server options and their settings (continued)

Option	Valid settings	Default setting	Applies to
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.		Sybase
PASSWORD	Specifies whether passwords are sent to a data source. 'Y' Passwords are always sent to the data source and validated. This is the default value. 'N' Passwords are not sent to the data source (regardless of any user mappings) and not validated. 'ENCRYPTION' Passwords are always sent to the data source in encrypted form and validated. Valid only for DB2 family data sources that support encrypted passwords.	'Y'	DB2 for iSeries DB2 for z/OS and OS/390 DB2 for UNIX and Windows Informix, MS SQL Server, ODBC, Oracle, Sybase
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.	'N'	Informix, MS SQL Server, ODBC, Oracle, Sybase

Table 42. Server options and their settings (continued)

Option	Valid settings		Default setting	Applies to
PUSHDOWN	'Y'	DB2 will consider letting the data source evaluate operations.	'Y'	DB2 for iSeries DB2 for z/OS and OS/390 DB2 for UNIX and Windows Informix, MS SQL Server, ODBC, Oracle, Sybase
	'N'	DB2 will only retrieve columns from the remote data source and will not let the data source evaluate other operations, such as joins.		
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'	Sybase
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, have non-blank-padded comparison semantics that return the same results as the DB2 for 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.		'N'	DB2 for iSeries DB2 for z/OS and OS/390 DB2 for UNIX and Windows Informix, MS SQL Server, ODBC, Oracle, Sybase
	'Y'	This data source has non-blank-padded comparison semantics similar to the federated server.		
	'N'	This data source has different varying-length character comparison semantics than the federated server.		

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).

Related concepts:

- "Server definitions and server options" on page 14
- "Server characteristics affecting pushdown opportunities" on page 235
- "Server characteristics affecting global optimization" on page 247

Related tasks:

- "Registering the server for table-structured files" in the *DB2 Life Sciences Data Connect Planning, Installation, and Configuration Guide*
- "Registering the server for Documentum data sources" in the *DB2 Life Sciences Data Connect Planning, Installation, and Configuration Guide*
- "Registering the server for an Excel data source" in the *DB2 Life Sciences Data Connect Planning, Installation, and Configuration Guide*
- "Registering the server for a BLAST data source" in the *DB2 Life Sciences Data Connect Planning, Installation, and Configuration Guide*
- "Registering the server for an XML data source" in the *DB2 Life Sciences Data Connect Planning, Installation, and Configuration Guide*

Related reference:

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

Appendix D. User options for federated systems

User 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 43. User 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. If this option is not specified, the ID used to connect to database is used.	None
REMOTE_DOMAIN	Indicates the Windows NT domain used to authenticate users connecting to this data source. Valid settings include any valid Windows NT domain name. If this option is not specified, the 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, the password used to connect to the database is used.	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 E. 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 upper- or lowercase.

Note: The Life Sciences Data Connect wrappers allow additional column options.

Table 44. 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 44. 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 lenght with trailing blanks. 'Y' Yes, trailing blanks are absent from this VARCHAR column. 'N' No, trailing blanks are present in this VARCHAR column. Some data sources, such as Oracle, have non-blank-padded comparison semantics that return the same results as the DB2 for UNIX and Windows comparison semantics. Set this option when you want it to apply only to a specifc VARCHAR or VARCHAR2 column in a data source object.	'N'

Related concepts:

- "Fast track to configuring your data sources" on page 85
- "Column options" on page 17
- "Pushdown analysis" on page 233

Related tasks:

- "Global optimization" on page 246

Appendix F. 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 45. 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 45. 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 G. 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 46. IBM DB2 for UNIX and Windows

Server Type	Data Source
DB2/UDB	IBM DB2 Universal Database
DATAJOINER	IBM DB2 DataJoiner V2.1 and V2.1.1

Table 46. IBM DB2 for UNIX and Windows (continued)

Server Type	Data Source
DB2/6000	IBM DB2 for AIX
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 47. IBM DB2 for iSeries (and AS/400)

Server Type	Data Source
DB2/400	IBM DB2 for iSeries and AS/400

Table 48. IBM 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 49. IBM 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 3.0 driver.

Server Type	Data Source
ODBCSERVER	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

Appendix H. Default forward data type mappings

When a nickname is created for a data source object, DB2 for 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 UNIX and Windows data type. The DB2 for 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 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 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 UNIX and Windows federated servers do not support mappings for these data types: LONG VARCHAR, LONG VARGRAPHIC, DATALINK, and 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 UNIX and Windows data types and data source data types.

These mappings are valid with all the supported versions, unless otherwise noted.

Note: For all default forward data types mapping from a data source to DB2 for UNIX and Windows, the DB2 federated schema is SYSIBM.

DB2 for z/OS and OS/390 data sources

Table 50. 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
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
DATE	-	-	-	-	-	-	DATE	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
CLOB	-	-	-	-	-	-	CLOB	-	-	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-

DB2 for iSeries data sources

Table 51. 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
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
NUMERIC	-	-	-	-	-	-	DECIMAL	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
DATE	-	-	-	-	-	-	DATE	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
CLOB	-	-	-	-	-	-	CLOB	-	-	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-

DB2 Server for VM and VSE data sources

Table 52. 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
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPH	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
DATE	-	-	-	-	-	-	DATE	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
CLOB	-	-	-	-	-	-	CLOB	-	-	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DBAHW	-	-	-	-	-	-	SMALLINT	-	0	-
DBAINT	-	-	-	-	-	-	INTEGER	-	0	-

DB2 for UNIX and Windows data sources

Table 53. DB2 for 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
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
BIGINT	-	-	-	-	-	-	BIGINT	-	0	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
DOUBLE	-	-	-	-	-	-	DOUBLE	-	-	-
CHAR	-	-	-	-	-	-	CHAR	-	0	N
VARCHAR	-	-	-	-	-	-	VARCHAR	-	0	N
CHAR	-	-	-	-	Y	-	CHAR	-	0	Y
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	0	Y
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	0	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	0	N
DATE	-	-	-	-	-	-	DATE	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
CLOB	-	-	-	-	-	-	CLOB	-	-	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-

Informix data sources

Table 54. 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	-	-	-	-	-	-	SMALLINT	2	-	-
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	-	-	-	-	-	-	DECIMAL	19	5	-
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 54. 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	19	0	
SMALLFLOAT	-	-	-	-	-	-	REAL	4	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
TEXT	-	-	-	-	-	-	CLOB	2147483647	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-

Oracle SQLNET data sources

Table 55. 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
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
FLOAT	1	63	0	0	-	\0	REAL	0	0	N

Table 55. Oracle SQLNET 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
FLOAT	64	126	0	0	-	\0	DOUBLE	0	0	N
CHAR	1	254	0	0	-	\0	CHAR	0	0	N
CHAR	255	32672	0	0	-	\0	VARCHAR	0	0	N
VARCHAR2	1	32672	0	0	-	\0	VARCHAR	0	0	N
RAW	1	254	0	0	-	\0	CHAR	0	0	Y
RAW	255	32672	0	0	-	\0	VARCHAR	0	0	Y
LONG	0	0	0	0	-	\0	CLOB	2147483647	0	N
LONG RAW	0	0	0	0	-	\0	BLOB	2147483647	0	Y
DATE	0	0	0	0	-	\0	TIMESTAMP	0	0	N
MLSLABEL	0	0	0	0	-	\0	VARCHAR	255	0	N
ROWID	0	0	0	NULL	-	\0	CHAR	18	0	N

Oracle NET8 data sources

Table 56. 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
NUMBER	1	38	-84	127	-	\0	DOUBLE	0	0	N

Table 56. Oracle NET8 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
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
FLOAT	1	63	0	0	-	\0	REAL	0	0	N
FLOAT	64	126	0	0	-	\0	DOUBLE	0	0	N
CHAR	1	254	0	0	-	\0	CHAR	0	0	N
CHAR	255	32672	0	0	-	\0	VARCHAR	0	0	N
VARCHAR2	1	32672	0	0	-	\0	VARCHAR	0	0	N
RAW	1	254	0	0	-	\0	CHAR	0	0	Y
RAW	255	32672	0	0	-	\0	VARCHAR	0	0	Y
CLOB	0	0	0	0	-	\0	CLOB	2147483647	0	N
BLOB	0	0	0	0	-	\0	BLOB	2147483647	0	Y
DATE	0	0	0	0	-	\0	TIMESTAMP	0	0	N
MLSLABEL	0	0	0	0	-	\0	VARCHAR	255	0	N
ROWID	0	0	0	NULL	-	\0	CHAR	18	0	N

Microsoft SQL Server data sources

Table 57. 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
int	-	-	-	-	-	-	INTEGER	4	-	-
intn	-	-	-	-	-	-	INTEGER	4	-	-
smallint	-	-	-	-	-	-	SMALLINT	2	-	-
tinyint	-	-	-	-	-	-	SMALLINT	2	-	-
bit	-	-	-	-	-	-	SMALLINT	2	-	-
float	-	8	-	-	-	-	DOUBLE	8	-	-
floatn	-	8	-	-	-	-	DOUBLE	8	-	-
float	-	4	-	-	-	-	REAL	4	-	-
floatn	-	4	-	-	-	-	REAL	4	-	-
real	-	-	-	-	-	-	REAL	4	-	-
money	-	-	-	-	-	-	DECIMAL	19	4	-
moneyn	-	-	-	-	-	-	DECIMAL	19	4	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
smallmoneyn	-	-	-	-	-	-	DECIMAL	10	4	-
decimal	1	31	0	31	-	-	DECIMAL	-	-	-
decimal	32	38	0	38	-	-	DOUBLE	-	-	-
decimaln	1	31	0	31	-	-	DECIMAL	-	-	-
decimaln	32	38	0	38	-	-	DOUBLE	-	-	-
numeric	1	31	0	31	-	-	DECIMAL	-	-	-
numeric	32	38	0	38	-	-	DOUBLE	8	-	-
numericn	1	31	0	31	-	-	DECIMAL	-	-	-
numericn	32	38	0	38	-	-	DOUBLE	-	-	-
char	1	254	-	-	-	-	CHAR	-	-	N

Table 57. 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
sysname	1	254	-	-	-	-	CHAR	-	-	N
char	255	8000	-	-	-	-	VARCHAR	-	-	N
varchar	1	8000	-	-	-	-	VARCHAR	-	-	N
text	-	-	-	-	-	-	CLOB	-	-	N
nchar	1	127	-	-	-	-	GRAPHIC	-	-	N
nchar	128	4000	-	-	-	-	VARGRAPHIC	-	-	N
nvarchar	1	4000	-	-	-	-	VARGRAPHIC	-	-	N
binary	1	254	-	-	-	-	CHARACTER	-	-	Y
binary	255	8000	-	-	-	-	VARCHAR	-	-	Y
varbinary	1	8000	-	-	-	-	VARCHAR	-	-	Y
image	-	-	-	-	-	-	BLOB	2147483647	-	Y
datetime	-	-	-	-	-	-	TIMESTAMP	10	-	-
datetimen	-	-	-	-	-	-	TIMESTAMP	10	-	-
smalldatetime	-	-	-	-	-	-	TIMESTAMP	10	-	-
timestamp	-	-	-	-	-	-	VARCHAR	8		Y
sysname	-	-	-	-	-	-	VARCHAR	30		Y
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	32	0	31	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-

Table 57. 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_REAL	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	8000	-	-	-	-	VARCHAR	-	-	N
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	8000	-	-	-	-	VARCHAR	-	-	N
SQL_VARBINARY	1	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N
SQL_LONGVARBINARY	-	-	-	-	-	-	BLOB	-	-	Y
SQL_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	-	-
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_BIGINT	-	-	-	-	-	-	DECIMAL	-	-	-
DUMMY65 [1]	1	38	-84	127	-	-	DOUBLE	-	-	-
uniqueidentifier [2]	1	4000	-	-	Y	-	VARCHAR	16	-	Y
SQL_GUID [2]	1	4000	-	-	Y	-	VARCHAR	16	-	Y
ntext [2]	-	-	-	-	-	-	CLOB	2147483647	-	Y
DUMMY2000 [3]	1	38	-84	127	-	-	DOUBLE	-	-	-

Notes:

1. This type mapping is only valid with Microsoft SQL Server Version 6.5.
2. This type mapping is only valid with Microsoft SQL Server Version 7.
3. This type mapping is only valid with Windows 2000 operating systems.

ODBC data sources

Table 58. 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_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_NUMERIC	32	32	0	31	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_REAL	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	32672	-	-	-	-	VARCHAR	-	-	N
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	N
SQL_VARBINARY	1	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N
SQL_LONGVARBINARY	-	-	-	-	-	-	BLOB	-	-	Y
SQL_DATE	-	-	-	-	-	-	DATE	4	-	Y
SQL_TIME	-	-	-	-	-	-	TIME	3	-	Y
SQL_TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_BIGINT	-	-	-	-	-	-	DECIMAL	-	-	-

Table 58. 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	-	-	-	-	GRAPHIC	-	-	N
SQL_WCHAR	128	16336	-	-	-	-	VARGRAPHIC	-	-	N
SQL_WVARCHAR	1	16336	-	-	-	-	VARGRAPHIC	-	-	N
SQL_WLONGVARCHAR		-	-	-	-	-	DBCLOB	1073741823	-	NY

Sybase data sources

Table 59. Sybase CTLIB 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
int	-	-	-	-	-	-	INTEGER	-	-	-
intn	-	-	-	-	-	-	INTEGER	-	-	-
smallint	-	-	-	-	-	-	SMALLINT	-	-	-
tinyint	-	-	-	-	-	-	SMALLINT	-	-	-
bit	-	-	-	-	-	-	SMALLINT	-	-	-
float	-	8	-	-	-	-	DOUBLE	-	-	-
floatn	-	8	-	-	-	-	DOUBLE	-	-	-

Table 59. Sybase CTLIB 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
float	-	4	-	-	-	-	REAL	-	-	-
floatn	-	4	-	-	-	-	REAL	-	-	-
real	-	-	-	-	-	-	REAL	-	-	-
money	-	-	-	-	-	-	DECIMAL	19	4	-
moneyn	-	-	-	-	-	-	DECIMAL	19	4	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
decimal	1	31	0	31	-	-	DECIMAL	-	-	-
decimal	32	32	-	-	-	-	DOUBLE	-	-	-
decimaln	1	31	0	31	-	-	DECIMAL	-	-	-
decimaln	32	32	-	-	-	-	DOUBLE	-	-	-
numeric	1	31	0	31	-	-	DECIMAL	-	-	-
numeric	32	32	-	-	-	-	DOUBLE	-	-	-
numericn	1	31	0	31	-	-	DECIMAL	-	-	-
numericn	32	32	-	-	-	-	DOUBLE	-	-	-
char	1	254	-	-	-	-	CHAR	-	-	Y
sysname	1	254	-	-	-	-	CHAR	-	-	Y
char	255	255	-	-	-	-	VARCHAR	-	-	Y
varchar	1	255	-	-	-	-	VARCHAR	-	-	Y
nchar	1	127	-	-	-	-	GRAPHIC	-	-	-
nchar	128	255	-	-	-	-	VARGRAPHIC	-	-	-
nvarchar	1	255	-	-	-	-	VARGRAPHIC	-	-	-
binary	1	254	-	-	-	-	CHAR	-	-	Y
binary	255	255	-	-	-	-	VARCHAR	-	-	Y
text	-	-	-	-	-	-	CLOB	-	-	-

Table 59. Sybase CTLIB 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
image	-	-	-	-	-	-	BLOB	-	-	-
varbinary	1	255	-	-	-	-	VARCHAR	-	-	Y
datetime	-	-	-	-	-	-	TIMESTAMP	-	-	-
datetimen	-	-	-	-	-	-	TIMESTAMP	-	-	-
smalldatetime	-	-	-	-	-	-	TIMESTAMP	-	-	-
timestamp	-	-	-	-	-	-	VARCHAR	8	-	Y

Appendix I. 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 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 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 type mapping, or create a new type mapping with the CREATE TYPE MAPPING statement.

The following tables show the default reverse mappings between DB2 for 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 60. 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
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	N
DATE	-	4	-	-	-	-	DATE	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
CLOB	-	-	-	-	-	-	CLOB	-	-	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-

DB2 for iSeries data sources

Table 61. 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
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
DECIMAL	-	-	-	-	-	-	NUMERIC	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	N
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHARACTER	-	-	Y
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
VARGRAPHIC	-	-	-	-	-	-	VARG	-	-	N
DATE	-	4	-	-	-	-	DATE	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
CLOB	-	-	-	-	-	-	CLOB	-	-	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-

DB2 Server for VM and VSE data sources

Table 62. DB2 Server 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
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
VARGRAPH	-	-	-	-	-	-	VARGRAPH	-	-	N
DATE	-	4	-	-	-	-	DATE	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
CLOB	-	-	-	-	-	-	CLOB	-	-	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-

DB2 for UNIX and Windows data sources

Table 63. DB2 for 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
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
BIGINT	-	8	-	-	-	-	BIGINT	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	-	N
DATE	-	4	-	-	-	-	DATE	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
CLOB	-	-	-	-	-	-	CLOB	-	-	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-

Informix data sources

Table 64. 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]	-	19	0	-	-	-	DECIMAL	21	-	-
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	32672	-	-	N	-	LVARCHAR	-	-	-

Table 64. 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 only valid with Informix server Version 7 (or lower).
2. This type mapping is only valid with Informix server Version 8 (or higher).

Oracle SQLNET data sources

Note: The DB2 for 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.

Table 65. 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
DOUBLE	0	8	0	0	N	\0	FLOAT	126	0	N
REAL	0	4	0	0	N	\0	FLOAT	63	0	N
DECIMAL	0	0	0	0	N	\0	NUMBER	0	0	N
SMALLINT	0	2	0	0	N	\0	NUMBER	5	0	N

Table 65. Oracle SQLNET 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
INTEGER	0	4	0	0	N	\0	NUMBER	10	0	N
CHARACTER	1	254	0	0	N	\0	CHAR	0	0	N
VARCHAR	1	4000	0	0	N	\0	VARCHAR2	0	0	N
CLOB	0	2147483647	0	0	N	\0	LONG	0	0	N
CHARACTER	0	0	0	0	Y	\0	RAW	0	0	Y
VARCHAR	1	2000	0	0	Y	\0	RAW	0	0	Y
BLOB	0	2147483647	0	0	Y	\0	LONG RAW	0	0	Y
TIMESTAMP	0	10	0	0	N	\0	DATE	0	0	N
DATE	0	4	0	0	N	\0	DATE	0	0	N
TIME	0	3	0	0	N	\0	DATE	0	0	N

Oracle NET8 data sources

Note: The DB2 for 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.

Table 66. 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
DOUBLE	0	8	0	0	N	\0	FLOAT	126	0	N
REAL	0	4	0	0	N	\0	FLOAT	63	0	N
DECIMAL	0	0	0	0	N	\0	NUMBER	0	0	N
SMALLINT	0	2	0	0	N	\0	NUMBER	5	0	N
INTEGER	0	4	0	0	N	\0	NUMBER	10	0	N
CHARACTER	1	254	0	0	N	\0	CHAR	0	0	N
VARCHAR	1	4000	0	0	N	\0	VARCHAR2	0	0	N
CLOB	0	2147483647	0	0	N	\0	CLOB	0	0	N
CHARACTER	0	0	0	0	Y	\0	RAW	0	0	Y
VARCHAR	1	2000	0	0	Y	\0	RAW	0	0	Y
BLOB	0	2147483647	0	0	Y	\0	BLOB	0	0	Y
TIMESTAMP	0	10	0	0	N	\0	DATE	0	0	N
DATE	0	4	0	0	N	\0	DATE	0	0	N
TIME	0	3	0	0	N	\0	DATE	0	0	N

Microsoft SQL Server data sources

Table 67. 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
INTEGER	-	-	-	-	-	-	int	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
DOUBLE	-	8	-	-	-	-	float	-	-	-
DECIMAL	-	-	-	-	-	-	decimal	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
VARCHAR	1	8000	-	-	N	-	varchar	-	-	-
VARCHAR	8001	32672	-	-	N	-	text	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
VARCHAR	1	8000	-	-	Y	-	varbinary	-	-	-
VARCHAR	8001	32672	-	-	Y	-	image	-	-	-
LONG VARCHAR	-	32700	-	-	Y	-	image	-	-	-
BLOB	-	-	-	-	-	-	image	-	-	-
TIMESTAMP	-	10	-	-	-	-	datetime	-	-	-
TIME	-	3	-	-	-	-	datetime	-	-	-
DATE	-	4	-	-	-	-	datetime	-	-	-

Sybase data sources

These data type mappings only apply to the CTLIB wrapper. The DBLIB wrapper is read-only and does not support transparent DDL in a Version 8 federated system.

Table 68. Sybase CTLIB 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
INTEGER	-	-	-	-	-	-	integer	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
BIGINT	-	-	-	-	-	-	decimal	19	0	-
DOUBLE	-	-	-	-	-	-	float	-	-	-
REAL	-	-	-	-	-	-	real	-	-	-
DECIMAL	-	-	-	-	-	-	decimal	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
VARCHAR	1	255	-	-	N	-	varchar	-	-	-
VARCHAR	256	32672	-	-	N	-	text	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
BLOB	-	-	-	-	-	-	image	-	-	-
VARCHAR	1	255	-	-	Y	-	varbinary	-	-	-
VARCHAR	256	32672	-	-	Y	-	image	-	-	-
GRAPHIC	-	-	-	-	-	-	nchar	-	-	-
VARGRAPHIC	1	255	-	-	-	-	nvarchar	-	-	-
DATE	-	-	-	-	-	-	datetime	-	-	-
TIME	-	-	-	-	-	-	datetime	-	-	-
TIMESTAMP	-	-	-	-	-	-	datetime	-	-	-

Appendix J. Quick reference - useful Internet Web sites

The following table lists Internet Web sites that provide useful information about DB2 products, updates, support, and education. The DB2 Relational Connect web site contains general information about federated systems.

Table 69. Quick reference table of useful Internet Web sites

Subject	Web site address
DB2 Product Family	www.ibm.com/software/data/db2/
DB2 Relational Connect	www.ibm.com/software/data/db2/relconnect/
DB2 Life Sciences Data Connect	www.ibm.com/solutions/lifesciences/
DB2 Connect	www.ibm.com/software/data/db2/db2connect/
DB2 FixPaks	www.ibm.com/software/data/db2/udb/winos2unix/support/
DB2 Documentation and FixPak Release Notes	www.ibm.com/software/data/db2/udb/winos2unix/support/
DB2 Support	www.ibm.com/software/data/db2/udb/winos2unix/support/
IBM Education	www.ibm.com/services/learning/
DB2 Spatial Extender	www.ibm.com/software/data/spatial/
DB2 DataPropagator (Replication)	www.ibm.com/software/data/dpropr/
DB2 Warehouse Manager	www.ibm.com/software/data/db2/datawarehouse/

Glossary

Glossary terms for federated systems

column options. In a federated system, parameters of the ALTER NICKNAME statement that describe the values in certain columns of the data source object that a nickname references. This information is added to the global catalog and used by the DB2 query optimizer to develop better access plans. Column options provide a way to tell the data source wrapper to handle a column in a different way than it normally would.

compensation. In a federated system, the ability for DB2 to process SQL that is not supported by a data source. DB2 will not push down a query fragment if the data source cannot process it, or if DB2 can process it faster than the data source can. If the data source cannot process it, DB2 will process it instead. A federated server compensates for the loss of functionality at the data source by either simulating the data source function, or by returning the set of data to the federated server and performing the function locally. See also query optimizer and push-down processing.

data source. In a federated system, typically a relational DBMS instance and one or more databases supported by that instance. However, there are other types of data sources that you can include in your federated system, such as flat-file databases and table-structured files.

data source object. In a federated system, an object at the data source that you want to perform operations against. Examples include: a database table, a database view, a spreadsheet list. You create nicknames on the federated server to identify the data source objects.

data type mapping. In a federated system, the mapping of a data type used at a data source to a DB2 data type. For example, the Oracle type FLOAT maps by default to the DB2 type DOUBLE. DB2 supplies default mappings for most kinds of data types; the default mappings are in the wrappers.

federated database. The DB2 database in a federated system. Contains the system catalog, referred to as the *global catalog*, and any local tables, local views, or federated views.

federated server. The DB2 server in a federated system. Any number of DB2 instances can be configured to function as federated servers. You can use existing DB2 instances as your federated server, or you can create new ones specifically for the federated system.

federated system. A special type of distributed database management system (DBMS) that allows you to query and manipulate data located on other servers. The data can be in database managers such as Oracle, Sybase, Informix, and Microsoft SQL Server, or it can be in lists or stores such as a spreadsheet, Web site, or data mart. A federated system consists of a DB2 instance that will operate as a server, a database that will serve as the federated database, one or more data sources, and clients (users and applications) who will access the database and data sources. A federated system allows you to send distributed requests to multiple data sources within a single SQL statement.

foreign server. In a federated system, another term for data sources. Most often seen with the SQL/MED standard. See data sources.

function mapping. In a federated system, a mapping between a data source function and an existing DB2 function. DB2 supplies default mappings between existing built-in data source functions and built-in DB2 functions; the default mappings are in the wrapper.

function mapping options. In a federated system, parameters of the CREATE FUNCTION MAPPING statement to which you can assign

values that pertain to the mapping being created or to the data source function within the mapping. Such values, for example, can include estimated statistics on the overhead that will be consumed when the data source function is invoked. The query optimizer uses these estimates to decide if the function should be invoked by the data source or by DB2 when the data is returned from the data source. See function mapping.

function template. A DB2 function that you create for the purpose of invoking a function on a data source. A federated server can recognize a data source function only if there is a mapping between the data source function and a counterpart function at the federated database. When no counterpart exists, or when you want to force the federated server to use the data source function, you can create a function template to act as the counterpart.

global catalog. In a federated system, the database system catalog. This catalog contains information about objects in the federated database and information about objects at the data sources. Because the catalog contains information about the entire federated system, it is called a *global catalog*. The information in the global catalog is used by the DB2 query optimizer to plan the best way to process SQL statements that involve the data sources.

heterogeneous replication. Replication between DB2 and non-DB2 relational databases. See also federated system

index specification. In a federated system, a set of metadata about a data source object index. The query optimizer uses this information to expedite the processing of distributed requests. When a nickname is created for a data source object, the federated server gathers any index information about that object and stores the information in the global catalog.

nickname. In a federated system, identifiers 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. Examples of data source objects include tables, views, synonyms, table-structured files, and search algorithms. Nicknames are not alternative names for data source objects in the same way that aliases are; they are pointers by which the federated server references these objects.

pass-through. In a federated system, a special DB2 session used to submit SQL statements directly to DBMSs using the SQL dialect associated with that data source. You use a pass-through session to perform an operation that is not possible with DB2 SQL/API, or to perform actions not supported by SQL.

push-down processing. In a federated system, the processing of segments of a query at a data source instead of at the federated server.

query optimizer. In a federated system, a feature of the DB2 SQL Compiler that analyzes the distributed queries and determines the most efficient way to run the query. The global optimizer evaluates queries based on resource cost. See push-down processing.

server definition. In a federated system, the name and information that defines the data sources to the federated database. The server definition is used by the wrapper when SQL statements that use nicknames are submitted to the federated database.

server options. In a federated system, information within a server definition that either configures the wrapper itself, or affects the way that DB2 uses the wrapper. Server option values are stored in the global catalog.

user mapping. In a federated system, the association between the authorization ID at the federated server and the authorization ID at the data source. User mappings are needed so that distributed requests can be sent to the data source.

user options. In a federated system, parameters of the CREATE USER MAPPING and ALTER USER MAPPING statements to which values related to authorization are assigned. For example, suppose that a user has the same ID

with different passwords for the federated database and a data source. For the user to access the data source, it is necessary to map the passwords to one another. This is accomplished with the user option REMOTE_PASSWORD. See user mapping.

wrapper. In a federated system, the mechanism that the federated server uses to communicate with and retrieve data from the data sources. To implement a wrapper, the federated server uses routines stored in a library called a wrapper module. These routines allow the federated server to perform operations such as connecting to a data source and retrieving data from it iteratively.

Notices

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 be incorporated in new editions of the publication. IBM may make

improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product, and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information that has been exchanged, should contact:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems, and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious, and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs, in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (*your company name*) (*year*). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. *_enter the year or years_*. All rights reserved.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both, and have been used in at least one of the documents in the DB2 UDB documentation library.

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

The following terms are trademarks or registered trademarks of other companies and have been used in at least one of the documents in the DB2 UDB documentation library:

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Index

E

F

G

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Contacting IBM

In the United States, call one of the following numbers to contact IBM:

- 1-800-237-5511 for customer service
- 1-888-426-4343 to learn about available service options
- 1-800-IBM-4YOU (426-4968) for DB2 marketing and sales

In Canada, call one of the following numbers to contact IBM:

- 1-800-IBM-SERV (1-800-426-7378) for customer service
- 1-800-465-9600 to learn about available service options
- 1-800-IBM-4YOU (1-800-426-4968) for DB2 marketing and sales

To locate an IBM office in your country or region, check IBM's Directory of Worldwide Contacts on the web at www.ibm.com/planetwide

Product information

Information regarding DB2 Universal Database products is available by telephone or by the World Wide Web at www.ibm.com/software/data/db2/udb

This site contains the latest information on the technical library, ordering books, client downloads, newsgroups, FixPaks, news, and links to web resources.

If you live in the U.S.A., then you can call one of the following numbers:

- 1-800-IBM-CALL (1-800-426-2255) to order products or to obtain general information.
- 1-800-879-2755 to order publications.

For information on how to contact IBM outside of the United States, go to the IBM Worldwide page at www.ibm.com/planetwide

Part Number: CT16ENA

Printed in U.S.A.

(1P) P/N: CT16ENA

GC27-1224-00

Spine information:

IBM® DB2 Universal Database™ DB2 Federated Systems Guide Version 8