IBM DB2 Information Integrator

Federated Systems Guide

Version 8.2

SC18-7364-01

IBM DB2 Information Integrator

Federated Systems Guide

Version 8.2

SC18-7364-01

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 317.

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

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

About this book

This book describes how to work with a federated system after you set up and configure the federated server to access the data sources.

This book covers the following topics:

- An introduction to federated system concepts, components, and capabilities
- Instructions for modifying the federated server setup and data source configurations
- An explanation of federated system transaction support
- Recommendations for administering and tuning your federated system for optimum performance
- An explanation of the items that you need to consider when you develop applications for a federated system
- Extensive reference information for each data source

In this book, a vertical line in the left margin indicates a technical change to the text.

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 federated system. Use this book to manage a federated system to access data from relational and nonrelational data sources. Programmers and other users who require an understanding of the configuration, administration, application development issues, and use of a federated system can also use this book.

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

Part 1. Introduction

Chapter 1. Overview of a federated system

This chapter describes the features of a federated system, defines federated concepts and terminology, 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 an XML tagged file in a single SQL statement. The following figure shows the components of a federated system and a sample of the data sources you can access.

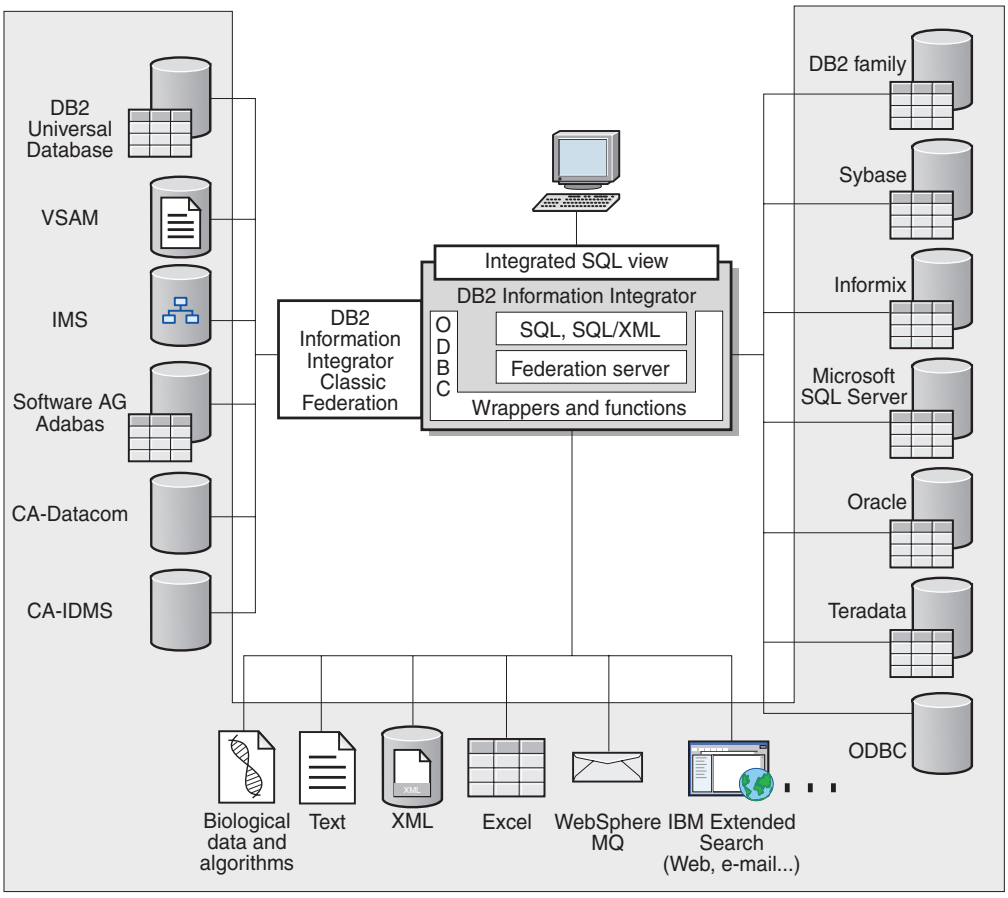


Figure 1. The components of a federated system

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 stored locally in the federated database

- Update data in relational data sources, as if the data is stored in the federated database
- Replicate data to and from relational data sources
- Take advantage of the data source processing strengths, by sending 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 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 servers, or you can create new ones specifically for the federated system.

The DB2 instance that manages the federated system is called a *server* because it 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 DB2 instance that manages the federated system 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. Application processes connect and submit requests to the database within the federated server. 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 (over TCP/IP) to communicate with DB2 family instances. However, unlike other application servers, a federated server uses the native client of the data source to access the data source. For example, a federated server uses the Sybase Open Client to access Sybase data sources and an Microsoft® SQL Server ODBC Driver to access Microsoft SQL Server data sources.

Related concepts:

- "What is a data source?" on page 4

What is a data source?

In a federated system, a *data source* can be a relational DBMS instance (such as Oracle or Sybase) or a nonrelational data source (such as BLAST search algorithm or an XML tagged file). Through some data sources you can access other data sources. For example, through the Extended Search data source you can access data sources such as Lotus® Notes databases, Microsoft® Access, Microsoft Index Server, Web search engines, and Lightweight Directory Access Protocol (LDAP) directories.

The method, or protocol, used to access a data source depends on the type of data source. For example, DRDA® is used to access DB2® for z/OS™ and OS/390® data sources and the Documentum Client API/Library is used to access Documentum data sources.

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.

Related concepts:

• "The federated database" on page 7

Related reference:

• "Supported data sources" on page 5

Supported data sources

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

Table 1. Supported data source versions and access methods.

Data source	Supported versions	Access method
DB2 Universal Database™ for Linux, UNIX, and Windows®	7.2, 8.1, 8.2	DRDA®
DB2 Universal Database for z/OS™ and OS/390®	6.1, 7.1 with the following APARs applied: • PQ62695 • PQ55393 • PQ56616 • PQ54605 • PQ46183 • PQ62139 8.1	DRDA
DB2 Universal Database for iSeries™	5.1 • with the following APARs applied: – SE06003 – SE06872 – II13348 • with the following PTFs applied: – SI05990 SI05991 5.2 with PTF SI0735 applied.	DRDA
DB2 Server for VM and VSE	7.1 (or later) with fixes for APARs for schema functions applied.	DRDA
Informix™	7.31, 8.32, 8.4, 9.3, 9.4	Informix Client SDK V2.7 (or later)

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

Data source	Supported versions	Access method
ODBC	3.x	ODBC driver for the data source, such as Redbrick ODBC Driver to access Redbrick.
OLE DB	2.7, 2.8	OLE DB 2.0 (or later)
Oracle	8.0.6, 8.1.6, 8.1.7, 9.0, 9.1, 9.2, 9i, 10g	Oracle net client or NET8 client software
Microsoft SQL Server	7.0, 2000 SP3 and later service packs on that release	On Windows, the Microsoft SQL Server Client ODBC 3.0 (or later) driver. On UNIX, the DataDirect Technologies (formerly MERANT) Connect ODBC 3.7 (or later) driver.
Sybase	11.9.2, 12.x	Sybase Open Client ctlib interface
Teradata	V2R3, V2R4, V2R5	Teradata Call-Level Interface, Version 2 (CLIv2) Release 04.06 (or later)
BLAST	2.2.3 and later 2.2 fixpacks supported	BLAST daemon (supplied with the wrapper)
BioRS	v5.0.14	None
Documentum	3.x, 4.x	Documentum Client library/APL3.1.7a (or later)
Entrez (PubMed and GenBank data sources)	1.0	None
HMMER	2.2g, 2.3	HMMER daemon (supplied with the wrapper)
IBM Lotus Extended Search	4.0.1, 4.0.2	Extended Search Client Library (supplied with the wrapper)
Microsoft Excel	97, 2000, 2002, 2003	Excel 97, 2000, 2002, or 2003 installed on the federated server
PeopleSoft	8.x	IBM WebSphere Business Integration Adapter for PeopleSoft v2.3.1, 2.4
SAP	3.x, 4.x	IBM WebSphere Business Integration Adapter for mySAP.com v2.3.1, 2.4
Siebel	7, 7.5, 2000	IBM WebSphere Business Integration Adapter for Siebel eBusiness Applications v2.3.1, 2.4
Table-structured files		None
User-defined functions for KEGG	Supported	
User-defined functions for Life Sciences	Supported	

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

Data source	Supported versions	Access method
Web services	SOAP 1.0., 1.1, WSDL 1.0, 1.1 specifications	HTTP
XML	1.0 specification	None

Related concepts:

- "What is a data source?" on page 4

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 a system catalog. The federated database system catalog contains 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 system processes SQL statements as if the data sources were ordinary relational tables or views within the federated database. As a result:

- The federated system can join relational data with data in nonrelational formats. This is true even when the data sources use different SQL dialects, or do not support SQL at all.
- The characteristics of the federated database take precedence when there are differences between the characteristics of the federated database and the characteristics of the data sources:
  - Suppose the code page used by the federated server is different than the code page used by the data source. Character data from the data source is converted based on the code page used by the federated database, when that data is returned to a federated user.
  - Suppose the collating sequence used by the federated server is different than the collating sequence used by the data source. Any sort operations on character data are performed at the federated server instead of at the data source.

Related concepts:

- "The SQL Compiler" on page 8
- "The federated database system catalog" on page 7

The federated database system catalog

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 the ALTER NICKNAME statement to change the local name of the data source columns.

For relational data sources, the information stored in the global catalog includes both remote and local information.

For nonrelational 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 SYSCAT.TABLES, SYSCAT.TABOPTIONS, SYSCAT.INDEXES, SYSCAT.COLUMNS, and SYSCAT.COLOPTIONS catalog views in the federated database.

The global catalog also includes other information about the data sources. For example, the global catalog includes information that the federated server uses to connect to the data source and map the federated user authorizations to the data source user authorizations. The global catalog contains attributes about the data source that you explicitly set, such as server options.

Related concepts:

- "The SQL Compiler" on page 8

Related reference:

- Chapter 19, "Views in the global catalog table containing federated information," on page 213

The SQL Compiler

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.

Related concepts:

- "Wrappers and wrapper modules" on page 11
- "The query optimizer" on page 8

The query optimizer

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® UDB 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 UDB 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
- Whether there is a usable materialized query table on the federated server that represents the same query result

The query optimizer generates local and remote access plans for processing a query fragment, based on resource cost. DB2 UDB 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 UDB submits these fragments to the data sources. After the data sources process the fragments, the results are retrieved and returned to DB2 UDB. If DB2 UDB performed any part of the processing, it combines its results with the results retrieved from the data source. DB2 UDB then returns all results to the client.

Related concepts:

- "The SQL Compiler" on page 8
- "Compensation" on page 9
- "Tuning query processing" on page 131

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 Information Integrator does not pushdown the CUBE grouping to the data source, but processes the CUBE itself. The ability by DB2 Information Integrator 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 Information Integrator 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.

For relational data sources, 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 the relational data source dialect, the federated server can implement this construct on behalf of the data source.

DB2 Information Integrator can compensate for differences in SQL dialects. An example of this ability is the common-table-expression clause. 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.

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

Related concepts:

• "Pass-through sessions" on page 10

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.

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.
- LOBs are not supported in pass-through sessions.

Related concepts:

- "Wrappers and wrapper modules" on page 11
- "Querying data sources directly with pass-through" on page 208

Wrappers and wrapper modules

Wrappers are mechanisms by which the federated server interacts with data sources. The federated server uses routines stored in a library called a *wrapper module* to implement a wrapper. These routines allow the federated server to perform operations such as connecting to a data source and retrieving data from it iteratively. Typically, the DB2® federated instance owner uses the CREATE WRAPPER statement to register a wrapper in the federated database. You can register a wrapper as fenced or trusted using the DB2_FENCED wrapper option.

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 one wrapper for the DB2 data source objects and one wrapper for the Informix data source objects. Once these wrappers are registered in the federated database, you can use these wrappers to access other objects from those data sources. For example, you can use the DRDA® wrapper with all DB2 family data source objects—DB2 for Linux, UNIX®, and Windows®, DB2 for z/OS and OS/390®, DB2 for iSeries, and DB2 Server for VM and VSE.

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

A wrapper performs many tasks. Some of these tasks are:

- It connects to the data source. The wrapper uses the standard connection API of the data source.
- It submits queries to the data source.
  - For data sources that support SQL, the query is submitted in SQL.
  - For data sources that do not support SQL, the query is translated into the native query language of the source or into a series of source API calls.
- It receives results sets from the data source. The wrapper uses the data source standard APIs for receiving results set.
- It responds to federated server queries about the default data type mappings for a data source. The wrapper contains the default type mappings that are used when nicknames are created for a data source object. For relational wrappers, data type mappings that you create override the default data type mappings. User-defined data type mappings are stored in the global catalog.

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

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

Related concepts:

- "Server definitions and server options" on page 13

Related tasks:

- "Trusted and fenced mode process environments" in the *IBM DB2 Information Integrator Wrapper Developer's Guide*
- "Altering a wrapper" on page 25

Related reference:

- "ALTER WRAPPER statement" in the *SQL Reference, Volume 2*
- Chapter 20, "Wrapper options for federated systems," on page 217
- "Default wrapper names" on page 12

Default wrapper names

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 names
DB2 Universal Database™ for Linux, 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
Informix	INFORMIX
Microsoft® SQL Server	MSSQLODBC3
ODBC	ODBC
OLE DB	OLEDB
Oracle	NET8
Sybase	CTLIB
Teradata	TERADATA
BLAST	None
BioRS	None

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

Data source	Default wrapper names
Documentum	None
Entrez	None
Extended Search	None
HMMER	None
Microsoft Excel	None
Table-structured files	None
Web Services	None
WebSphere Business Integration	None
XML	None

Related concepts:

- "Wrappers and wrapper modules" on page 11

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. This information includes:

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

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 Information Integrator uses the wrapper.

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

Related concepts:

- "User mappings" on page 14

Related reference:

- Chapter 21, "Server options for federated systems," on page 219

User mappings

When a federated server needs to pushdown a request to a data source, the server must first establish a connection to the data source.

For most data sources, the federated server does this by using a valid user ID and password to that data source. When a user ID and password is required to connect to a data source, you can define an association between the federated server authorization ID and the data source user ID and password. This association can be created for each user ID that will be using the federated system to send distributed requests. This association is called a *user mapping*.

In some cases, you do not need to create a user mapping if the user ID and password you use to connect to the federated database are the same as those you use to access the remote data source.

Related tasks:

- "Registering user mappings for a data source" in the *IBM DB2 Information Integrator Data Source Configuration Guide*

Related reference:

- Chapter 22, "User mapping options for federated systems," on page 235

Nicknames and data source objects

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

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

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

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

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 some DB2 UDB utility operations on nicknames.

You cannot use the Cross Loader utility to cross load into a nickname.

Related concepts:

- "Nickname column options" on page 16

Related reference:

- Chapter 24, "Nickname column options for federated systems," on page 247
- Chapter 23, "Nickname options for federated systems," on page 237
- "Valid data source objects" on page 15

Valid data source objects

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

Table 3. Valid data source objects

Data source	Valid objects
DB2 for Linux, UNIX, and Windows	Nicknames, materialized query tables, tables, views
DB2 for z/OS and OS/390	Tables, views
DB2 for iSeries	Tables, views
DB2 for VM and VSE	Tables, views
Informix	Tables, views, synonyms
Microsoft SQL Server	Tables, views
ODBC	Tables, views
Oracle	Tables, views, synonyms
Sybase	Tables, views
Teradata	Tables, views
BLAST	FASTA files indexed for BLAST search algorithms
BioRS	BioRS databanks
Documentum	Objects and registered tables in a Documentum Docbase
Entrez	Entrez databases
Extended Search	Files from data sources such as Lotus Notes databases, Microsoft Access, Microsoft Index Server, Web search engines, and LDAP directories.
HMMER	HMM database files (libraries of Hierarchical Markov Models, such as PFAM), that can be searched by HMMER's hmmpfam or hmmsearch programs.

Table 3. Valid data source objects (continued)

Data source	Valid objects
Microsoft Excel	.xls files (only the first sheet in the workbook is accessed)
Table-structured files	Text files that meet a specific format.
Websphere Business Integration adapters	Websphere Business Integration business objects that map to BAPIs in SAP, business components in Siebel, and component interfaces in PeopleSoft
Web Services	Operations in a Web services description language file
XML-tagged files	Sets of items in an XML document

Related concepts:

- "Nicknames and data source objects" on page 14
- "Nickname column options" on page 16

Nickname 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 *nickname column options*. The nickname 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.

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

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

Suppose that a data source has a collating sequence that differs from the federated database collating sequence. The federated server typically would not sort any columns containing character data at the data source. It would return the data to the federated database and perform the sort locally. However, suppose that the column is a character data type (CHAR or VARCHAR) and contains only numeric characters ('0','1',...,'9'). You can indicate this by assigning a value of 'Y' to the NUMERIC_STRING nickname 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 nickname column options for relational nicknames using the ALTER NICKNAME statemens. You can define nickname column options for nonrelational nicknames using the CREATE NICKNAME and ALTER NICKNAME statements.

Related concepts:

- "Data type mappings" on page 17

Related tasks:

- "Working with nicknames" on page 97

Related reference:

- Chapter 24, "Nickname column options for federated systems," on page 247

Data type mappings

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

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

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

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

For relational data sources, you can override the default data type mappings. For example, by default the Informix INTEGER data type maps to the DB2 INTEGER data type. You could override the default mappings and map Informix's INTEGER data type to DB2 DECIMAL(10,0) data type.

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

Related concepts:

- "Data type mappings in a federated system" on page 45

Function mappings

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

For relational data sources, you can create a function mapping when you want to use a data source function that the federated server does not recognize. The

mapping that you create is between the data source function and a DB2 counterpart function at the federated database. Function mappings are typically used when a new built-in function or a new user-defined function become available at the data source. Function mappings are also used when a DB2 counterpart function does not exist. In this case, you must also create a function template.

Related concepts:

- "Function mappings in a federated system" on page 57
- "Index specifications" on page 18

Index specifications

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

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

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

Related concepts:

- "Index specifications in a federated system" on page 71

Collating sequences

The order in which character data is sorted in a database depends on the structure of the data and the collating sequence defined for the database.

Suppose that the data in a database is all uppercase letters and does not contain any numeric or special characters. A sort of the data should result in the same output, regardless of whether the data is sorted at the data source or at the federated database. The collating sequence used by each database should not impact the sort results. Likewise, if the data in the database is all lowercase letters or all numeric characters, a sort of the data should produce the same results regardless of where the sort actually is performed.

If the data consists of any of the following structures:

- A combination of letters and numeric characters

- Both uppercase and lowercase letters
- Special characters such as @, #, €

Sorting this data can result in different outputs, if the federated database and the data source use different collating sequences.

In general terms, a *collating sequence* is a defined ordering for character data that determines whether a particular character sorts higher, lower, or the same as another character.

How collating sequences determine sort orders

A collating sequence determines the sort order of the characters in a coded character set. A *character set* is the aggregate of characters that are used in a computer system or programming language. In a *coded* character set, each character is assigned to a different number within the range of 0 to 255 (or the hexadecimal equivalent thereof). The numbers are called *code points*; the assignments of numbers to characters in a set are collectively called a *code page*.

In addition to being assigned to a character, a code point can be mapped to the character's position in a sort order. In technical terms, then, a collating sequence is the collective mapping of a character set's code points to the sort order positions of the set's characters. A character's position is represented by a number; this number is called the *weight* of the character. In the simplest collating sequence, called an *identity sequence*, the weights are identical to the code points.

Suppose that database ALPHA uses the default collating sequence of the EBCDIC code page, and that database BETA uses the default collating sequence of the ASCII code page. Sort orders for character strings at these two databases would differ, as shown in the following example:

```
SELECT.....
  ORDER BY COL2

EBCDIC-Based Sort           ASCII-Based Sort

COL2                        COL2
----                        ----
V1G                         7AB
Y2W                         V1G
7AB                         Y2W
```

Similarly, character comparisons in a database depend on the collating sequence defined for that database. In this example, database ALPHA uses the default collating sequence of the EBCDIC code page. Database BETA uses the default collating sequence of the ASCII code page. Character comparisons at these two databases would yield different results, as shown in the following example:

```
SELECT.....
  WHERE COL2 > 'TT3'

EBCDIC-Based Results     ASCII-Based Results

COL2                     COL2
----                     ----
TW4                      TW4
X82                      X82
39G
```

Setting the local collating sequence to optimize queries

Administrators can create federated databases with a particular collating sequence that matches a data source collating sequence. Then for each data source server definition, the COLLATING_SEQUENCE server option is set to 'Y'. This setting tells the federated database that the collating sequences of the federated database and the data source match.

You set the federated database collating sequence as part of the CREATE DATABASE API. Through this API, you can specify one of the following sequences:

- An identity sequence
- A *system* sequence (the sequence used by the operating system that supports the database)
- A *customized* sequence (a predefined sequence that DB2 UDB supplies or that you define yourself)

Suppose that the data source is DB2 for z/OS and OS/390. Sorts that are defined in an ORDER BY clause are implemented by a collating sequence based on an EBCDIC code page. To retrieve DB2 for z/OS and OS/390 data sorted in accordance with ORDER BY clauses, configure the federated database so that it uses the predefined collating sequence based on the appropriate EBCDIC code page.

Related concepts:

- "Server characteristics affecting pushdown opportunities" on page 134
- "Collating sequences in a federated system" in the *IBM DB2 Information Integrator Data Source Configuration Guide*

Related tasks:

- "Creating a federated database" in the *IBM DB2 Information Integrator Data Source Configuration Guide*

Related reference:

- "National language versions" in the *Administration Guide: Planning*
- "Federated database national language considerations" in the *IBM DB2 Information Integrator Data Source Configuration Guide*

How you interact with a federated system

Because the federated database is a DB2® Universal Database, you can interact with a federated system using any one of these methods:

- The DB2 command line processor (CLP)
- The DB2 Command Center GUI
- The DB2 Control Center GUI
- Application programs
- DB2 family tools
- Web services providers

The steps in the federated documentation provide the commands and SQL statements that can be entered in the DB2 command line processor or the DB2 Command Center GUI. The documentation indicates when tasks can be performed

through the DB2 Control Center GUI. Since the DB2 Control Center GUI is intuitive, the steps to perform these tasks through the DB2 Control Center are not included in this documentation.

DB2 command line processor (CLP)

You can perform most of the tasks necessary to setup, configure, tune, and maintain the federated system through the DB2 command line processor. In some cases you must use either the DB2 command line processor or the DB2 Command Center. For example:

- Create, alter, or drop user-defined data type mappings
- Create, alter, or drop user-defined function mappings

DB2 Command Center

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

DB2 Control Center

The DB2 Control Center GUI allows you to perform most of the tasks necessary to setup, configure, and modify the federated system. The DB2 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.

You can either use a wizard to create the federated objects, or you can create each object individually.

Use the DB2 Control Center to configure access to Web services, WebSphere® Business Integration, and XML data sources. The features built into the DB2 Control Center simplify the steps that are required for you to configure the federated server to access these data sources.

The DB2 Control Center GUI is the easiest way to perform the essential data source configuration tasks:

- Create the wrappers and set the wrapper options
- Specify the environment variables for your data source
- Create the server definitions and set the server options
- Create the user mappings and set the user options
- Create the nicknames and set the nickname options or column options

After you configure the federated server to access your data sources, you can use the DB2 Control Center to:

- Modify the data source configuration
- Monitor the status of the nicknames and servers
- Maintain current statistics for your nicknames
- Create and modify cache tables

- Specify informational constraints on nicknames
- Create remote tables through DB2 Information Integrator using transparent DDL

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, applications submit queries in DB2 SQL to the federated database. DB2 Information Integrator then distributes the queries to the appropriate data sources, collects the requested data, and returns this data to the applications. However, since DB2 Information Integrator 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

DB2 family tools

You can also interact with a federated database using host and midrange tools such as:

- DB2 SPUFI on DB2 for z/OS™ and OS/390®
- Interactive SQL (STRSQL) on DB2 for iSeries™

Web services providers

You can also interact with a federated database through web services providers using the Web Services wrapper.

Related concepts:

- "The Web services wrapper and the Web services description language document" in the *IBM DB2 Information Integrator Data Source Configuration Guide*

Related tasks:

- "Adding Web services data sources to a federated server" in the *IBM DB2 Information Integrator Data Source Configuration Guide*

Part 2. Administering and maintaining

Chapter 2. Modifying data source configurations

Periodically, you need to make adjustments to the configuration that you initially established to access your data sources. For example, you might need to register a server definition for a new server that you want to access. You might need to change the user mapping between the federated database and the remote data source when the password on the remote data source changes. You might need to add a column option to a nickname to improve performance.

This chapter contains:

Altering a wrapper

After you configure a wrapper, you might want to modify the configuration based on your system requirements. Use the ALTER WRAPPER statement to:

- Add, set, or drop one or more wrapper options
- Set the environment, registry, or profile variables

Prerequisites:

The authorization ID associated with the statement must have SYSADM or DBADM authority.

Restrictions:

You cannot drop the DB2_FENCED wrapper option.

Procedure:

You can alter a wrapper from the DB2 Control Center or the DB2 command line.

To alter a wrapper from the DB2 Control Center:

1. Expand the Federated Database Objects folder. The wrapper objects are displayed in the contents pane of the DB2 Control Center window.
2. Right-click on the wrapper that you want to change and click **Alter** from the list of actions. The Alter Wrapper notebook opens.
   - On the Settings page, make the changes.
   - Click **Set Variables** to set the data source environment variables for the wrapper. Environment variables are not required for all wrappers.
3. Click **OK** to alter the wrapper and close the Alter Wrapper notebook.

To alter a wrapper from the DB2 command line, issue the ALTER WRAPPER statement.

Related tasks:

- "Installing DB2 Information Integrator fix packs" in the *IBM DB2 Information Integrator Data Source Configuration Guide*
- "Dropping a wrapper" on page 39

Related reference:

- "ALTER WRAPPER statement" in the *SQL Reference, Volume 2*
- "Altering a wrapper - examples" on page 26

Altering a wrapper - examples

To change the DB2_FENCED option to 'Y' for the wrapper named drda, issue the following statement:

```
ALTER WRAPPER drda OPTIONS (SET DB2_FENCED 'Y');
```

To change the MODULE option to '/opt/odbc/lib/libodbc.a(odbc.so)' for the wrapper named odbc , issue the following statement:

```
ALTER WRAPPER odbc OPTIONS (SET MODULE '/opt/odbc/lib/libodbc.a(odbc.so)');
```

Related tasks:

- "Altering a wrapper" on page 25

Altering server definitions and server options

A server definition identifies a data source to the federated database. The server definition consists of a local name and other information about that data source server. The server definition is used by the wrapper when SQL statements that use nicknames are submitted to the federated database. You 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 UDB uses the wrapper. Server options are specified as parameters in the CREATE SERVER and ALTER SERVER statements. Additionally, for relational data sources, server options can be set temporarily using the SET SERVER OPTION statement. This statement overrides the server option value in the server definition for the duration of a single connection to the federated database.

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 or to DRDA application servers.

Prerequisites:

The authorization ID issuing the ALTER SERVER statement must include either SYSADM or DBADM authority on the federated database.

Restrictions:

You cannot specify a wrapper in the ALTER SERVER statement that is not registered with the federated server.

The federated server cannot process an ALTER SERVER statement within a given unit of work (UOW) under either of the following conditions:

- The statement references a single data source, and the UOW already includes one of the following statements:
  - A SELECT statement that references a nickname for a table or view within the data source
  - An open cursor on a nickname for a table or view within the data source
  - An insert, delete or update issued against a nickname for a table or view within the data source
- The statement references a category of data sources (for example, all data sources of a specific type and version), and the UOW already includes one of the following statements:
  - A SELECT statement that references a nickname for a table or view within one of the data sources
  - An open cursor on a nickname for a table or view within one of the data sources
  - An insert, delete or update issued against a nickname for a table or view within one of the data sources

Modify a server definition when:

- You upgrade to a new version of the data source
- You want to make the same change to all of the server definitions for a specific data source type
- You want to add or change a server option on an existing server definition

Related tasks:

- "Altering the data source version in a server definition" on page 27
- "Dropping a server definition" on page 40

Related reference:

- Chapter 21, "Server options for federated systems," on page 219

Altering server definitions and server options-details

Use the ALTER SERVER statement to modify a server definition.

Altering the data source version in a server definition

You can alter an existing server definition to change the version of the data source that the remote server uses.

Prerequisites:

The authorization ID issuing the ALTER SERVER statement must include either SYSADM or DBADM authority on the federated database.

Procedure:

You can alter a server definition from the DB2 Control Center or the command line prompt.

To do this task from the DB2 Control Center:

1. Expand the Federated Database Objects folder. The server definition objects are displayed in the contents pane of the DB2 Control Center window.
2. Right-click on the server definition that you want to change and click **Alter** from the list of actions. The Alter Server Definition notebook opens.
3. On the Server page, click the **Version** arrow to specify a different version for the data source.
4. Click **OK** to alter the server definition and close the Alter Server Definition notebook.

To do this task from the command line prompt:

Issue the ALTER SERVER statement.

Suppose that you have a server definition for a Microsoft SQL Server Version 6.5 data source. The name that 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
```

Related tasks:

- "Altering all of the server definitions for a specific data source type" on page 28

Related reference:

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

Altering all of the server definitions for a specific data source type

You can alter all of the existing server definitions for a particular type of data source in a single ALTER SERVER statement. Use a single statement when you want the same change to apply to all of the server definitions of the same type.

Prerequisites:

The authorization ID issuing the ALTER SERVER statement must include either SYSADM or DBADM authority on the federated database.

Restrictions:

You can only set or drop server options using the ALTER SERVER statement for an entire type of data sources if the server options were added by a prior ALTER SERVER statement operation.

Procedure:

Suppose that there are five Sybase servers registered in the global catalog for your Sybase data sources. Whenever a user ID is sent to any of these Sybase servers for authentication, you want the federated server to always fold the user ID to

uppercase. In addition, you want to set how long the federated server will wait for a response from these Sybase servers to an SQL statement. You specify the amount of time in seconds.

You can modify all five server definitions at the same time using a single statement, such as:

```
ALTER SERVER TYPE sybase
   OPTIONS (ADD FOLD_ID 'U', ADD TIMEOUT '600')
```

Related tasks:

- "Altering the data source version in a server definition" on page 27

Related reference:

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

Using server options in server definitions

There are general server options and server options that are for specific data source types. You can alter a server definition to add or change a server option.

Server options are set to values that persist over successive connections to the data source. These values are stored in the federated system catalog.

Prerequisites:

The authorization ID issuing the ALTER SERVER statement must include either SYSADM or DBADM authority on the federated database.

Procedure:

You can alter a server definition from the DB2 Control Center or the command line prompt.

To do this task from the DB2 Control Center:

1. Expand the Federated Database Objects folder. The server definition objects are displayed in the contents pane of the DB2 Control Center window.
2. Right-click on the server definition that you want to change and click **Alter** from the list of actions. The Alter Server Definition notebook opens.
3. On the Settings page, select the server option that you want to add or remove.
4. For options that you are adding or changing, specify the value of an option.
5. Click **OK** to alter the server definition and close the Alter Server Definition notebook.

   Some server options are required and cannot be dropped. Other server options cannot be added if specific server options are already set. See the server options for federated systems for a list of descriptions for each of the options.

To do this task from the command line prompt:

Issue the ALTER SERVER statement. Examples of server options include:

- Suppose that you created server definition for an Informix server using the server name of INFMX01. You now want to change the DB2_MAXIMAL_PUSHDOWN option to Y. The statement to alter the server definition is:

```
ALTER SERVER INFMX01 OPTIONS (SET DB2_MAXIMAL_PUSHDOWN 'Y')
```

- Suppose that you created server definition for an Oracle server using the server name of ORCL99. You now want to add the FOLD_ID and FOLD_PW options to the definition. The statement to alter the server definition is:

  ```
  ALTER SERVER ORCL99 OPTIONS (ADD FOLD_ID 'U', FOLD_PW 'U')
  ```

- Suppose that you want to set the timeout value to the number of seconds the CTLIB wrapper should wait for a response from the Sybase server. You use the TIMEOUT server option to set this value. The statement to alter the server definition is:

  ```
  ALTER SERVER SYBSERVER OPTIONS (ADD TIMEOUT '60')
  ```

Changing server options temporarily for relational data sources

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

An example of the SET SERVER OPTION statement is:

```
SET SERVER OPTION PLAN_HINTS TO Y' FOR SERVER ORA_SERVER
```

When used with static SQL statements, the SET SERVER OPTION statement will have no effect with the IUD_APP_SVPT_ ENFORCE server option.

The hierarchy of server option settings

When you have the same server option set with one value for a data source type and set with another value on a specific data source server, there is a hierarchy among the settings. For example, suppose the PLAN_HINTS server option is set to 'Y' for the data source type ORACLE. However, the PLAN_HINTS server option is set to 'N' in the server definition for a specific Oracle data source server PURNELL. The setting for the specific data source server overrides the setting for the data source type. This configuration causes PLAN_HINTS to be enabled at all Oracle data source servers except PURNELL.

Related concepts:

- "Server definitions and server options" on page 13

Related reference:

- Chapter 21, "Server options for federated systems," on page 219
- "ALTER SERVER statement" in the *SQL Reference, Volume 2*
- "SET SERVER OPTION statement" in the *SQL Reference, Volume 2*

Altering a user mapping

A user mapping is 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.

The ALTER USER MAPPING statement is used to change the authorization ID or password that is used at the data source for a specific federated server authorization ID.

Prerequisites:

If the authorization ID issuing the statement is different than the authorization ID that is mapped to the data source, then the authorization ID issuing the statement must include either SYSADM or DBADM authority on the federated database.

Restrictions:

The federated server cannot process an ALTER USER MAPPING statement within a given unit of work (UOW) if the UOW already includes one of the following statements:

- A SELECT statement that references a nickname for a table or view at the data source the mapping includes
- An open cursor on a nickname for a table or view at the data source that the mapping includes
- An insert, delete, or update issued for a nickname of a table or view at the data source the mapping includes
-

Procedure:

You can alter a user mapping from the DB2 Control Center or the command line prompt.

To do this task from the DB2 Control Center:

1. Expand the Federated Database Objects folder. The user mapping objects are displayed in the contents pane of the DB2 Control Center window.
2. Right-click on the user mapping that you want to change and click **Alter** from the list of actions. The Alter User Mapping window opens.
3. Change the value of the option.
4. Click **OK** to alter the user mapping and close the Alter User Mapping window.

To do this task from the command line prompt:

Issue the ALTER USER MAPPING statement.

For example, Jenny uses the federated server to connect to a Sybase server called SYBSERVER. She accesses the federated server with the authorization ID of *jennifer*. The authorization ID *jennifer* is mapped to the authorization ID *jenn* on the Sybase server. The authorization ID for Jenny on the Sybase server is changed to *jen123*. The ALTER USER MAPPING statement to map *jennifer* to *jen123* is:

```
ALTER USER MAPPING FOR jennifer SERVER SYBSERVER
   OPTIONS (SET REMOTE_AUTHID 'jen123')
```

Tomas uses the federated server to connect to an Oracle server called ORASERVER. He accesses the federated server with the authorization ID of *tomas*. The authorization ID *tomas* is mapped to the authorization ID *tom* on the Oracle server. The password for Tomas on the Oracle server is changed. His new password is day2night. The ALTER USER MAPPING statement to map *tomas* to the new password is:

```
ALTER USER MAPPING FOR tomas SERVER ORASERVER
   OPTIONS (SET REMOTE_PASSWORD 'day2night')
```

The REMOTE_AUTHID and REMOTE_PASSWORD user options are case sensitive unless you set the FOLD_ID and FOLD_PW server options to 'U' or 'L' in the CREATE SERVER statement.

Related reference:

- "ALTER USER MAPPING statement" in the *SQL Reference, Volume 2*
- Chapter 22, "User mapping options for federated systems," on page 235

Altering a nickname

Nicknames are identifiers that are used to reference an object that you want to access at a data source.

You might want to alter a nickname to:

- Alter the local column names for the columns of the data source object
- Alter the local data types for the columns of the data source object
- Add, set, or drop nickname and column options
- Add or drop a primary key
- Add or drop one or more unique, referential, or check constraints
- Alter one or more referential, check, or functional dependency constraint attributes

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:

See the topic on restrictions to altering nicknames.

Procedure:

You can alter a nickname from the DB2 Control Center or the DB2 command line.

To do this task from the DB2 Control Center:

1. Select the **Nicknames** folder.
2. Right-click on the nickname that you want to change and click **Alter**. The Alter Nickname notebook opens.
3. On the Nicknames page, change the local column names, local data types, or column options for the columns that are stored in the global catalog.
4. On the Keys page, set the referential integrity constraints for the nickname. You can set a primary key, unique key, or foreign key constraint.
5. On the Check Constraints page, set the check constraints or functional dependency constraints for the nickname.
6. On the Settings page, set the nickname options for the nickname.
7. Click **OK** to alter the nickname and close the notebook.

Some nickname options are required and cannot be dropped. Other nickname options cannot be added if specific nickname options are already set. See the nickname options for federated systems and the nickname column options for federated systems for a list of descriptions for each of the options.

To do this task from the DB2 command line, issue the ALTER NICKNAME statement with the appropriate parameters set.

When the data source object structure or content changes significantly, you should update the nickname statistics. Significant changes include the addition or removal of multiple rows.

Related concepts:

- "Informational constraints on nicknames" on page 181
- "Nickname statistics update facility - overview" on page 187

Related tasks:

- "Altering nickname options" on page 36
- "Altering a local type for a data source object" on page 52
- "Altering nickname column names" on page 35
- "Altering nickname column options" on page 37

Related reference:

- "Restrictions on altering nicknames" on page 33
- Chapter 23, "Nickname options for federated systems," on page 237
- Chapter 24, "Nickname column options for federated systems," on page 247
- "ALTER NICKNAME statement" in the *SQL Reference, Volume 2*

Altering nicknames-details

You can change the data source column names that are stored in the global catalog and set column options by altering the nicknames.

Restrictions on altering nicknames

The following restrictions apply when you alter a nickname.

Column names

The ALTER NICKNAME statement cannot be used to alter column names for the following data sources. You must drop the nickname and create the nickname again with the correct column names.

- BLAST
- Documentum
- HMMER

Column options

If one of the following options is set on a column, you cannot add any other options to that column:

- SOAPACTIONCOLUMN
- URLCOLUMN
- PRIMARY_KEY
- FOREIGN_KEY

For BioRS

- If you change the element name of a column by using the ELEMENT_NAME option, the new name is not checked to ensure that it is correct. An incorrect option might result in errors when the column is referenced in a query.
- If you make changes to the IS_INDEXED column option, the changes are not verified with the BioRS server. An incorrect option might result in errors when the column is referenced in a query.

Data types

- If you change the data type of a column, the new data type must be compatible with the data type of the corresponding data source column or element. Changing the local data type to a data type that is incompatible with the remote data type might cause unpredictable errors.
- 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.
- You cannot override the existing local types or create new local types for some of the nonrelational data sources. Check the documentation for the specific data source wrapper for more information on this restriction.
- When the local specification of a column's data type is changed, the federated database manager invalidates any statistics (for example, HIGH2KEY and LOW2KEY) that are gathered for that column.
- The local type is set for the specific data source object when it is accessed with that nickname. The same data source object can have another nickname that uses the default data type mapping.

Indexes

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 an index specification.

LOCAL NAME and LOCAL TYPE parameters

- ALTER NICKNAME statement cannot be used to change the local names or data types for the columns in the nickname if:
  - The nickname is used in a view, SQL method, or SQL function
  - You define an informational constraint on the nickname
- 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..

Nicknames

The ALTER NICKNAME statement cannot be used to change the name of the BioRS databank that is referenced by or used in a BioRS nickname. If the name of a BioRS databank changes, you must drop the nickname and create the nickname again.

Units of work

The federated server cannot process an ALTER NICKNAME statement within a given unit of work under either of the following conditions:

- If the nickname referenced in the ALTER NICKNAME statement has a cursor open on it in the same units of work.
- If an insert, delete or update is issued in the same unit of work for the nickname that is referenced in the ALTER NICKNAME statement.

- For nonrelational data sources, if the ALTER NICKNAME statement references a nickname that is referenced by a SELECT statement in the same unit of work.

Related tasks:

- "Altering nickname options" on page 36
- "Altering a local type for a data source object" on page 52
- "Altering a nickname" on page 32
- "Altering nickname column names" on page 35
- "Altering nickname column options" on page 37

Altering nickname column names

When you create a nickname, the column names that are associated with the data source object are stored in the federated database. For some data sources, the wrapper specifies the column names. For other data sources, you must specify the column names when you create the nickname.

You can alter a nickname to change the column names.

Prerequisites:

The authorization ID issuing the statement must include at least one of the following privileges:

- 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:

See the topic on restrictions to altering nicknames.

Procedure:

You can change column names from the DB2 Control Center or the DB2 command line.

To do this task from the DB2 Control Center:

1. Select the **Nicknames** folder.
2. Right-click on the nickname that you want to change and click **Alter**. The Alter Nickname notebook opens.
3. On the Nicknames page, select the column that you want to change and click **Change**. The Change Column window opens.
4. Type the column name.
5. Click **OK** to change the column name and close the window.
6. Click **OK** to alter the nickname and close the notebook.

To do this task from the DB2 command line, issue the ALTER NICKNAME statement:

```
ALTER NICKNAME nickname
   ALTER COLUMN current_name
   LOCAL NAME new_name
```

Example: Changing the local name of a nickname column:

For example, the nickname Z_EMPLOYEES for a DB2 UDB for z/OS table includes a column with the name of EMPNO. To alter the nickname so that the local column name that users work with is *Employee_Number* instead of *EMPNO*, issue the following statement:

```
ALTER NICKNAME Z_EMPLOYEES ALTER COLUMN EMPNO
   LOCAL NAME "Employee_Number"
```

Related tasks:

- "Altering a nickname" on page 32

Related reference:

- "Restrictions on altering nicknames" on page 33
- "ALTER NICKNAME statement" in the *SQL Reference, Volume 2*

Altering nickname options

Nickname options are parameters that you specify on the nickname when you issue the CREATE NICKNAME and ALTER NICKNAME statements.

You can add, set, or drop nickname options by using the ALTER NICKNAME statement.

Prerequisites:

The authorization ID issuing the statement must include at least one of the following privileges:

- 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:

See the topic on restrictions to altering nicknames.

Procedure:

You can change column names from the DB2 Control Center or the DB2 command line.

To do this task from the DB2 Control Center:

1. Select the **Nicknames** folder.
2. Right-click on the nickname that you want to change and click **Alter**. The Alter Nickname notebook opens.

3. On the Settings page, select the check box next to any option that you want to add or remove. You cannot remove a required option.
4. To specify or change the value of an option, click the **Value** field for the option. Depending on the option, you can either select a value from the list, click to select multiple values, or you can type a new value.
5. Click **OK** to alter the nickname and close the notebook.

To do this task from the command line prompt, use the ALTER NICKNAME statement. For example:

```
ALTER NICKNAME nickname
    OPTIONS (SET option_name 'option_string_value')
```

For example, the nickname DRUGDATA1 is created for the table-structured file `drugdata1.txt`. The fully qualified path that was originally defined in the CREATE NICKNAME statement was `/user/pat/drugdata1.txt`.

To change the FILE_PATH nickname option, issue the following statement :

```
ALTER NICKNAME DRUGDATA1 OPTIONS (SET FILE_PATH '/usr/kelly/data/drugdata1.txt')
```

Related tasks:

- "Altering a nickname" on page 32

Related reference:

- "Restrictions on altering nicknames" on page 33
- "ALTER NICKNAME statement" in the *SQL Reference, Volume 2*
- Chapter 23, "Nickname options for federated systems," on page 237

Altering nickname column options

You specify column information in the CREATE NICKNAME and ALTER NICKNAME statements by using parameters called *nickname column options.* You can specify any of these values in either uppercase or lowercase letters.

You can add, set, or drop nickname column options using the ALTER NICKNAME statement.

Prerequisites:

The authorization ID issuing the statement must include at least one of the following privileges:

- 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:

See the topic on restrictions to altering nicknames.

Procedure:

You can change column names from the DB2 Control Center or the DB2 command line.

To do this task from the DB2 Control Center:

1. Select the **Nicknames** folder.
2. Right-click on the nickname that you want to change and click **Alter**. The Alter Nickname notebook opens.
3. On the Nicknames page, select the column that you want to change and click **Change**. The Change Column window opens.
4. Select the column option that you want to add or remove.
5. For options that you are adding or changing, specify the value of an option.
6. Click **OK** to change the column option and close the window.
7. Click **OK** to alter the nickname and close the notebook.

To do this task from the command line prompt, use the ALTER NICKNAME statement.

Example 1: Specifying the NUMERIC_STRING column option with relational data sources:

The NUMERIC_STRING 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 UDB query optimizer the option of performing the sort at the data source. If the sort is performed remotely, you can avoid the overhead of sorting the data at the federated server.

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 characters. To change the NUMERIC_STRING column option to 'N', use this statement:

```
ALTER NICKNAME ORA_INDSALES ALTER COLUMN POSTAL_CODE
   OPTIONS (SET NUMERIC_STRING 'N')
```

Example 2: Specfying the VARCHAR_NO_TRAILING_BLANKS column option with relational data sources:

The VARCHAR_NO_TRAILING_BLANKS column option can be used to identity specific columns that contain no trailing blanks. The SQL Compiler will factor in this setting when it checks for all operations (such as comparison operations) performed on columns.

The nickname ORA_INDSALES is 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, use this statement:

```
ALTER NICKNAME ORA_INDSALES ALTER COLUMN NAME
   OPTIONS (ADD VARCHAR_NO_TRAILING_BLANKS 'Y')
```

Example 3: Specifying the XPATH column option with nonrelational data sources:

The nickname EMPLOYEE is for an XML data source. An XPATH was specified for the *fname* column. To set the XPATH column option to a different path, use this statement:

```
ALTER NICKNAME EMPLOYEE ALTER COLUMN fname
   OPTIONS (SET XPATH './@first')
```

Related tasks:

- "Altering a nickname" on page 32

Related reference:

- "Restrictions on altering nicknames" on page 33
- "ALTER NICKNAME statement" in the *SQL Reference, Volume 2*
- Chapter 24, "Nickname column options for federated systems," on page 247

Dropping a wrapper

There are several reasons why you might want to drop a wrapper.

Sometimes there is more than one wrapper that you can use to access a data source. The one you choose might depend on the version of the data source client software that you are using. Or it might depend on the operating system that 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 created the SQLNET wrapper. Since DB2 Information Integrator no longer supports the SQLNET wrapper, you can drop the SQLNET wrapper and create the NET8 wrapper.

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 Informix 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 you can drop it.

Attention: There are serious consequences when you drop a wrapper. Other objects that you registered with the federated server are impacted:

- All server definitions, user-defined functions mappings, and user-defined data type mappings that are dependent upon the wrapper drop are also dropped.
- Dropping all server definitions dependent on that wrapper, affects the objects dependent on those server definitions. All user-defined function mappings, user-defined data type mappings, and user mappings that are dependent on the dropped server definitions are also dropped.
- All nicknames that are dependent on the dropped server definitions are also dropped. Dropping the nicknames dependent on the server definitions, affects the objects dependent on those nicknames:
  - Any index specifications dependent on the dropped nicknames are also dropped.

  - Any federated views dependent on the dropped nicknames are marked inoperative.
  - Any materialized query tables dependent on the dropped nicknames are also dropped.
- All applications dependent on the dropped objects and inoperative views are invalidated.

Prerequisites:

To issue the DROP WRAPPER statement, you must have SYSADM or DBADM authority.

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 reference:

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

Dropping a server definition

Dropping a server definition deletes the definition from the global catalog, the data source object that the server definition references is not affected.

When you drop a server definition, other objects that you registered with the federated server are impacted:

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

Use the DROP statement to delete a server definition.

Prerequisites:

You must have SYSADM or DBADM authority to drop a server definition.

Restrictions:

The federated server cannot process a DROP SERVER statement within a given unit of work (UOW) under either of the following conditions:

- The statement references a single data source, and the UOW already includes one of the following statements:

  - A SELECT statement that references a nickname for a table or view within the data source
  - An open cursor on a nickname for a table or view within the data source
  - An insert, delete or update issued against a nickname for a table or view within the data source
- The statement references a category of data sources (for example, all data sources of a specific type and version), and the UOW includes one of the following statements:
  - A SELECT statement that references a nickname for a table or view within one of the data sources
  - An open cursor on a nickname for a table or view within one of the data sources
  - An insert, delete or update issued against a nickname for a table or view within one of the data sources

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 server_name
```

where *server_name* 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
```

Related tasks:

- "Altering server definitions and server options" on page 26

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*

Dropping a user mapping

When a user no longer needs access to a remote data source, drop the user mapping between the federated server and the remote data source server.

If the user is mapped to more than one data source server, you will need to drop each mapping separately.

Prerequisites:

To issue the DROP USER MAPPING statement, the authorization ID of the DROP statement must have SYSADM or DBADM authority, if this authorization ID is different from the federated database user ID specified in the user mapping. Otherwise, if the authorization ID and the user ID in the user mapping match, no authorities or privileges are required.

Procedure:

To drop a user mapping, use the DROP statement:

```
DROP USER MAPPING FOR user_ID SERVER local_server_name
```

The *user_ID* is the authorization ID for the user on the federated server. The *local_server_name* is the local name that is used to identify the remote data source server in the server definition.

Related tasks:

- "Altering a user mapping" on page 30

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*

Dropping a nickname

There are several reasons why you might want to drop a nickname. For example:

- If the underlying data source object structure or content is 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.

Dropping a nickname deletes the nickname from the global catalog on the federated server. The data source object that the nickname references is not affected.

When you drop a nickname, other objects that you registered with the federated server are impacted:

- Dropping a nickname affects the objects dependent on those nicknames:
  - Any index specifications dependent on the dropped nicknames are also dropped.
  - Any federated views dependent on the dropped nicknames are marked inoperative.
- All applications dependent on the dropped objects and inoperative views are invalidated.

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

Restrictions:

For nicknames to relational data sources, the federated server cannot process the DROP NICKNAME statement within a given unit of work (UOW) under either of the following conditions:

- A nickname referenced in the statement has a cursor open on it in the same UOW.
- An insert, delete or update is issued in the same UOW for the nickname referenced in the statement.

For nicknames to nonrelational data sources, the federated server cannot process the DROP NICKNAME statement within a given unit of work (UOW) under any of the following conditions:

- A nickname referenced in this statement has a cursor open on it in the same UOW.
- A nickname referenced in this statement is already referenced by a SELECT statement in the same UOW.
- An insert, delete or update is issued in the same UOW for the nickname referenced in this statement.

Procedure:

To drop a nickname, the syntax is:

```
DROP NICKNAME nickname
```

where *nickname* identifies the nickname to be dropped.

Related tasks:

- "Altering a nickname" on page 32

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*

Chapter 3. Data type mappings

The wrappers that are included with DB2 Information Integrator contain default data type mappings between the data sources and DB2 Universal Database for Linux, UNIX, and Windows.

This chapter contains:

Data type mappings in a federated system

The data types at a data source must map to corresponding DB2® data types. This mapping enables the federated server to retrieve data from the data source.

DB2 Information Integrator supplies a set of default data type mappings for some data sources. For other data sources you must provide the data type mappings that you want to use. For nonrelational data sources, you cannot override the existing data type mappings or create new mappings.

Some examples of default data type mappings are:

- The Oracle type FLOAT maps by default to the DB2 type DOUBLE
- The Oracle type DATE maps by default to the DB2 type TIMESTAMP
- The DB2 Universal Database™ for z/OSand OS/390® type DATE maps by default to the DB2 type DATE

Nicknames that are created after a mapping is changed use the new type mapping. Nicknames that are created before the mapping is changed use the default data type mapping.

If you already created the nicknames, you can update the existing nicknames in one of the following ways:

- You can alter each nickname
- You can drop and re-create each nickname

DB2 federated servers do not support mappings for the following data types:

- The local data type cannot be LONG VARCHAR, LONG VARGRAPHIC, DATALINK, or a user-defined data type.
- The remote data type cannot be a user-defined type.

  However, you can use a cast function to convert the user-defined data type in a view at the remote data source that is identical to the data source to a built-in or system data type. You can then create a nickname for the view. If you create such views, the views have no statistics or indexes and you cannot update the views.

Related concepts:

- "Data type mappings and the federated database global catalog" on page 46
- "When to create alternative data type mappings" on page 47
- "Data type mappings for nonrelational data sources" on page 47
- "Forward and reverse data type mappings" on page 48

Related tasks:

- "Creating data type mappings" on page 48

Related reference:

- Chapter 27, "Default forward data type mappings," on page 261
- Chapter 28, "Default reverse data type mappings," on page 277
- "Altering long data types to varchar data types" on page 55

Data type mappings and the federated database global catalog

When you write a CREATE NICKNAME statement, you specify a data source object that the nickname represents. In most cases, the federated server defines a DB2®-supported data type for each column or field in that data source object. For some nonrelational data sources, you must supply the DB2 data type. These local data type definitions are stored in the SYSCAT.COLUMNS catalog view of the federated database global catalog.

For relational data sources, to determine which local data type to store in the SYSCAT.COLUMNS catalog view, the federated server looks for forward data type mapping information in the wrappers and in the SYSCAT.TYPEMAPPINGS catalog view. Mappings in the SYSCAT.TYPEMAPPINGS catalog view take precedence over the default mappings in the wrappers. If you create alternative mappings to override the default data type mappings, the federated server uses the alternative mappings. If multiple mappings apply to a column, the federated server uses the most recently created mappings.

For nonrelational data sources, to determine which local data type to store in the SYSCAT.COLUMNS catalog view, the federated server looks for data type mapping information in the wrappers. Depending on the nonrelational data source, the degree to which you can modify the data types defined by the wrapper varies. For some nonrelational data sources, you do not specify any columns. The wrapper defines the data types. For other data sources you can override the data types. And for other data sources, you must specify the column data types on the CREATE NICKNAME statement.

When you write CREATE TABLE transparent DDL for relational data sources, specify DB2 data types in the statement. The federated server checks for

information about the reverse data type mappings between DB2 UDB and the data source. The federated server looks for this information in the wrapper and the SYSCAT.TYPEMAPPINGS catalog view.

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 mapping is a default mapping, the values also conform fully to the data source type in the mapping. For example, if 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. The values that are returned from the column conform fully to both the FLOAT and DOUBLE data types.

Related concepts:

- "Data type mappings in a federated system" on page 45

When to create alternative data type mappings

You can create alternative data type mappings for relational data sources.

You might want to create alternative data type mappings in the following situations:

- To override a default data type mapping

  For some wrappers, you can change the format or length of values that are returned. You can change the format or length by changing the DB2® data type that the values must conform to. For example, the Oracle DATE data type is used as a timestamp and contains the century, year, month, day, hour, minute, and second. By default, the Oracle DATE data type maps to the DB2 TIMESTAMP data type. To return only the hour, minute, and second information, you can override the default data type mapping so that the Oracle DATE data type maps to the DB2 TIME data type. When the Oracle DATE columns are queried, only the time portion of the Oracle timestamp values is returned to the federated server.
- When a default mapping does not exist

  If a default data type mapping is not available for a data source data type, you must create a mapping for the new data type.

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

Related concepts:

- "Data type mappings in a federated system" on page 45

Related tasks:

- "Creating data type mappings" on page 48

Data type mappings for nonrelational data sources

For some nonrelational data sources, the data type mappings are not in the wrappers. In some cases, you must specify the local type information in the CREATE NICKNAME statement.

The following example shows how column data types are specified in the CREATE NICKNAME statement for some of the nonrelational data sources:

```
CREATE NICKNAME DRUGDATA1
   (Dcode Integer NOT NULL, Drug CHAR(20), Manufacturer CHAR(20))
   FOR SERVER biochem_lab
   OPTIONS (FILE_PATH '/usr/pat/DRUGDATA1.TXT', COLUMN_DELIMITER ',',
   SORTED 'Y', KEY_COLUMN 'DCODE', VALIDATE_DATA_FILE 'Y')
```

Related concepts:

- "Data type mappings in a federated system" on page 45

Related tasks:

- "Registering nicknames for a data source" in the *IBM DB2 Information Integrator Data Source Configuration Guide*

Forward and reverse data type mappings

A *forward type mapping* is a mapping from a remote data type to a comparable local data type. Forward type mappings are used when you a create a nickname for a data source object. The comparable local type for each column in the data source object is stored in the global catalog.

A *reverse type mapping* is a mapping from a local data type to a comparable remote data type. Reverse type mapping is used with transparent DDL.

Figure 2 shows forward and reverse data type mapping.

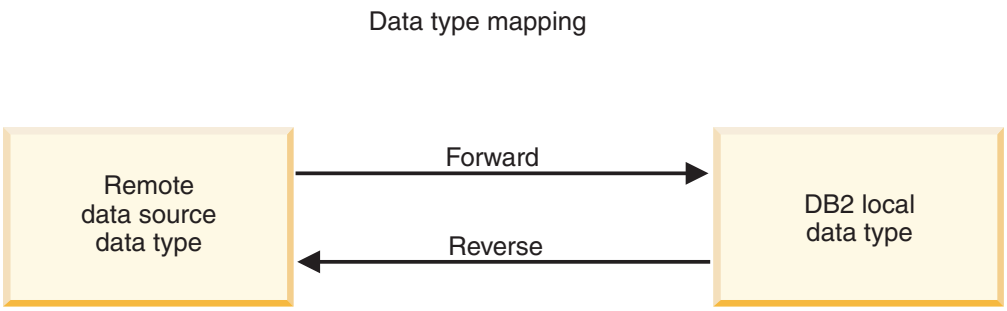


Figure 2. Forward and reverse data type mappings

Related concepts:

- "Data type mappings in a federated system" on page 45

Related reference:

- Chapter 27, "Default forward data type mappings," on page 261
- Chapter 28, "Default reverse data type mappings," on page 277

Creating data type mappings

To create a data type mapping you use the CREATE TYPE MAPPING statement. You can issue it from the DB2® Command Center or the command line processor, or include it in an application program. You cannot use the DB2 Control Center to create or modify data type mappings.

Prerequisites:

The privileges held by the authorization ID associated with the statement must have SYSADM or DBADM authority.

Restrictions:

- The *local_data_type* value cannot be long varchar, long vargraphic, DATALINK, or a user-defined data type.
- The *data_source_data_type* value cannot be a user-defined type.
- For nonrelational data sources, you cannot override existing data type mappings or create mappings.

Procedure:

To create a data type mapping, issue the CREATE TYPE MAPPING statement.

Related concepts:

- "Data type mappings in a federated system" on page 45

Related reference:

- Chapter 27, "Default forward data type mappings," on page 261
- Chapter 28, "Default reverse data type mappings," on page 277
- "Creating a data type mapping for a data source data type – example" on page 49
- "Creating a type mapping for a data source data type and version – example" on page 50
- "Creating a type mapping for all data source objects on a server – example" on page 51
- "Altering a local type for a data source object – examples" on page 54
- "Altering long data types to varchar data types" on page 55
- "Plan the data type mappings" in the *IBM DB2 Information Integrator Data Source Configuration Guide*

Creating a data type mapping for a data source data type – example

In this example, all Oracle tables and views that use the Oracle NUMBER data type must map to the DB2 DECIMAL(8,2) data type. The Oracle NUMBER data type is mapped by default to the DB2 DOUBLE data type, a floating decimal data type.

Use the ALTER NICKNAME statement to change the local types of existing nicknames. You must modify each nickname separately to change the local data type to DECIMAL(8,2).

If the nicknames do not exist, create a data type mapping that specifies the data source type.

For example, to create the type mapping from Oracle NUMBER data type to the DB2 DECIMAL(8,2) data type, issue the following statement:

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

MY_ORACLE_DEC
The name that you give to the type mapping. The name cannot duplicate a data type mapping name that already exists in the catalog.

FROM *SYSIBM.DECIMAL(8,2)*
The local DB2 schema and the local 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*
The type of data source.

TYPE *NUMBER*
The data source data type that you are mapping to the local data type. User-defined data types are not allowed.

The DB2 DECIMAL(8,2) data type is defined locally for the Oracle columns.

All new Oracle tables and views that contain NUMBER columns also map the Oracle NUMBER data type to the DB2 DECIMAL(8,2) data type when you create the nicknames for those tables and views.

Related tasks:

- "Creating data type mappings" on page 48

Related reference:

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

Creating a type mapping for a data source data type and version – example

In this example, Oracle tables and views exist on different versions of the Oracle server. For all tables and views on Oracle Version 8.0.3 servers, columns that use the Oracle NUMBER(23,3) data type must map to the DB2 DECIMAL(8,2) data type. The Oracle NUMBER(23,3) data type is mapped by default to the DB2 DECIMAL(23,3) data type.

Use the ALTER NICKNAME statement to change the local types of existing nicknames. You must modify each nickname separately to change the local data type to DECIMAL(8,2).

If the nicknames do not exist, create a data type mapping that specifies the data source type.

For example, to map the Oracle NUMBER(23,3) data type to the DB2 DECIMAL(8,2) data type for Oracle servers using Version 8.0.3, issue the following statement:

```
CREATE TYPE MAPPING ORA_DEC FROM SYSIBM.DECIMAL(8,2)
    TO SERVER TYPE ORACLE VERSION 8.0.3 TYPE NUMBER(23,3)
```

ORA_DEC
The name that you give to the type mapping. The name cannot duplicate a data type mapping name that already exists in the catalog.

FROM *SYSIBM.DECIMAL(8,2)*
The local DB2 schema and the local 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*
The type of data source.

VERSION *8.0.3*
The version of data source server. You must specify the version. You can also specify the release and the modification of the release, as shown in this example.

TYPE *NUMBER(23,3)*
The data source data type that you are mapping to the local data type. User-defined data types are not allowed.

DB2 UDB defines the DB2 DECIMAL(8,2) data type locally for the Oracle columns on Version 8.0.3 servers.

All new Oracle tables and views on Oracle Version 8.0.3 servers that contain NUMBER(23,3) columns also map the Oracle NUMBER(23,3) data type to the DB2 DECIMAL(8,2) data type when you create the nicknames for those tables and views.

Oracle tables and views on servers that do not use Version 8.0.3 use the default data type mapping.

Related tasks:

- "Creating data type mappings" on page 48

Related reference:

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

Creating a type mapping for all data source objects on a server – example

In this example, the server is defined to the federated database as ORA2SERVER. Each table contains a column with an Oracle DATE data type. The Oracle DATE data type contains the century, year, month, day, hour, minute, and second. The Oracle DATE data type is mapped by default to the local DB2 TIMESTAMP data type. However, when you query any object on this server, the result set must return only the time information (hour, minute, and second).

Use the ALTER NICKNAME statement to change the local types of existing nicknames. You must modify each nickname separately to change the local data type to TIME.

If the nicknames do not exist, create a data type mapping that specifies the data source type.

To map the Oracle DATE data type to the DB2 TIME data type for the ORA2SERVER, issue the following statement:

```
CREATE TYPE MAPPING ORA2_DATE FROM SYSIBM.TIME
   TO SERVER ORA2SERVER TYPE DATE
```

ORA2_DATE
The name that you give to the type mapping. The name cannot duplicate a data type mapping name that already exists in the catalog.

FROM *SYSIBM.TIME*
The local DB2 schema and the local data type. If the length or precision and scale are not specified, then these values are determined from the source data type.

TO SERVER *ORA2SERVER*
The local name of the data source server.

TYPE *DATE*
The data source data type that you are mapping to the local data type. User-defined data types are not allowed.

DB2 UDB locally defines the DB2 TIME data type for the Oracle columns of data type DATE.

Any new objects that are added to this server that contain DATE columns also map the Oracle DATE data type to the DB2 TIME data type when you create the nicknames for those objects.

Data source objects on other Oracle servers are not affected by this data type mapping.

Related tasks:

- "Creating data type mappings" on page 48

Related reference:

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

Altering a local type for a data source object

When you create a nickname, the data types that are associated with the data source object are stored in the federated database. For some data sources, the wrapper specifies the data types for you. For other data sources, you must specify the data types when you create the nickname.

You can specify a local type for a column of a specific data source object. You use the ALTER NICKNAME statement instead of the CREATE TYPE MAPPING statement.

Attention: Changing the local data type can result in errors or loss of information if you change the local data type for a column to a type that differs greatly from its remote type.

Prerequisites:

The authorization ID issuing the statement must include at least one of the following privileges:

- 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

The authorization ID associated with the statement must be the definer of the nickname as recorded in the DEFINER column of the catalog view for the nickname.

Restrictions:

See the topic on restrictions to altering nicknames.

Procedure:

You can change the data type from the DB2 Control Center or the DB2 command line.

To do this task from the DB2 Control Center:

1. Select the **Nicknames** folder.
2. Right-click on the nickname that you want to change and click **Alter**. The Alter Nickname notebook opens.
3. On the Nicknames page, select the column that you want to change and click **Change**. The Change Column window opens.
4. Select the data type.
5. Click **OK** to change the data type and close the window.
6. Click **OK** to alter the nickname and close the notebook.

To do this task from the command line prompt, use the ALTER NICKNAME statement. For example:

```
ALTER NICKNAME nickname ALTER COLUMN column_name
   LOCAL TYPE data_type
```

To treat the contents of a local column that has a character data type as bit (binary) data, use the FOR BIT DATA clause in the ALTER NICKNAME statement. When you use this clause to change the local data type of a column, code page conversions are not performed when data is exchanged with other systems. Comparisons are done in binary, irrespective of the remote database collating sequence.

Related tasks:

- "Altering a nickname" on page 32

Related reference:

- "Restrictions on altering nicknames" on page 33
- "ALTER NICKNAME statement" in the *SQL Reference, Volume 2*
- Chapter 27, "Default forward data type mappings," on page 261
- "Altering a local type for a data source object – examples" on page 54
- Chapter 30, "Data types supported for nonrelational data sources," on page 291

Altering a local type for a data source object – examples

The following examples show how to change the data types for a data source object.

Example: A numeric data type mapping:

In an Oracle table for employee information, the BONUS column is defined with a data type of NUMBER(32,3). The Oracle data type NUMBER(32,3) is mapped by default to the DB2 data type DOUBLE, a double-precision floating-point number data type. A query that includes the BONUS column might return values that look like this:

```
5.0000000000000E+002
1.0000000000000E+003
```

The scientific notation indicates the number of decimal places and the direction that the decimal point should be moved. In this example +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.

Queries that include the BONUS column can return values that look like dollar amounts. You change the local definition for the BONUS column in the table from the DOUBLE data type to DECIMAL data type. Use a precision and scale that reflect the format of actual bonuses. For example, if the dollar portion of the bonuses would not exceed six figures, map NUMBER(32,3) to DECIMAL(8,2). Under the constraint of this new local type, queries that include the BONUS column return values like this:

```
 500.00
1000.00
```

The nickname for the Oracle table is ORASALES. To map the BONUS column in the ORASALES table to the DB2 DECIMAL (8,2) data type, issue the following ALTER NICKNAME statement:

```
ALTER NICKNAME ORASALES ALTER COLUMN BONUS
   LOCAL TYPE DECIMAL(8,2)
```

ORASALES
The nickname that you defined for the Oracle table.

ALTER COLUMN *BONUS*
The name of the column that is defined locally in the federated database SYSCAT.COLUMNS catalog view.

LOCAL TYPE *DECIMAL(8,2)*
Identifies the new local type for the column.

This mapping applies only to the BONUS column in the Oracle table that is identified by the nickname ORASALES. All other Oracle data source objects that include the BONUS column use the default data type mapping for the Oracle NUMBER data type.

Example: A date data type mapping:

The nickname for an Oracle table named SALES is ORASALES. The SALES 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

want to display only the date value when you retrieve data from this column. You can alter the nickname for the SALES table to change the local type to the DB2 DATE data type.

```
ALTER NICKNAME ORASALES ALTER COLUMN ORDER_DATE
   LOCAL TYPE DATE
```

Example: A data type mapping for a nonrelational data source:

The nickname for a table-structured file named `drugdata1.txt` is DRUGDATA1. The `drugdata1.txt` file contains a column that lists pharmaceutical drug names. The column name is DRUG. The DRUG column was originally defined as a CHAR(20). The length of the column must be changed to CHAR(30). You can alter the nickname for the `drugdata1.txt` file to change the mapping to the correct length:

```
ALTER NICKNAME DRUGDATA1 ALTER COLUMN DRUG
   LOCAL TYPE CHAR(30)
```

Related tasks:

- "Creating data type mappings" on page 48
- "Altering a local type for a data source object" on page 52

Related reference:

- "ALTER NICKNAME statement" in the *SQL Reference, Volume 2*
- "Restrictions on altering nicknames" on page 33

Altering long data types to varchar data types

To enable insert and update operations for long data types, you can alter the long data types to the VARCHAR data type. Table 4 lists the long data type by data source that you can alter.

Table 4. Long data types by data source that can be altered to the varchar data type

Data source	Remote data type	Length	Local default data type	ALTER to VARCHAR
DRDA	BLOB	1–32672	BLOB	varchar for bit data
	CLOB	1–32672	CLOB	varchar
	long varchar	1–32672	CLOB	varchar
	long varchar for bit data	1–32672	BLOB	varchar for bit data
Informix	byte	1–32672	BLOB	varchar for bit data
	text	1–32672	CLOB	varchar
Microsoft SQL Server	image	1–32672 host vars; 1–8000 literal	BLOB	varchar for bit data
	text	1–32672 host vars; 1–8000 literal	CLOB	varchar
Oracle NET8	long	1–32672 host vars; 1–4000 literal	CLOB	varchar

Table 4. Long data types by data source that can be altered to the varchar data type (continued)

Data source	Remote data type	Length	Local default data type	ALTER to VARCHAR
	long raw	1–32672 host vars; 1–4000 literal	BLOB	varchar for bit data
Sybase CTLIB	image	1–32672	BLOB	varchar for bit data
	text	1–32672	CLOB	varchar
Teradata	byte	32673–64000	BLOB	varchar for bit data(32672)
	CHAR	32673–64000	CLOB	varchar(32672)
	varbyte	32673–64000	BLOB	varchar for bit data(32672)
	varchar	32673–64000	CLOB	varchar(32672)

Related concepts:

- "Data type mappings in a federated system" on page 45
- "INSERT, UPDATE, and DELETE statements and large objects (LOBs)" on page 95

Related reference:

- Chapter 27, "Default forward data type mappings," on page 261

Chapter 4. Function mappings and user-defined functions

The wrappers that are included with DB2 Information Integrator contain default function mappings between the data sources and DB2 for Linux, UNIX, and Windows.

Function mappings in a federated system

DB2® Information Integrator supplies default mappings between existing built-in data source functions and built-in DB2 counterpart functions. For the federated server to recognize a data source function, the function must be mapped to an existing counterpart function in DB2 for Linux, UNIX®, and Windows®.

The default function mappings are in the wrapper modules.

For nonrelational data sources, you cannot override the exisitng function mappings or create new mappings.

When to create your own function mappings

When a default function mapping is not available for a data source function, you can create a function mapping. One reason that a mapping is not available is that there is no corresponding function between the data source and DB2 for Linux, UNIX, and Windows.

Another reason a mapping is not available is that the data source has a similar function to a DB2 function, but it does not return the same results. If the data source returns slightly different results or different results for certain sets of input data, the wrappers do not normally map to these functions. However, if you do not care about the differences in the result sets, then you can create a mapping between the functions. Creating a mapping might improve performance.

Use functions mappings when

- A new built-in function becomes available at the data source
- A new user-defined function becomes available at the data source
- A DB2 counterpart function does not exist

- A counterpart function exists but returns slightly different results, which you do not care about

The settings for function mappings are stored in the SYSCAT.FUNCMAPPINGS catalog view.

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 Information Integrator uses the function mapping, but it might result in an SQL syntax error or unexpected results.

Why function mappings are important

Function mappings are one of several important inputs to the pushdown analysis performed by the query optimizer. In deciding on the best query access plan, the query optimizer factors the capabilities of the data source to perform a particular type of SQL function or operation. If the function does not have a mapping, the function will not be sent to the data source for processing. Functions and other operations that can be pushed down to the data source improve performance.

If the data source has a similar function to a DB2 function, but it returns slightly different results, creating a function mapping might improve performance. For example, the Informix® STDEV (standard deviation) function produces different results than the DB2 STDDEV function for some sets of input data. For this reason the Informix wrapper does not have a default mapping between these two functions. If you do not care about the result set differences, you might improve the performance of queries that access Informix data sources and use the DB2 STDDEV function. By creating a function mapping between the Informix STDEV and the DB2 STDDEV function, you provide the query optimizer the choice of sending the processing of that function down to the data source.

Related concepts:

- "How function mappings work in a federated system" on page 58
- "Requirements for mapping user-defined functions (UDFs)" on page 59
- "Function templates" on page 60
- "How to create function mappings" on page 64

Related tasks:

- "Providing function mapping overhead information to the query optimizer" on page 62
- "Specifying function names in a function mapping" on page 64

How function mappings work in a federated system

When you submit queries to the federated server that contain one or more functions, the federated 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. It also contains new mappings that you create when there is no default function mapping. When multiple mappings can be applied to a function, the most recently created one is applied.

Function mapping options specify information about the function and the potential cost of processing a function at the data source. 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.

When you create a function mapping, you are mapping a data source function to a counterpart function at the federated database. When a DB2 counterpart function does not exist, 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.

Related concepts:

- "Function mappings in a federated system" on page 57
- "Requirements for mapping user-defined functions (UDFs)" on page 59

Requirements for mapping user-defined functions (UDFs)

Before you can invoke a data source user-defined function in a federated system, the federated database must associate the data source function with a function specification stored in the global catalog on the federated server.

There are two conditions under which the federated database can associate a function specification with a data source function:

- DB2® UDB has a function whose signature corresponds to that of the signature of the data source function. A *signature* consists of a function name and function input parameters. Signatures *correspond* when both the following conditions are true:
  - They contain the same names and the same number of parameters
  - 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.
- If DB2 UDB does not have a function with the requisite signature, you can define a function template that contains this signature. You then map the function template to the data source function that you want to invoke.

The settings for function mappings are stored in the SYSCAT.FUNCMAPPINGS catalog view.

Related concepts:

- "Function mappings in a federated system" on page 57
- "User-defined functions in applications" on page 67

Related tasks:

- "Dropping a user-defined function mapping" on page 69

Function templates

The federated server recognizes a data source function when there is a mapping between the data source function and a DB2® counterpart function at the federated database.

You can create a function template to act as the DB2 counterpart function when no counterpart exists.

A *function template* is a DB2 function that 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. When the federated server receives queries that specify the function template, the federated server will invoke the data source function.

The function template is created with the CREATE FUNCTION statement using the AS TEMPLATE parameter.

After you create a function template, you must then create the function mapping between the template and the data source function. A function mapping is created using the CREATE FUNCTION MAPPING statement.

Related concepts:

- "Function mappings in a federated system" on page 57

Related tasks:

- "Creating function templates" on page 60

Creating function templates

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.

Prerequisites:

The privileges held by the authorization ID of the statement must include at least one of the following:

- SYSADM or DBADM authority
- IMPICIT_SCHEMA authority on the databaes, if the implicit or explicit schema name of the function does not exist
- CREATEIN privilege on the schema, if the schema name of the function exists

Restrictions:

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. The data type cannot be LONG VARCHAR, LONG VARGRAPHIC, DATALINK, or a user-defined type.

If the data source function has no input parameters, the DB2 counterpart function cannot have any input parameters.

Procedure:

A 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
   DETERMINISTIC
   NO EXTERNAL ACTION
```

BONUS ()
The name you give to the function template.

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

AS TEMPLATE
Indicates that this is a function template, not a function.

DETERMINISTIC
Specifies that the function always returns the same results for a given set of argument values.

NO EXTERNAL ACTION
Specifies that the function has no external impact on objects that are not managed by the database manager.

You must specify the DETERMINISTIC and NO EXTERNAL ACTION clauses according to whether the function itself is deterministic and whether it causes any external action. Otherwise, restrictions will be imposed on the SQL operations that are supported with this function template.

After you create a function template, you must then create the function mapping between the template and 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()')
```

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

FOR *BONUS()*
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 that you want to map to.

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

Related concepts:

- "Function mappings in a federated system" on page 57
- "Function templates" on page 60

Related reference:

- "CREATE FUNCTION (Sourced or Template) statement" in the *SQL Reference, Volume 2*

Providing function mapping overhead information to the query optimizer

When a query that contains a function is received by the DB2 SQL Complier, the query optimizer determines if the function can be pushed down to the data source. Suppose that pushdown analysis determines that either the data source or DB2 UDB is able to process the function.

When you create a function mapping, you can provide DB2 UDB with important information about the potential cost, or *overhead*, of executing a data source function at the data source. The query optimizer uses these overhead estimates to compare the estimated cost of executing the data source function with the estimated cost of executing the DB2 function.

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. The strategy that is expected to cost the least amount of overhead is used.

You include estimated overhead statistics 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.ROUTINES view.

Prerequisites:

The privileges held by the authorization ID of the statement must have SYSADM or DBADM authority.

Procedure:

To specify estimated statistics in the CREATE FUNCTION MAPPING statement, use the function mapping options. The following table lists the function mapping options that specify function overhead and the default values for these options.

Table 5. Function mapping options that specify function overhead

Option	Valid settings	Default setting
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'

Example: PERCENT_ARGBYTES function mapping option:

Suppose that you want to map a user-defined function named US_DOLLAR at an Oracle data source to a DB2 user-defined function that you create. The Oracle data source server is called ORACLE2. You decide to name the DB2 user-defined function DOLLAR and to name this function mapping ORACLE_DOLLAR. You want set the PERCENT_ARGBYTES function mapping option to provide the optimizer with more accurate overhead information. The SQL statement would be:

```
CREATE FUNCTION MAPPING ORACLE_DOLLAR FOR DOLLAR()
   SERVER ORACLE2
   OPTIONS (REMOTE_NAME 'US_DOLLAR()', PERCENT_ARGBYTES'250')
```

Example: INSTS_PER_INVOC function mapping option:

Suppose that you want to map the local function UCASE(CHAR) to an Oracle user-defined function called UPPERCASE. The Oracle function is at a data source called ORACLE2. You want to include the estimated number of instructions per invocation of the Oracle user-defined function. The syntax is:

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN4 FOR SYSFUN.UCASE(CHAR)
   SERVER ORACLE2
   OPTIONS (REMOTE_NAME 'UPPERCASE', INSTS_PER_INVOC '1000')
```

Updating overhead information

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, you must first drop or disable the function mapping. Then re-create the mapping specifying the new estimates in the CREATE FUNCTION MAPPING statement. The new estimates will be added to the SYSCAT.FUNCMAPPINGS catalog view. To record changed estimates for the DB2 function, you can update the SYSSTAT.ROUTINES catalog view directly.

Related concepts:

- "Function mappings in a federated system" on page 57

Related reference:

- Chapter 25, "Function mapping options for federated systems," on page 253

Specifying function names in a function mapping

The values you enter in the CREATE FUNCTION MAPPING statement depends on whether the functions that you are mapping together have the same name or different names.

Prerequisites:

The privileges held by the authorization ID of the statement must have SYSADM or DBADM authority.

Mapping functions with the same names:

You can create a mapping between two functions (or a DB2 function template and a data source function) that have the same name. For example, suppose you want to map a user-defined function named MYFUN at an Informix data source to the DB2 user-defined function named TINA.MYFUN. The Informix data source server is called INFORMIX2. The syntax would be:

```
CREATE FUNCTION MAPPING FOR TINA.MYFUN(SYSTEM.INTEGER) SERVER INFORMIX2
```

Mapping functions with different names:

If you create a mapping between two functions (or a DB2 function template and a data source function) that have different names, then:

- Assign the name of the DB2 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 REMOTE_NAME must be less than 255 characters.

For example, suppose that you want to map a user-defined function named UPPERCASE at an Oracle data source to the DB2 function UCASE(CHAR). The Oracle data source server is called ORACLE2. 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')
```

Related concepts:

- "Function mappings in a federated system" on page 57

How to create function mappings

Use the CREATE FUNCTION MAPPING statement specify alternative function mapping that override the default function mappings. When you create alternative function mappings, the entries appear in the SYSCAT.FUNCMAPPINGS catalog view.

You can also use the CREATE FUNCTION MAPPING statement to specify function mapping options. When you specify function mapping options, the information appears in the SYSCAT.FUNCMAPOPTIONS catalog view.

With the CREATE FUNCTION MAPPING statement, you can:

- Create a function mapping for all data sources of a specific type. For example, all Informix® data sources.
- Create a function mapping for all data sources of a specific type and version. For example, all Informix 9 data sources.
- Create a function mapping for a specific server.
- Provide function mapping statistical information to the optimizer
- Disable a default function mapping or a function mapping that you defined.

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.

Related tasks:

- "Creating a function mapping for a specific data source type" on page 65
- "Creating a function mapping for a specific data source type and version" on page 66
- "Creating a function mapping for all data source objects on a specific server" on page 67

How to create function mappings-details

Use the CREATE FUNCTION MAPPING statement to create a mapping to a DB2 function or function template.

Creating a function mapping for a specific data source type

You can create a mapping to a function for all data sources of a specific type.

Prerequisites:

The privileges held by the authorization ID of the statement must have SYSADM or DBADM authority.

Restrictions:

You cannot override the exisitng function mappings or create new mappings for nonrelational data sources.

Procedure:

Suppose that you want to map a DB2 function template to an Oracle user-defined function, for all Oracle data sources. The template is called STATS and belongs to a schema called NOVA. The Oracle user-defined function is called STATISTICS and belongs to a schema called STAR. The CREATE FUNCTION MAPPING statement is:

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN1
   FOR NOVA.STATS ( DOUBLE, DOUBLE )
   SERVER TYPE ORACLE
   OPTIONS (REMOTE_NAME 'STAR.STATISTICS')
```

Related concepts:

- "How to create function mappings" on page 64

Related tasks:

- "Creating a function mapping for a specific data source type and version" on page 66
- "Creating a function mapping for all data source objects on a specific server" on page 67
- "Dropping a user-defined function mapping" on page 69
- "Specifying function names in a function mapping" on page 64

Related reference:

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

Creating a function mapping for a specific data source type and version

You can create a mapping to a function for all data sources that use a specific version of the data source type.

Prerequisites:

The privileges held by the authorization ID of the statement must have SYSADM or DBADM authority.

Restrictions:

You cannot override the exisitng function mappings or create new mappings for nonrelational data sources.

Procedure:

Suppose that you want to map a DB2 function template to a Sybase user-defined function, for all Sybase data sources that use Version 12. The template is called SYB_STATS and belongs to a schema called EARTH. The Sybase user-defined function is called STATISTICS and belongs to a schema called MOON. The CREATE FUNCTION MAPPING is:

```
CREATE FUNCTION MAPPING SYBASE_STATS
   FOR EARTH.SYB_STATS ( DOUBLE, DOUBLE )
   SERVER TYPE SYBASE VERSION 12
   OPTIONS (REMOTE_NAME 'MOON.STATISTICS')
```

Related concepts:

- "How to create function mappings" on page 64

Related tasks:

- "Creating a function mapping for a specific data source type" on page 65
- "Creating a function mapping for all data source objects on a specific server" on page 67

- "Dropping a user-defined function mapping" on page 69
- "Specifying function names in a function mapping" on page 64

Related reference:
- "CREATE FUNCTION MAPPING statement" in the *SQL Reference, Volume 2*

Creating a function mapping for all data source objects on a specific server

You can create a mapping to a function that is used by all data sources objects on a specific remote server.

Prerequisites:

The privileges held by the authorization ID of the statement must have SYSADM or DBADM authority.

Restrictions:

You cannot override the exisitng function mappings or create new mappings for nonrelational data sources.

Procedure:

Suppose that you want to map a function template called BONUS to a user-defined function called BONUS. You only want the mapping to apply to an Oracle data source server called ORA_SALES. Because the function names are the same, you do not need to specify the REMOTE_NAME function mapping option.

```
CREATE FUNCTION MAPPING BONUS_CALC FOR BONUS()
 SERVER ORA_SALES
```

Related concepts:
- "How to create function mappings" on page 64

Related tasks:
- "Creating a function mapping for a specific data source type" on page 65
- "Creating a function mapping for a specific data source type and version" on page 66
- "Dropping a user-defined function mapping" on page 69
- "Specifying function names in a function mapping" on page 64

Related reference:
- "CREATE FUNCTION MAPPING statement" in the *SQL Reference, Volume 2*

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. User-defined functions 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 user-defined functions exactly like built-in functions such as SUBSTR and LENGTH. You can develop applications using different application language environments, such as C, C++, COBOL, and FORTRAN. The applications can share a set of SQL user-defined functions even though they are developed using different application language environments.

User-defined functions can manipulate data and perform actions. For example, you might enable a user-defined function to send an electronic message or to update a flat file.

In DB2®, user-defined functions 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*
- "Function mappings in a federated system" on page 57
- "Requirements for mapping user-defined functions (UDFs)" on page 59
- "How to create function mappings" on page 64

Related tasks:

- "Dropping a user-defined function mapping" on page 69

Disabling a default function mapping

Default function mappings cannot be dropped. However, you can render them inoperable by disabling them.

Prerequisites:

The privileges held by the authorization ID of the statement must have SYSADM or DBADM authority.

Procedure:

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'.

Suppose 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 query optimizer to require less overhead.

You want to determine how performance is affected if only the WEEK function on the Oracle data source is invoked. To ensure that the WEEK function on the Oracle data source is invoked each time, you must disable the default function mapping. The syntax is:

```
CREATE FUNCTION MAPPING FOR SYSFUN.WEEK(INT)
TYPE ORACLE OPTIONS (DISABLE 'Y')
```

Related concepts:

- "Function mappings in a federated system" on page 57

Related tasks:

- "Dropping a user-defined function mapping" on page 69

Related reference:

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

Dropping a user-defined function mapping

When you no longer require a function mapping that you created, you can delete the function mapping.

If you drop a user-defined function mapping that was created to override a default function mapping, the default function mapping will be used.

User-defined function mappings are listed in the SYSCAT.FUNCMAPPINGS catalog view.

Prerequisites:

The privileges held by the authorization ID of the statement must have SYSADM or DBADM authority.

Procedure:

To drop a function mapping that you created, use the DROP FUNCTION MAPPING statement.

Suppose that you have a function mapping called BONUS_CALC. To drop the function mapping, the DROP FUNCTION MAPPING is:

```
DROP FUNCTION MAPPING BONUS_CALC
```

Related concepts:

- "How to create function mappings" on page 64

Related tasks:

- "Disabling a default function mapping" on page 68

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*

Chapter 5. Index specifications

An *index specification* is a set of metadata that is added to the global catalog when you create a nickname for a data source object. This information is used by the query optimizer to expedite the processing of distributed requests. There are some situations in which the index specification is not created when you create a nickname for a data source object. In these situations, you will need to create the index specification yourself.

In this chapter you will learn about:

Index specifications in a federated system

In a federated system, you use the CREATE INDEX statement with 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. The index specification tells the federated server about the columns and their uniqueness properties that comprise a remote index. It does not tell the federated server about the statistical properties of the index, such as, the number of unique values of the index key.

You do not need to supply index specifications if the remote index was in place at the time that the nickname was created.

A federated server does not create an index specification when you create a nickname for:

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

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

Use index specifications with relational data sources. Creating an index specification for a nonrelational data source will not improve performance.

Related tasks:

- "Creating index specifications for data source objects" on page 72
- "Creating index specifications on tables that acquire new indexes" on page 73
- "Creating index specifications on views" on page 75
- "Creating index specifications on Informix synonyms" on page 76
- "Nickname characteristics affecting global optimization" on page 147

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 is 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, Excel spreadsheet, BLAST algorithm, or XML tagged file.
- The remote index is on a LOB column.
- The remote index contains a total key length greater than 1024 bytes.
- The maximum number of key parts is more than 16.

In these circumstances the federated server does not store index specifications for the data source objects. However, for the first two items in the previous list you can supply the necessary index information to the global catalog. You can use the CREATE INDEX statement to specify the index information.

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 can 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 in a federated system" on page 71

Related tasks:

- "Creating index specifications on tables that acquire new indexes" on page 73
- "Creating index specifications on views" on page 75
- "Creating index specifications on Informix synonyms" on page 76

Related reference:

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

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 can 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.

Example: A table that has no index, later acquires an index:

Suppose that you create the nickname *employee* for 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.

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.

Example: 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.

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 in a federated system" on page 71

Related tasks:

- "Creating index specifications for data source objects" on page 72
- "Creating index specifications on views" on page 75
- "Creating index specifications on Informix synonyms" on page 76

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 Compiler 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 can 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_sales2003* for a view called JAPAN_SALES2003. 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 must be created separately. For example, to create an index specification that describes the REGION index, the syntax would be:

```
CREATE UNIQUE INDEX jp_2003_region ON jp_sales2003
(REGION) SPECIFICATION ONLY
```

where *jp_2003_region* is the index name, and *jp_sales2003* is the nickname for the view JAPAN_SALES2003.

Related concepts:

- "Index specifications in a federated system" on page 71

Related tasks:

- "Creating index specifications for data source objects" on page 72
- "Creating index specifications on tables that acquire new indexes" on page 73
- "Creating index specifications on Informix synonyms" on page 76

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 that 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 can 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.

Example: 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.

Example: A nickname is created on an Informix synonym that is based on a view:

Suppose that you create the nickname *jp_sales2003* for a view called JAPAN_SALES2003. 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 must be created separately.

To create an index specification that describes REGION index, the syntax would be:

```
CREATE UNIQUE INDEX jp_2003_region ON jp_sales2003 (REGION) SPECIFICATION ONLY
```

where *jp_2003_region* is the index name and *jp_sales2003* is the nickname for the view JAPAN_SALES2003.

Related concepts:

- "Index specifications in a federated system" on page 71

Related tasks:

- "Creating index specifications for data source objects" on page 72
- "Creating index specifications on tables that acquire new indexes" on page 73
- "Creating index specifications on views" on page 75

Related reference:

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

Chapter 6. Transparent DDL

In a federated system, you can create tables through the DB2 Linux, UNIX, and Windows federated server on the remote data sources. Tables that were created through the federated server, can be altered and dropped through the federated server. The ability to create remote tables through the DB2 federated server is called *transparent DDL*.

In this chapter you will learn:

What is transparent DDL?

Transparent DDL provides the ability to create and modify remote tables through DB2® Information Integrator without the need to use pass-through sessions.

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. 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.

The advantage of using transparent DDL is that DB2 database administrators can use procedures that they are familiar with to create both local and remote tables. Transparent DDL centralizes table administration and facilitates granting authorizations.

Transparent DDL is supported with the following data sources:

- DB2 for z/OS™ and OS/390®
- DB2 for iSeries™
- DB2 for Linux, UNIX®, and Windows®
- DB2 Server for VM and VSE
- Informix
- Microsoft® SQL Server
- ODBC
- Oracle
- Sybase
- Teradata

The database administrator can either use the DB2 Control Center or DDL statements in the DB2 command line processor (CLP) to create the tables. Using transparent DDL avoids the need to learn the different DDL syntax required for each data source.

Before you can create remote tables on a data source through DB2 Information Integrator you need to configure access to the data source:

- The wrapper for that data source needs to be registered in the global catalog
- 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 Information Integrator and the data source server

Use the remote table wizard in the DB2 Control Center to create remote tables.

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

To issue transparent DDL statements, your authorization ID must have the necessary privileges on the nickname (for the federated database to accept the request), and the comparable privileges on the remote data source server (for the data source to accept the request).

Transparent DDL restrictions

Transparent DDL has some limitations:

- You cannot modify or drop tables that were natively created at the remote data source
- Materialized query tables cannot be created on remote data sources
- 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, unique key, or check constraints. The columns used for a primary key must be NOT NULL, and cannot include columns containing LOBs.
- You cannot modify the parameters of existing columns, such as the data type or length
- The DEFAULT clause in the CREATE TABLE and ALTER TABLE statements is not supported

Related tasks:

- "Creating new remote tables using transparent DDL" on page 81
- "Altering remote tables using transparent DDL" on page 84

• "Dropping remote tables using transparent DDL" on page 86

Remote LOB columns and transparent DDL

Some data sources, such as Oracle and Informix®, do not store the lengths of LOB columns in their system catalogs. When you create a nickname on a table, information from the data source system catalog is retrieved including column length. Since no length exists for the LOB columns, the federated database assumes that the length is the maximum length of a LOB column in DB2® for Linux, UNIX®, and Windows®. The federated database stores the DB2 for Linux, UNIX, and Windows maximum length in the federated database catalog as the length of the nickname column.

However, when you create a remote table using transparent DDL you must specify the length of the LOB column. When the federated server creates a nickname on the remote table, it stores the length you specify in the federated database catalog as the length of the nickname column.

Related concepts:

• "What is transparent DDL?" on page 79
• "Federated LOB support" on page 204

Related tasks:

• "Creating new remote tables using transparent DDL" on page 81

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.

Prerequisites:

Before you create a remote table, you must configure the federated server to access that data source. This configuration includes:

- Creating the wrapper for that data source type
- Supplying the server definition for the server where the remote table will be located
- Creating the user mappings between DB2 Information Integrator and the data source server

To issue transparent DDL statements, your authorization ID must have the necessary privileges on the nickname (for the federated database to accept the request), and the comparable privileges on the remote data source server (for the data source to accept the request).

The privileges held by the authorization ID issuing 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:

Materialized query tables cannot be created on remote data sources.

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. The columns used for a primary key must be NOT NULL, and cannot include columns containing LOBs.

The DEFAULT clause in the CREATE TABLE statement is not supported.

Procedure:

You can create a remote table using transparent DDL from the DB2 Control Center or the command line prompt.

To create a remote table in the DB2 Control Center, use the Create Remote Table wizard:

1. Expand the **Federated Database Objects** folder.
2. Expand the wrapper and server definition objects for the data source that you want to create a remote table for.
3. Right-click the **Remote Tables** folder and click **Create**. The Create Remote Table wizard starts.
4. Complete the steps in the wizard.

To do this task from the command line prompt, issue the CREATE TABLE statement with the appropriate parameters set.

The remote data source must support the column data types and primary key option in the CREATE TABLE statement. For example, suppose that the remote data source does not support primary keys. Depending on how the data source responds to requests it does not support, an error might be returned or the request might be ignored.

The remote server must be specified in the OPTIONS clause. The OPTIONS clause can be used to override the remote name or the remote schema of the table being created.

Suppose that you want to create the table EMPLOYEE on an Oracle server. In the CREATE TABLE statement, use the DB2 data types when you specify each column. The syntax to use the CLP to create the table is:

```
CREATE TABLE EMPLOYEE
   ( EMP_NO       CHAR(6) NOT NULL,
     FIRSTNAME    VARCHAR(12) NOT NULL,
     MIDINT       CHAR(1) NOT NULL,
     LASTNAME     VARCHAR(15) NOT NULL,
     HIREDATE     DATE,
     JOB          CHAR(8),
     SALARY       DECIMAL(9,2),
     PRIMARY KEY (EMP_NO) )
     OPTIONS (REMOTE_SERVER 'ORASERVER',
     REMOTE_SCHEMA 'J15USER1', REMOTE_TABNAME 'EMPLOY' )
```

EMPLOYEE
The local table name. This name is also used for the nickname associated with the table.

REMOTE_SERVER *'ORASERVER'*
The name that you supplied for the server in the CREATE SERVER statement. This value is case-sensitive.

REMOTE_SCHEMA *'J15USER1'*
The remote schema name. Although this parameter is optional, it is recommended that you specify a schema name. If this parameter is not specified, the local AUTHID in uppercase is used for the remote schema name. This value is case-sensitive.

REMOTE_TABNAME *'EMPLOY'*
The remote table name. This parameter is optional. If this parameter is not specified, the local table name is used for the remote table name. This value must be a valid name on the remote data source and cannot be an existing table name. This value is case-sensitive.

When a remote table is created through DB2 Information Integrator using transparent DDL, several other actions occur:

- A nickname is automatically created for the remote table. The nickname has the same name as the local table. The remote table has the same name as the local table unless you specify another name using the REMOTE_TABNAME option. The remote table nickname can be used like any other nickname. In addition, you can ALTER and DROP the remote table (something you cannot do with a nickname created using CREATE NICKNAME).
- A row is added in the SYSCAT.TABOPTIONS catalog view with an option name of TRANSPARENT and a value of '.

In the example above, DB2 Information Integrator uses reverse data type mappings to map the DB2 data types to Oracle data types. On the remote Oracle server, the EMPLOY table is created using Oracle data types. The following table shows the mappings from the DB2 data types to the Oracle data types for the columns specified in the example.

Table 6. An example of reverse data type mappings from DB2 Information Integrator to Oracle

Column	DB2 data type specified in the CREATE TABLE statement	Oracle data type used in the remote table
EMP_NO	CHAR(6) NOT NULL	CHAR(6) NOT NULL
FIRST_NAME	VARCHAR(12) NOT NULL	VARCHAR2(12) NOT NULL

Table 6. An example of reverse data type mappings from DB2 Information Integrator to Oracle (continued)

Column	DB2 data type specified in the CREATE TABLE statement	Oracle data type used in the remote table
MID_INT	CHAR(1) NOT NULL	CHAR(1) NOT NULL
LAST_NAME	VARCHAR(15) NOT NULL	VARCHAR2(15) NOT NULL
HIRE_DATE	DATE	DATE
JOB	CHAR(8)	CHAR(8)
SALARY	DECIMAL(9,2)	NUMBER(9,2)

The SQL_SUFFIX option is allowed at the end of the CREATE TABLE statement. This option is primarily used to specify the IN TABLESPACE clause when creating remote tables on DB2 family data sources.

Related concepts:

- "What is transparent DDL?" on page 79

Related tasks:

- "Altering remote tables using transparent DDL" on page 84
- "Dropping remote tables using transparent DDL" on page 86

Related reference:

- "CREATE TABLE statement" in the *SQL Reference, Volume 2*
- Chapter 28, "Default reverse data type mappings," on page 277

Altering remote tables using transparent DDL

You can alter remote data source tables that were created through DB2 Information Integrator using transparent DDL. You cannot alter tables that were created directly at the remote data source.

Use the ALTER TABLE statement to modify tables created through DB2 Information Integrator using transparent DDL. Using the ALTER TABLE statement you can:

- Add new columns
- Add the table primary key

Prerequisites:

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

To issue transparent DDL statements, your authorization ID must have the necessary privileges on the nickname (for the federated database to accept the request), and the comparable privileges on the remote data source server (for the data source to accept the request).

Restrictions:

You cannot modify tables that were natively created at the remote data source.

An existing primary key cannot be altered or dropped in a remote table.

Altering a remote table invalidates any packages dependent on the nickname associated with the remote table.

The remote data source must support the changes in the ALTER TABLE statement. For example, suppose that the remote data source does not support primary keys. Depending on how the data source responds to requests it does not support, an error might be returned or the request might be ignored.

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. The columns used for a primary key must be NOT NULL, and cannot include columns containing LOBs.

You cannot modify the parameters of existing columns, such as the data type or length.

The DEFAULT clause in the ALTER TABLE statement is not supported.

Procedure:

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.

Do not use the ALTER TABLE statement to add or modify column options. Use the ALTER NICKNAME statement instead.

Suppose you want to add a primary key on a remote table EMPLOYEE that you created using transparent DDL. Using the CLP to modify the table, the syntax is:

```
ALTER TABLE EMPLOYEE
   ADD PRIMARY KEY (EMP_NO, WORK_DEPT)
```

The columns used for a primary key must be NOT NULL, and cannot be columns containing LOBs.

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
   ADD COLUMN SHIP_DATE  DATE
```

Related concepts:

- "What is transparent DDL?" on page 79

Related tasks:

- "Creating new remote tables using transparent DDL" on page 81
- "Dropping remote tables using transparent DDL" on page 86

Related reference:

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

Dropping remote tables using transparent DDL

You can drop remote data source tables that were created through DB2 Information Integrator using transparent DDL. You cannot drop tables that were created directly at the remote data source.

Prerequisites:

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

To issue transparent DDL statements, your authorization ID must have the necessary privileges on the nickname (for the federated database to accept the request), and the comparable privileges on the remote data source server (for the data source to accept the request).

Restrictions:

You cannot drop tables that were natively created at the remote data source.

Procedure:

To drop a remote table that was created through DB2 Information Integrator using transparent DDL, you can use either the DB2 Control Center or the DROP statement. Dropping a nickname for a remote table created using transparent DDL merely drops the local nickname for that table. The DROP NICKNAME statement does not drop the remote table. You must use the DROP TABLE statement to drop the remote table.

Dropping a remote table first deletes the table on the data source, then deletes the corresponding nickname for the remote table in the federated database. Deleting the nickname invalidates any packages based on that nickname.

To drop the table SPALTEN, the syntax is:

```
DROP TABLE SPALTEN
```

where *SPALTEN* is the local name for the remote table.

Related concepts:

- "What is transparent DDL?" on page 79

Related tasks:

- "Creating new remote tables using transparent DDL" on page 81
- "Altering remote tables using transparent DDL" on page 84

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*

Chapter 7. Transaction support in a federated system

Federated system transaction support builds on DB2 distributed database transaction functionality. This chapter assumes that you understand basic DB2 distributed transaction processing concepts, which are described in the following DB2 manuals:

- *DB2 Administration Guide: Planning (SC09–4822–00)*
- *DB2 Application Development Guide: Programming Client Applications (SC09–4826–00)*
- *DB2 Application Development Guide: Programming Server Applications (SC09–4827–00)*

This chapter includes the following topics:

Understanding federated system transaction support

Federated system transaction processing builds on DB2® distributed database transaction capabilities. To understand federated system transaction processing, you must understand the following essential DB2 distributed transaction processing concepts:

- Unit of work (UOW)
- Remote unit of work (RUOW)
- Distributed unit of work (DUOW)
- Multisite update
- Transaction manager (TM)
- Resource manager (RM)
- Type 1 connection
- Type 2 connection
- One-phase commit
- Two-phase commit

These concepts work identically in both federated and non-federated DB2 systems. However, the scope of each concept changes in a federated system.

For example, a unit of work is implicitly begun when any data in a database is read or written. For a unit of work in a federated system, the database can be a federated database or a data source database. For a distributed unit of work in a federated system, you can access both a federated database and a data source database.

An application must end a unit of work by issuing either a COMMIT or a ROLLBACK statement, regardless of the number of databases that are accessed. The COMMIT statement makes permanent all changes made within a unit of work. The ROLLBACK statement removes these changes from a database. Changes made by a unit of work become visible to other applications after a successful commit.

Recommendation: Always explicitly commit or roll back units of work in your applications.

When only one data source is updated in a unit of work, a COMMIT statement for this unit of work can be sent in a single operation. This operation is called a *one-phase commit* or single site update. A *site* corresponds to a server definition in a federated system. A federated server is the local site for update operations in a federated system. Any remote data source is a remote site for update operations in a federated system.

In a one-phase commit operation, no other data sources in the same unit of work are required to verify their ability to commit data.

Examples of a one-phase commit operation include:

- A non-distributed unit of work
- A distributed unit of work that involves reading from one or more data sources but updating data at only one data source

In a distributed unit of work that involves updates of multiple databases on multiple sites, data must be consistent. The multisite update or two-phase commit protocol is commonly used to ensure data consistency across multiple databases within a distributed unit of work. Federated systems do not currently support the two-phase commit protocol.

Related concepts:

- "Units of work" in the *Administration Guide: Planning*
- "DB2 transaction manager" in the *Administration Guide: Planning*
- "Two-phase commit" in the *Administration Guide: Planning*
- "Resource manager setup" in the *Administration Guide: Planning*
- "Multisite Update" in the *Application Development Guide: Programming Client Applications*
- "What is an update in a federated system?" on page 90

Related tasks:

- "Updating a database from a host or iSeries client" in the *Administration Guide: Planning*
- "Updating a single database in a transaction" in the *Administration Guide: Planning*

What is an update in a federated system?

In a federated system, an update is not just a transaction that includes an INSERT, UPDATE, or DELETE statement. There are certain actions, including some transactions that have other statements, which are also considered updates by the federated server.

Updates can be performed locally or remotely.

Local site updates are updates to objects on a federated database. Tables and views in the federated database are local objects. You do not create nicknames for these objects, you use the actual object name in the statements.

Remote site updates are updates to objects on a remote data source. Remote data sources include:

- Another DB2® for Linux, UNIX®, and Windows® database or instance on the federated server
- Another DB2 for Linux, UNIX, and Windows database or instance on another server
- Data sources other than DB2 for Linux, UNIX, and Windows, such as DB2 for iSeries™, Informix®, Oracle, and Teradata

It is important that you understand what the federated server considers an update transaction and the types of updates that are allowed in a federated system. There are four types of actions that the federated server considers to be update transactions. The following table shows the updates that you can perform on a federated system.

Table 7. Types of updates and the site where the updates are performed

Type of action	Local site	Remote site	Explanation
Local update (DDL and DML)	Y	N	An update on an object in the federated database.
Remote update (nickname)	N	Y	An update on a remote data source object that you created a nickname for.
Dynamic SQL in pass-through sessions	N	Y	An update on a remote data source object. You cannot use a pass-through session to update local objects.
Transparent DDL	Y	Y	A pair of transactions that create, alter, or drop remote tables and their corresponding nicknames in a federated database. For example, a pair of transactions that create a remote table on a data source and a nickname on the federated server.

What is an update transaction in a pass-through session?

A federated server treats all dynamic SQL statements sent through pass-through sessions as updates. This behavior ensures data integrity. If a dynamic SQL statement that is sent through a pass-through session is successful, the transaction is recorded as an update. The SQL can be any type of statement, including SELECT statements.

Transaction support with transparent DDL

Transparent DDL creates a table on a remote data source and a nickname in the local federated database for the remote table. A federated server treats transparent DDL transactions as updates.

COMMIT or ROLLBACK statements must be issued before and after transparent DDL transactions occur.

Because transparent DDL is creating both local and remote objects at the same time, each transparent DDL statement that is issued must be the only update

within a one-phase commit transaction. If there is an update before the transparent DDL transaction, a COMMIT or ROLLBACK statement must be issued before the transparent DDL transaction occurs. Likewise, a COMMIT or ROLLBACK statement must be issued after the transparent DDL transaction, before any other update can occur.

Data sources that automatically commit DDL statements

Some data sources automatically commit the current transaction locally after a successful DDL transaction. If you create a remote table using transparent DDL or in a pass-through session, these data sources cannot rollback the remote table after the table is created. You must delete the remote table manually.

The only federated data source that automatically commits DDL statements is Oracle.

User-defined functions that are pushed down to the data source for processing

A federated server treats user-defined functions that are pushed down to a data source as read-only statements. If a remote user-defined function performs an update on a data source, the federated server is unaware of the update. Because the federated server does not treat these user-defined functions as update statements, there is no statement rollback.

Important: Data integrity cannot be guaranteed when a user-defined function that is pushed down to a data source performs an update.

Related concepts:

- "Querying data sources directly with pass-through" on page 208
- "Understanding federated system transaction support" on page 89

Chapter 8. INSERT, UPDATE, and DELETE operations

This chapter describes how to access and update data at the data sources.

This chapter contains:

Authorization privileges for INSERT, UPDATE, and DELETE statements

The privileges required to issue INSERT, UPDATE, and DELETE statements on nicknames are similar to the privileges required to issue these same statements on tables. In addition, you must hold adequate privileges on the data source to perform select, insert, update, and delete operations on the underlying object.

You can grant or revoke SELECT, INSERT, UPDATE, and DELETE privileges on a nickname.

However, granting or revoking privileges on a nickname does not grant or revoke privileges at the data source. At the data source, the privileges must be granted or revoked for the REMOTE_AUTHID specified in the user mapping at the federated server.

The privileges held by the authorization ID of the statement must include the necessary privileges on the nickname (for the federated database to accept the request). The user ID at the data source that is mapped to the authorization ID (through a user mapping) must have the necessary privileges on the underlying table object (for the data source to accept the request).

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 data source object that the nickname refers to.

Likewise, just because you have UPDATE privilege on the nickname does not mean you will automatically be authorized to update the data source 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 authorization ID is mapped to the data source user ID. The privilege restrictions will be enforced at the data source.

Related tasks:

- "Altering a user mapping" on page 30
- "The SQL statements you can use with nicknames" on page 98

Federated system INSERT, UPDATE, and DELETE restrictions

In federated systems, an update is not just a transaction that includes an INSERT, UPDATE, or DELETE statement. There are certain actions which are also considered updates by the federated server. This includes some transactions that have SELECT statements. See What is an update in a federated system?.

Updates can be performed locally or remotely. Local site updates are updates to objects on the federated database. Tables and views in the federated database are local objects. Remote site updates are updates to objects on a remote data source.

The following restrictions apply to updates on nicknames:

- An object whose data source does not permit updates cannot be updated
- A data source object that is read-only, such as a JOIN view, cannot be updated
- You cannot perform insert, update, and delete operations on federated views created with UNION ALL statements. Federated views created with UNION ALL statements are read-only views.

Unsupported data sources

Federation does not support insert, update, and delete operations on the following data sources:

- BLAST
- BioRS
- Documentum
- Entrez
- Excel
- Extended Search
- HMMER
- Table-structured files
- Web services
- WebSphere® Business Integration
- XML

Related concepts:

- "Authorization privileges for INSERT, UPDATE, and DELETE statements" on page 93
- "Referential integrity in a federated system" on page 94
- "INSERT, UPDATE, and DELETE statements and large objects (LOBs)" on page 95

Referential integrity in a federated system

In a federated system, DB2® UDB does not compensate for referential integrity differences between data sources. DB2 UDB does not interfere with referential integrity enforcement at the data sources.

However, referential integrity constraints at a data source can affect nickname updates. Suppose that the federated database receives a transaction to insert data into nickname. When the federated server sends the insert to the data source, it violates a referential integrity constraint at that data source. DB2 UDB maps the resulting error to a DB2 UDB error.

Applications are responsible for referential integrity between data sources.

DB2 Information Integrator supports informational constraints that you can use to inform the query processor of referential integrity constraints.

Related concepts:

- "Informational constraints on nicknames" on page 181
- "How client applications interact with data sources" on page 197
- "Preserving statement atomicity in a federated system" on page 95

INSERT, UPDATE, and DELETE statements and large objects (LOBs)

The three types of LOB data types in DB2® UDB are character large objects (CLOBs), double-byte character large objects (DBCLOBs), and binary large objects (BLOBs).

With federation you can perform read operations on LOBs located in all the relational data sources. You can perform write operations on LOBs located in Oracle (Version 8 or higher) data sources using the NET8 wrapper.

Related concepts:

- "Federated LOB support" on page 204
- "LOB locators" on page 205
- "Restrictions on LOBs" on page 206

Related reference:

- "Altering long data types to varchar data types" on page 55

Preserving statement atomicity in a federated system

During update operations, federated systems always attempt to keep data in an atomic state at the completion of a DML statement. When data is in an atomic state, it is guaranteed to either be successfully processed or to remain unchanged.

When a client or application issues an INSERT, UPDATE, or DELETE statement on a nickname, a federated server internally processes that statement either as a single DML statement, or as a series of multiple DML statements. If a federated server must send multiple DML statements to a target data source for processing, it is possible that data atomicity might be compromised. To prevent compromising data atomicity, federated systems use data source savepoint APIs to monitor the series of multiple DML statements.

Some data sources or versions of data sources do not externalize savepoint APIs that the federated system can use. Under these circumstances, federated INSERT, UPDATE, or DELETE statements are run without the protection of savepoint APIs.

When an error occurs during federated insert, update, delete transactions, partial update results can occur at the data sources. To correct inconsistency problems, a federated system automatically performs an internal transaction rollback before it returns an sqlcode error to the applications.

The following data sources do not externalize savepoint APIs that the federated server can use:

- DB2® for Linux, UNIX®, and Windows®, Version 6 (or earlier)
- DB2 for iSeries™
- DB2 for VM and VSE
- DB2 for z/OS™ and OS/390®, Version 5 (or earlier)
- Informix®
- Microsoft® SQL Server
- ODBC
- Teradata

When an entire insert, update, delete transaction is pushed down to the data source for processing, the federated server assumes that the data source will preserve the statement atomicity if an error occurs.

When only part of the insert, update, delete transaction is pushed down to the data source for processing, the entire transaction is rolled back if an error occurs.

The IUD_APP_SVPT_ENFORCE server option controls this behavior. By default, this server option is set to 'Y', which rolls back transactions that encounter an error.

Using the SET SERVER OPTION statement to set the IUD_APP_SVPT_ENFORCE server option for the duration of a connection, will not affect on static SQL statements. Using the SET SERVER OPTION statement to set this server option will affect only dynamic SQL statements.

Scenario for the IUD_APP_SVPT_ENFORCE server option behavior:

Suppose that you create the nickname INFMX_UT for an Informix table named UT. The UT table has four integer columns, i1, i2, i3, and i4. The i1 column is a unique index column.

The UT table is empty. You issue an INSERT statement on nickname INFX_UT to insert the values 1, 22, 34, and 40 into Row 1 of the table. The statement is successful.

Then you issue a multiple row INSERT statement on the nickname INFX_UT to insert three rows of data:

- Row 2: 2, 37, 34, 55
- Row 3: 3 ,42, 59, 40
- Row 4: 1, 55, 62, 75

Because the data in the last row to be inserted violates the unique index requirement on column i1, the Informix server returns an error message to the federated system. The federated system returns the SQL error SQL0803N to your application. The SQL0803N error message describes the violation of unique indexes.

The following table lists the rows of the UT table after the inserts:

Table 8. Example of the Informix UT table

	Column (unique index)	Column	Column	Column
Row	**i1**	**i2**	**i3**	**i4**
Row 1	1	22	34	40
Row 2	2	37	34	55
Row 3	3	42	59	40

Default behavior of the IUD_APP_SVPT_ENFORCE server option:

By default, the IUD_APP_SVPT_ENFORCE server option is set to 'Y'. When set to 'Y', this server option performs an internal rollback of the entire transaction. Although the first two rows of data were successfully inserted, these rows are rolled back because the entire transaction is rolled back.

To change the default value of the IUD_APP_SVPT_ENFORCE server option, use the ALTER SERVER statement. This change applies to all data source objects that are accessed through the server you specify.

Alternate behavior of the IUD_APP_SVPT_ENFORCE server option:

When the IUD_APP_SVPT_ENFORCE server option is set to 'N' the transaction is not rolled back. The second and third rows of data remain in the table. Your application must handle the error recovery.

Related concepts:

- "Referential integrity in a federated system" on page 94

Related reference:

- Chapter 21, "Server options for federated systems," on page 219

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 UDB for z/OS table on one server, a DB2 UDB 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 nonrelational formats.

Tables and views in the federated database are *local objects*. Do not create nicknames for these objects, use the actual object name in the statements.

Remote objects are objects not located in the federated database. You need to create nicknames for these objects. For example:

- Tables and views in another DB2 for Linux, UNIX and Windows database or instance on the federated system
- Tables and views in another DB2 for Linux, UNIX and Windows database or instance on another system
- Objects located in data sources other than DB2 for Linux, UNIX and Windows, such as: Oracle, Sybase, Documentum, and ODBC

Procedure:

To take advantage of the capabilities of a federated system, you need to understand the following features:

- Be familiar with the SQL statements you can use with nickname.
- Know how to access new data source objects
- Learn when you use a pass-through session to access data sources directly
- Understand the advantages of using federated views to access heterogeneous data

WITH HOLD syntax

You can use the WITH HOLD syntax on a cursor defined on a nickname. However, you will receive an error if you try to use the syntax (with a COMMIT) and the data source does not support the WITH HOLD syntax.

Triggers

A nickname cannot be an update target in a trigger. You can include SELECT statements on nicknames in the trigger body. You cannot include INSERT, UPDATE, or DELETE statements on nicknames in the trigger body.

Related concepts:

- "Authorization privileges for INSERT, UPDATE, and DELETE statements" on page 93
- "Federated system INSERT, UPDATE, and DELETE restrictions" on page 94

Related tasks:

- "Reference data source objects by nicknames in SQL statements" on page 198
- "The SQL statements you can use with nicknames" on page 98
- "Accessing new data source objects" on page 102
- "Accessing data sources using pass-through sessions" on page 104
- "Accessing heterogeneous data through federated views" on page 105

Working with nicknames-details

This section describes insert, update, and delete operations on nicknames.

The SQL statements you can use with nicknames

With a federated system you can easily access data, regardless of where it is actually stored. To access data, you create 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 can 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, these are some of the issues that you do not need to be concerned with:

- The name of the object at the data source.
- The server on which the data source object resides.
- The data source type on which the object resides, such as Informix and Oracle
- The query language or SQL dialect that the data source uses
- The data type mappings between the data source and DB2 Information Integrator
- The function mappings between the data source and DB2 Information Integrator

All the underlying metadata stored in the federated database catalog provide the federated server with the information that it needs to process your queries. This metadata is gathered from the data sources when the federated server and database are setup and configured to access the data sources.

After the federated system is set up, you use the nicknames to query the data sources or to further enhance the federated system configuration.

The following table lists the SQL statements that support the use of nicknames:

Table 9. 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, the federated column options, or the informational constraints. 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
ALTER TABLE	Changes a remote table that was created through DB2 Information Integrator using transparent DDL. You cannot alter tables that were created natively on the data source. Can use informational constraints to add a referential integrity constraint to a nickname.	• SYSADM or DBADM • ALTER or CONTROL privilege on the nickname • ALTERIN privilege on the schema, if the schema name of the nickname exists
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

Table 9. Common SQL statements that support the use of nicknames (continued)

SQL statement	Description	Authorization required
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 (with SPECIFICATION ONLY clause)	Creates an index specification (metadata) which indicates to the query optimizer that a data source object has an index. No actual index is created, only the specification is created.	• SYSADM or DBADM • CONTROL or INDEX privilege on the underlying data source object — *and either* IMPLICIT_SCHEMA authority on the database, or CREATEIN privilege on the schema
CREATE TABLE (with the OPTIONS clause)	Creates a remote table through DB2 Information Integrator using transparent DDL.	• SYSADM or DBADM • CREATETAB privilege on the database and USE privilege on the tablespace —*and either* IMPLICIT_SCHEMA authority on the database, or CREATEIN privilege on the schema
CREATE TABLE (with the AS fullselect and DATA INITIALLY DEFERRED REFRESH clauses)	Creates a materialized query table using a fullselect that references a nickname.	• SYSADM or DBADM • CREATETAB privilege on the database and USE privilege on the tablespace —*and either* IMPLICIT_SCHEMA authority on the database, or CREATEIN privilege on the schema • CONTROL privilege on the table or view • SELECT privilege on the table or view and ALTER privilege if REFRESH DEFERRED is specified
CREATE VIEW	Creates a view that references one or more nicknames.	• SYSADM or DBADM • CONTROL or SELECT privilege on the nickname—*and either* IMPLICIT_SCHEMA authority on the database, or CREATEIN privilege on the schema
DELETE	Deletes rows from the data source object (such as a table or view) that has a nickname.	• 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

Table 9. Common SQL statements that support the use of nicknames (continued)

SQL statement	Description	Authorization required
DROP	Deletes an object, such as a nickname, federated view, index specification. The table, view, or index at the data source is not affected. When tables created using transparent DDL are dropped, the corresponding nickname for that table is also dropped.	• 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. Privileges at the data source must be granted separately.	• 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 has a nickname.	• 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 has a nickname. This statement is not supported with ODBC, Microsoft SQL Server, and nonrelational data sources.	• 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. Privileges at the data source must be revoked separately.	• SYSADM or DBADM • CONTROL privilege on the object
SELECT	Selects rows from the data source object (such as a table or view) that has a nickname.	• 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 has a nickname.	• 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 that the nickname refers to are only applied when the query is actually processed at the data source.

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 tasks:

- "Reference data source objects by nicknames in SQL statements" on page 198

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "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*
- "CREATE VIEW 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 must create nicknames for them using the CREATE NICKNAME statement.

Prerequisites:

The federated system must be configured to access the data source.

A server definition for the data source server on which the object resides must exist in the federated database. You create a server definition using the CREATE SERVER statement.

To insert, update, or delete data using a nickname, all of the following privileges must be true:

- The privileges held by the authorization ID of the statement must include the necessary SELECT, INSERT, UPDATE, and DELETE privileges on the nickname (for the federated database to accept the request)
- The user ID at the data source must have the necessary SELECT, INSERT, UPDATE, and DELETE privileges on the underlying table object (for the data source to accept the request)
- The user ID at the data source must be mapped to the authorization ID at the federated server through a user mapping.

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

Procedure:

The CREATE NICKNAME statement differs slightly for relational and nonrelational data sources.

For relational data sources the CREATE NICKNAME statement syntax is:

```
CREATE NICKNAME nickname_name FOR server_name."remote_schema"."object_name"
   OPTIONS (options_list)
```

nickname_name
A unique nickname for the data source object.

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.

FOR *server_name."remote_schema"."object_name"*
A three-part identifier for the remote data source object. If your data source does not support schemas, omit the schema from the CREATE NICKNAME statement.

- *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.

OPTIONS (*options_list***)**
Information about the nickname that enables the SQL Query Compiler and the wrapper to efficiently execute queries.

For some nonrelational data sources the CREATE NICKNAME statement syntax is:

```
CREATE NICKNAME nickname_name column_definition_list
   FOR SERVER server_name
   OPTIONS (options_list)
```

nickname_name
A unique nickname for the data source object.

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.

column_definition_list
A list of nickname columns and data types.

FOR SERVER *server_name*
The local name that you created for the remote server in the server definition information CREATE SERVER statement.

OPTIONS (*options_list***)**
Information about the nickname that enables the SQL Query Compiler and the wrapper to efficiently execute queries.

Related tasks:

- "Working with nicknames" on page 97

Related reference:

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

Accessing data sources using 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.

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. The administrative tasks you can perform depend on the data source. For example, for DB2 UDB 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 the SET PASSTHRU RESET command, the current pass-through session is closed and a new pass-through session is opened.

You can use the WITH HOLD syntax on a cursor defined in a pass-through session. However, you will receive an error if you try to use the syntax (with a COMMIT) and the data source does not support the WITH HOLD syntax.

Pass-through sessions do not support nonrelational data sources.

Related concepts:

- "Pass-through sessions" on page 10

- "Querying data sources directly with pass-through" on page 208

Related tasks:

- "Working with nicknames" on page 97

Related reference:

- "SET PASSTHRU statement" in the *SQL Reference, Volume 2*

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.

Prerequisites:

You must have one of the following authorizations to issue the CREATE VIEW statement:

- SYSADM or DBADM
- For each nickname in any fullselect, both:
  - CONTROL or SELECT privilege on the underlying table or view
  - One of the following authorities or privileges:
    - 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.

Restrictions:

Federated views with UNION ALL statements are read-only views.

Federated views that include more than one nickname in the FROM clause are read-only views.

Federated views that include only one nickname in the FROM clause might be read-only views.

- If the nickname in the FROM clause is to a nonrelational data source, the federated view is read-only.
- If you include other nicknames as predicates or as subqueries when you create the view, the federated view can be updated.

Procedure:

Use the CREATE VIEW statement to create a federated view.

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 can be mapped to a different remote authorization ID by a user mapping.

Creating a federated view that merges similar data from several data source objects:

Suppose that 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 VIEW FV1
   AS SELECT * FROM ORA_EU_CUST
   UNION ALL
   SELECT * FROM SYB_AS_CUST
   UNION ALL
   SELECT * FROM INFMX_SA_CUST
```

Joining data to create a federated view:

Suppose that 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 that those customers made. Each table has a column containing the customer number (CUST_NO). The syntax to create a federated view from these two nicknames that joins this data is:

```
CREATE 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
```

Related tasks:

- "Creating and using federated views" on page 201
- "Working with nicknames" on page 97

Related reference:

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

Creating a nickname on a nickname

Occasionally, you might need to create a nickname on a nickname.

Procedure:

Suppose that 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, install DB2 Information Integrator.
2. Configure the Windows federated server to access Excel data sources.
3. On the Windows federated server, create a nickname for the Excel spreadsheet.
4. On the AIX federated server, install DB2 Information Integrator.
5. Configure the AIX federated server to access DB2 family data sources.
6. On the AIX federated server, create a nickname for the Excel nickname on the Windows federated server.

Selecting data in a federated system

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
- A combination of remote and local data sources

To select data from the data sources, use the nicknames for the data source objects in the SELECT statement.

Prerequisites:

To select data using a nickname, all of the following privileges must be true:

- The privileges held by the authorization ID of the statement must include the SELECT privilege on the nickname (for the federated database to accept the request).
- The user ID at the data source must have the SELECT privilege on the underlying table object (for the data source to accept the request).
- The user ID at the data source must be mapped to the authorization ID at the federated server through a user mapping.

Procedure:

The federated database is a local data source. Tables and views in the federated database are local objects. You do not create nicknames for these objects, you use the actual object name in the SELECT statements.

Remote data sources include: another DB2 for Linux, UNIX, and Windows database instance on the federated server, another DB2 for Linux, UNIX, and Windows database instance on another server, and data sources other than DB2 for Linux, UNIX, and Windows. Objects that reside on remote data sources are remote objects.

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. This configuration is depicted in the following figure.

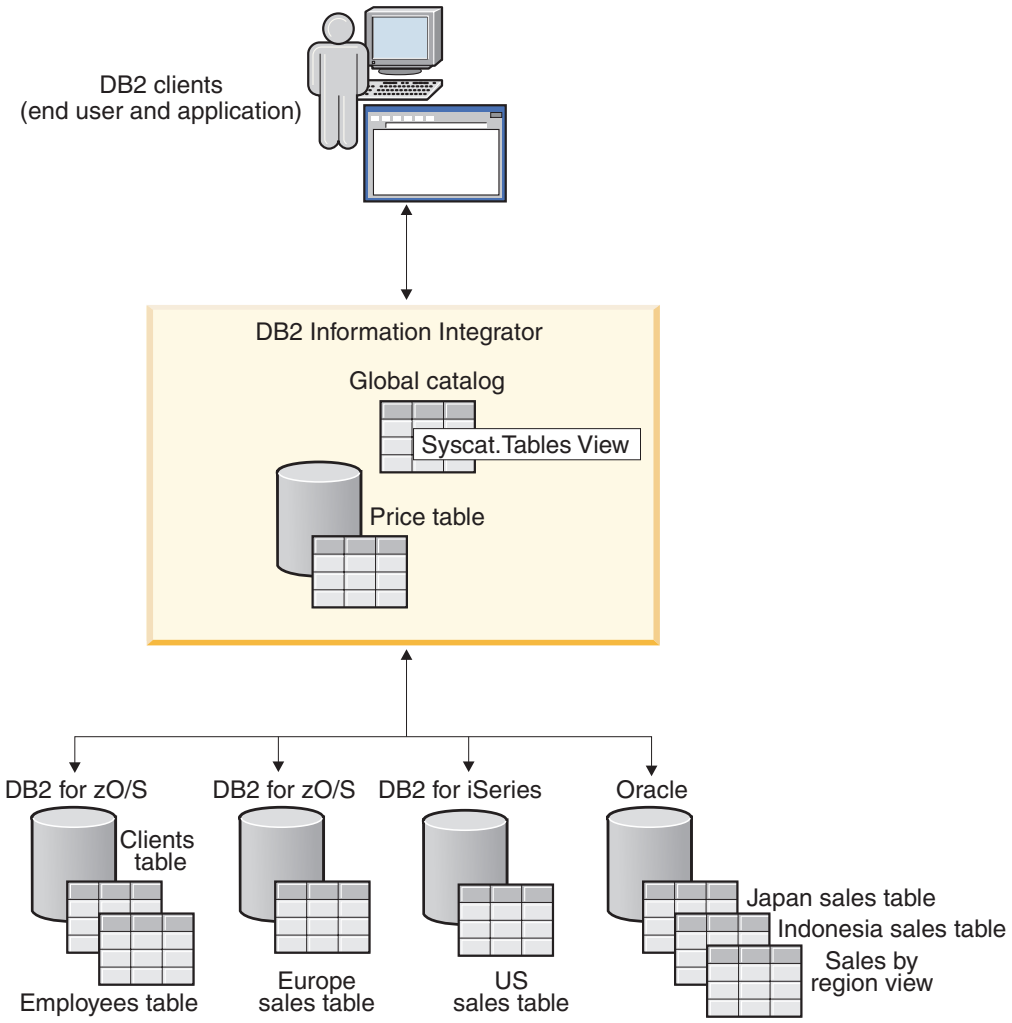


Figure 3. Sample federated system with DB2 and Oracle data sources

The sales tables include columns that record the customer number (CUST_NO), the quantity ordered (QUANTITY), and the product number ordered (PROD_NO). You also have a local table in the federated database that contains price information. The price 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 tables, as shown in the following figure. The TYPE column indicates the type of object, such as nickname (N), local table (T), or view (V).

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 Tables

TABNAME	TYPE
PRICES	T
FED_PRICES	N
Z_EU_SALES	N
iS_US_SALES	N
ORA_JAPANSALES	N
ORA_REGIONSALES	N
.....	

Figure 4. Tables and nicknames for sample queries

The following are SELECT statement examples using the sample federated system described above.

Example: 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. This query uses a SELECT statement with an ORDER BY clause to list the sales in Europe and sorts the list by customer number:

```
SELECT CUST_NO, PROD_NO, QUANTITY
   FROM Z_EU_SALES
   ORDER BY CUST_NO
```

Example: 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 that 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=PRICES.PROD_NO
   ORDER BY sales.CUST_NO
```

Example: 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 that 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 issue INSERT, UPDATE, and DELETE statements on data source objects using nicknames. The following sections include examples for performing these operations.

Inserting data into data source objects

To insert data into data sources, use the nicknames for the data source objects in the INSERT statement.

Prerequisites:

To insert data using a nickname, all of the following privileges must be true:

- The privileges held by the authorization ID of the statement must include the INSERT privilege on the nickname (for the federated database to accept the request).
- The user ID at the data source must have the INSERT privilege on the underlying table object (for the data source to accept the request).
- The user ID at the data source must be mapped to the authorization ID at the federated server through a user mapping.

Restrictions:

Federation does not support INSERT operations with some data sources, see Federated system INSERT, UPDATE, and DELETE restrictionsFederated system INSERT, UPDATE, and DELETE restrictions

Procedure:

Suppose an Informix table consists of two columns. The first column contains INTEGER data and the second column contains VARCHAR data (up to 20 characters). The nickname *infx_table_nn* is registered with the federated server for the Informix table.

You can issue INSERT, UPDATE, and DELETE statements on the Informix table using the *infx_table_nn* nickname. For example, to insert a new row of information into the Informix table, the statement is:

```
INSERT INTO db2user1.infx_table_nn VALUES(1,'Walter')
```

Related concepts:

- "Federated system INSERT, UPDATE, and DELETE restrictions" on page 94

Related tasks:

- "Selecting data in a federated system" on page 107
- "Updating data in data source objects" on page 111
- "Deleting data from data source objects" on page 112

Related reference:

- "INSERT statement" in the *SQL Reference, Volume 2*

Updating data in data source objects

To update data into data sources, use the nicknames for the data source objects in the UPDATE statement.

Prerequisites:

To update data using a nickname, all of the following privileges must be true:

- The privileges held by the authorization ID of the statement must include the UPDATE privilege on the nickname (for the federated database to accept the request)
- The user ID at the data source must have the UPDATE privilege on the underlying table object (for the data source to accept the request)
- The user ID at the data source must be mapped to the authorization ID at the federated server through a user mapping.

Restrictions:

Federation does not support UPDATE operations with some data sources, see Federated system INSERT, UPDATE, and DELETE restrictionsFederated system INSERT, UPDATE, and DELETE restrictions

Procedure:

Suppose an Informix table consists of two columns. The first column contains INTEGER data and the second column contains VARCHAR data (up to 20 characters). The nickname *infx_table_nn* is registered with the federated server for the Informix table.

You can issue INSERT, UPDATE, and DELETE statements on the Informix table using the *infx_table_nn* nickname. For example, to update a row of information the Informix table, the statement is:

```
UPDATE db2user1.infx_table_nn SET c2='Bill' WHERE c1=2
```

Related concepts:

- "Federated system INSERT, UPDATE, and DELETE restrictions" on page 94

Related tasks:

- "Selecting data in a federated system" on page 107
- "Inserting data into data source objects" on page 110
- "Deleting data from data source objects" on page 112

Related reference:

- "UPDATE statement" in the *SQL Reference, Volume 2*

Deleting data from data source objects

To delete data from data sources, use the nicknames for the data source objects in the DELETE statement.

Prerequisites:

To delete data using a nickname, all of the following privileges must be true:

- The privileges held by the authorization ID of the statement must include the DELETE privilege on the nickname (for the federated database to accept the request)
- The user ID at the data source must have the DELETE privilege on the underlying table object (for the data source to accept the request)
- The user ID at the data source must be mapped to the authorization ID at the federated server through a user mapping.

Restrictions:

Federation does not support DELETE operations with some data sources, see Federated system INSERT, UPDATE, and DELETE restrictions

Procedure:

Suppose an Informix table consists of two columns. The first column contains INTEGER data and the second column contains VARCHAR data (up to 20 characters). The nickname *infx_table_nn* is registered with the federated server for the Informix table.

You can issue INSERT, UPDATE, and DELETE statements on the Informix table using the *infx_table_nn* nickname. For example, to delete a row of information the Informix table, the statement is:

```
DELETE FROM infx_table_nn WHERE c1=3
```

Related concepts:

- "Federated system INSERT, UPDATE, and DELETE restrictions" on page 94

Related tasks:

- "Selecting data in a federated system" on page 107
- "Inserting data into data source objects" on page 110
- "Updating data in data source objects" on page 111

Related reference:

- "DELETE statement" in the *SQL Reference, Volume 2*

Assignment semantics in a federated system

When you assign data to a nickname column, the data type might change based on the assignment rules DB2 Information Integrator uses. You should understand the assignment rules so you get the results that you expect.

The rules for determining the target data type of an assignment to a nickname column are:

- Determine the local source type: The local source type is determined by the local column type and the local result type of the expressions. If the source is constant, the local source type is the same as the type of the constant.
- Determine the target type:
  - If the assignment source has no type, such as parameter markers and NULLs, then the target type is MIN(local_target_type, remote_target_type), where local_target_type is the updated column local data type and remote_target_type is the updated column data source data type. The remote_target_type refers to the default forward type mapping type of the remote target column's data type.
  - If the assignment source is not NULL or parameter markers, then the target type is MIN(local_target_type, remote_target_type, local_source_type).

The definition of MIN(type1, type2)

- Type1 and type2 are not exactly the same.
- MIN(type1,type2) = MIN(type2, type1)
- MIN(type1, type2) = remote_target_type(local_target_type), when MIN(type1, type2) = DECIMAL(0,0)
- BLOB is only compatible with BLOB, so MIN(BLOB(x), BLOB(y))=BLOB(z) where z=min(x,y)
- TIME and DATE data types are not compatible.
- Datetime types and character strings are compatible.
- In Unicode databases, character strings and graphic strings are compatible.

The following tables list the minimum of two data types for numeric, character string, graphic string, and date and time data types.

Table 10. Numeric data types

type1	type2	MIN(type1, type2)
SMALLINT	SMALLINT or INTEGER or BIGINT or REAL or DOUBLE	SMALLINT
INTEGER	BIGINT or REAL or DOUBLE	INTEGER
BIGINT	REAL or DOUBLE	BIGINT
REAL	DOUBLE	REAL
DECIMAL(w,x)	SMALLINT	DECIMAL(p,0) where p=w-x, if p<5; SMALLINT, otherwise
DECIMAL(w,x)	INTEGER	DECIMAL(p,0) where p=w-x, if p<11; INTEGER, otherwise
DECIMAL(w,x)	BIGINT	DECIMAL(p,0) where p=w-x, if p<19; BIGINT, otherwise
DECIMAL(w,x)	DECIMAL(y,z)	DECIMAL(p,s) where p=min(w,y)+min(w-x,y-z), s=min(x,z)
DECIMAL(w,x)	DOUBLE or REAL	DECIMAL(w,x)

The following table lists the minimum of two data types for character string data types.

Table 11. Character string data types

type1	type2	MIN(type1, type2)
CHAR(x)	CHAR(y) or VARCHAR(y) or LONG VARCHAR or CLOB(y)	CHAR(z) where z=min(x,y)
VARCHAR(x)	VARCHAR(y) or LONG VARCHAR or CLOB(y)	VARCHAR(z) where z=min(x,y)
LONG VARCHAR	CLOB(y)	LONG VARCHAR where x>32700, CLOB(x) where x<=32700
CLOB(x)	CLOB(y)	CLOB(z) where z=min(x,y)

The following table lists the minimum of two data types for graphic string data types.

Table 12. Graphic string data types

type1	type2	MIN(type1, type2)
GRAPHIC(x)	GRAPHIC(y) or VARGRAPHIC(y) or LONG VARGRAPHIC or DBCLOB(y)	GRAPHIC(z) where z=min(x,y)
VARGRAPHIC(x)	VARGRAPHIC(y) or LONG VARGRAPHIC or DBCLOB(y)	VARGRAPHIC(z) where z=min(x,y)
LONG VARGRAPHIC	DBCLOB(y)	LONG VARGRAPHIC where x>32700, DBCLOB(x) where x<=32700
DBCLOB(x)	DBCLOB(y)	DBCLOB(z) where z=min(x,y)

The following table lists the minimum of two data types for data and time data types.

Table 13. Date and time data types

type1	type2	MIN(type1, type2)
DATE	TIMESTAMP	DATE
TIME	TIMESTAMP	TIME

If the data of data type CHAR you are inserting is shorter than the target length, the data source pads the rest of the column.

If you are inserting data of DATE or TIME data type into a remote column of TIMESTAMP data type, the data source pads the rest of the column.

Assignment semantics in a federated system - examples

Table 14 shows several examples of the application of federated assignment semantics in queries given a local type and remote type.

Table 14. Examples of assignment semantics

Local type	Remote type	Your query	Generated remote query
FLOAT	INTEGER	set c1=123.23	set c1=INTEGER(123.23)

Table 14. Examples of assignment semantics (continued)

Local type	Remote type	Your query	Generated remote query
INTEGER	FLOAT	set c1=123.23	set c1=INTEGER(123.23)
FLOAT	INTEGER	set c1=123	set c1=123
CHAR(10)	CHAR(20)	set c1='123'	set c1='123' ('123' has a type VARCHAR(3) and it's the shortest of all)
CHAR(10)	CHAR(20)	set c1=char23col	set c1=CHAR(char23col, 10)
CHAR(10)	CHAR(20)	set c1=expr1	• set c1=expr1 -- if expr1 returns char(n) n<= 10 • set c1=CHAR(expr1, 10) if expr1 returns char(n) n>10
TIMESTAMP	DATE	set c1= current timestamp	set c1=DATE(current timestamp)

Chapter 9. Monitoring a federated system

This chapter describes how to monitor the health of nicknames and servers and how to snapshot monitor query fragments.

This chapter contains:

Health indicators for federated nicknames and servers

You can use health indicators in the DB2® Health Center to monitor the status of your federated nicknames and servers. The health indicator for nicknames is db.fed_nicknames_op_status. The health indicator for server definitions is db.fed_servers_op_status. The federated health indicators are installed when the health monitor is installed.

By default, the Health Center does not activate the federated health indicators. You must activate the indicators.

When the state of a nickname or server is not normal, the health indicators issue an alert. You can view the results of monitoring by using the Health Center or the command line.

Federated servers that use AIX®, HP-UX, Linux, Microsoft® Windows®, and Solaris operating systems support the health indicators.

Table 15 describes the health indicators for federated nicknames and servers.

Table 15. Nickname and server health indicators

Health indicator	Description
db.fed_nicknames_op_status	Indicates the aggregate health of all the relational nicknames defined in a database on a DB2 UDB federated server. Alerts you if a nickname is invalid. Provides details about the invalid nicknames and recommends actions that you can take to repair them.
db.fed_servers_op_status	Indicates the aggregate health of all the federated servers defined in a database on a DB2 UDB federated server. Alerts you if a server is unavailable. Provides details about the unavailable servers and recommends actions that you can take to make them available.

The health indicators can evaluate the following data sources:

- DB2 family (DRDA)
- Excel
- Informix®
- Microsoft SQL Server
- ODBC
- Oracle (NET8)
- Sybase (CTLIB)
- Table-structured files
- Teradata
- XML (root nicknames only)

Related concepts:

- "Introduction to the health monitor" in the *System Monitor Guide and Reference*

Related tasks:

- "Monitoring the health of federated nicknames and servers" on page 119
- "Activating the federated health indicators" on page 118

Related reference:

- "db2hc - Start Health Center Command" in the *Command Reference*
- "db.fed_nicknames_op_status - Nickname Status health indicator" in the *System Monitor Guide and Reference*
- "db.fed_servers_op_status - Data Source Server Status health indicator" in the *System Monitor Guide and Reference*

Activating the federated health indicators

To monitor the health of nicknames and servers, you must activate the federated health indicators.

The health indicator for nicknames is db.fed_nicknames_op_status. The health indicator for server definitions is db.fed_servers_op_status.

Procedure:

To activate the federated health indicators open the DB2 Health Center and configure the health indicators.

Related tasks:

- "Configuring health indicators using Health Center" in the *System Monitor Guide and Reference*
- "Monitoring the health of federated nicknames and servers" on page 119

Monitoring the health of federated nicknames and servers

Monitoring nickname and server status can help you determine and resolve problems in your federated system. You can monitor the status of federated nicknames and servers by using health indicators in the Health Center.

You can view the results of monitoring by using the Health Center or the command line. Use the DB2 Control Center or DB2 command line processor to resolve the problems that are identified by the health indicators.

Prerequisites:

- Ensure that SELECT privileges on the nicknames are defined on the DB2 federated server.
- Set the FEDERATED database manager configuration parameter to YES.
- If the data source requires authentication, the data source must have user mappings from the Health Monitor's ID. The Health Monitor uses this mapping to connect to the data source.

Restrictions:

The health indicators cannot evaluate the following data sources:

- BioRS
- BLAST
- Documentum
- Entrez
- Extended Search
- HMMER
- Web services
- WebSphere Business Integration
- XML (nonroot nicknames)

Procedure:

To do this task from the DB2 Control Center:

1. Open the Health Center.
2. Open the Recommendation Advisor to view recommendations for how to resolve the invalid nicknames or unavailable servers.

To do this task from the command line, issue the GET HEALTH SNAPSHOT command.

Related concepts:

- "Introduction to the health monitor" in the *System Monitor Guide and Reference*
- "Health indicators for federated nicknames and servers" on page 117

Related tasks:

- "Resolving alerts using the Health Center" in the *System Monitor Guide and Reference*

Related reference:

- "Monitoring the health of federated nicknames and servers - example" on page 120

Monitoring the health of federated nicknames and servers - example

The following example shows the health snapshot of a database named fedhi. The federated health indicator names are db.fed_nicknames_op_status and db.fed_servers_op_status. In this example both health indicators are in normal states. Normal means the nicknames and servers are valid.

```
GET HEALTH SNAPSHOT FOR DATABASE ON fedhi:

             Database Health Snapshot

Snapshot timestamp                          = 02/10/2004 12:10:55.063004

Database name                               = FEDHI
Database path                               = C:\DB2\NODE0000\SQL00006\
Input database alias                        = FEDHI
Operating system running at database server= NT
Location of the database                    = Local
Database highest severity alert state       = Attention

Health Indicators:

    Indicator Name                             = db.fed_servers_op_status
       Value                                   = 0
       Evaluation timestamp                    = 02/10/2004 12:09:10.961000
       Alert state                             = Normal

    Indicator Name                             = db.fed_nicknames_op_status
       Value                                   = 0
       Evaluation timestamp                    = 02/10/2004 12:09:10.961000
       Alert state                             = Normal

    Indicator Name                             = db.db_op_status
       Value                                   = 0
       Evaluation timestamp                    = 02/10/2004 12:08:10.774000
       Alert state                             = Normal

    Indicator Name                             = db.sort_shrmem_util
       Value                                   = 0
       Unit                                    = %
       Evaluation timestamp                    = 02/10/2004 12:08:10.774000
       Alert state                             = Normal


    Indicator Name                             = db.spilled_sorts
       Value                                   = 0
       Unit                                    = %
       Evaluation timestamp                    = 02/10/2004 12:09:10.961000
       Alert state                             = Normal
```

Related concepts:

- "Health indicators for federated nicknames and servers" on page 117

Related tasks:

- "Monitoring the health of federated nicknames and servers" on page 119

Snapshot monitoring of federated systems - Overview

You can use the snapshot monitor to capture information about federated data sources and any connected applications at a specific time. Snapshots are useful for determining the status of a federated system. Taken at regular intervals, snapshots are also useful for observing trends and foreseeing potential problems.

Related concepts:

- "Snapshot monitor" in the *System Monitor Guide and Reference*

Related tasks:

- "Monitoring federated query fragments" on page 121

Related reference:

- Chapter 31, "Federated database systems monitor elements," on page 297
- "Snapshot monitoring of federated query fragments - example" on page 121

Monitoring federated query fragments

By monitoring query fragments, you can see how your federated system is performing. To help you understand how your federated system is processing a query, you can get a snapshot of the remote query segments.

Procedure:

To monitor query fragments, issue the GET SNAPSHOT FOR DYNAMIC SQL ON <dbname> command, where *dbname* is the name of the local database on the federated server.

Related concepts:

- "Snapshot monitoring of federated systems - Overview" on page 121

Related reference:

- "GET SNAPSHOT Command" in the *Command Reference*
- "Snapshot monitoring of federated query fragments - example" on page 121

Snapshot monitoring of federated query fragments - example

The following example shows the dynamic SQL snapshot output for a query fragment sent to a remote Oracle data source called ORACLE817. It was generated from the statement GET SNAPSHOT FOR DYNAMIC SQL ON LOCAL_FEDERATED_DATABASE. The results provide information for each of the remote queries and any queries in the statement cache. Since buffer pool information is not applicable to remote queries, the snapshot did not collect any buffer pool information.

```
    Dynamic SQL Snapshot Result

Number of executions              = 1
Number of compilations            = 1
Worst preparation time (ms)       = 215
Best preparation time (ms)        = 215
Internal rows deleted             = 0
```

```
Internal rows inserted             = 0
Rows read                          = 25412
Internal rows updated              = 0
Rows written                       = 25410
Statement sorts                    = 0
Buffer pool data logical reads     = Not Collected
Buffer pool data physical reads    = Not Collected
 Buffer pool temporary data logical reads   = Not Collected
 Buffer pool temporary data physical reads  = Not Collected
 Buffer pool index logical reads    = Not Collected
 Buffer pool index physical reads   = Not Collected
 Buffer pool temporary index logical reads  = Not Collected
 Buffer pool temporary index physical reads = Not Collected
Total execution time (sec.ms)      = 20.229786
Total user cpu time (sec.ms)       = 10.080000
Total system cpu time (sec.ms)     = 0.520000
Statement text                     = [ORACLE817]SELECT A0.C1,A0.C2 FROM ORA_T A0
                                     WHERE A0.C3 = :H0
```

Related concepts:

- "Snapshot monitoring of federated systems - Overview" on page 121

Related tasks:

- "Monitoring federated query fragments" on page 121

Related reference:

- "GET SNAPSHOT Command" in the *Command Reference*

Chapter 10. Unicode support for federated data sources

This chapter describes Unicode support in a federated system.

This chapter contains:

Unicode support for federated systems

Relational and nonrelational wrappers and user-defined functions can run on a Unicode database (UTF-8 database). The Unicode database provides federated server environments that are platform independent. The Unicode database can manipulate data that is stored in various code pages on different data sources.

The wrappers and user-defined functions that support Unicode are:

- Relational wrappers
  - DRDA®
  - Informix®
  - MS SQL Server
  - ODBC
  - OLE DB
  - Oracle
  - Sybase
  - Teradata
- Nonrelational wrappers and user-defined functions
  - BioRS wrapper
  - BLASTwrapper
  - Documentum wrapper
  - Entrez wrapper
  - Excel wrapper
  - HMMER wrapper
  - IBM® Lotus® Extended Search wrapper
  - KEGG user-defined functions
  - MQ user-defined functions
  - Table-structured file wrapper
  - Web services user-defined functions
  - Web services wrapper
  - WebSphere® Business Integration wrapper
  - XML wrapper

In Figure 5 a company has branch offices in different countries. Each branch office stores customer data with its own databases in their own code page. The Microsoft® SQL Server database stores data in code page A. The Oracle database stores data in code page B. Code page A and code page B are in different territories. To integrate the data from the different territories, the company can set the federated database's code page to Unicode. The company can then join the tables to see the total number of purchase orders, regardless of territory.

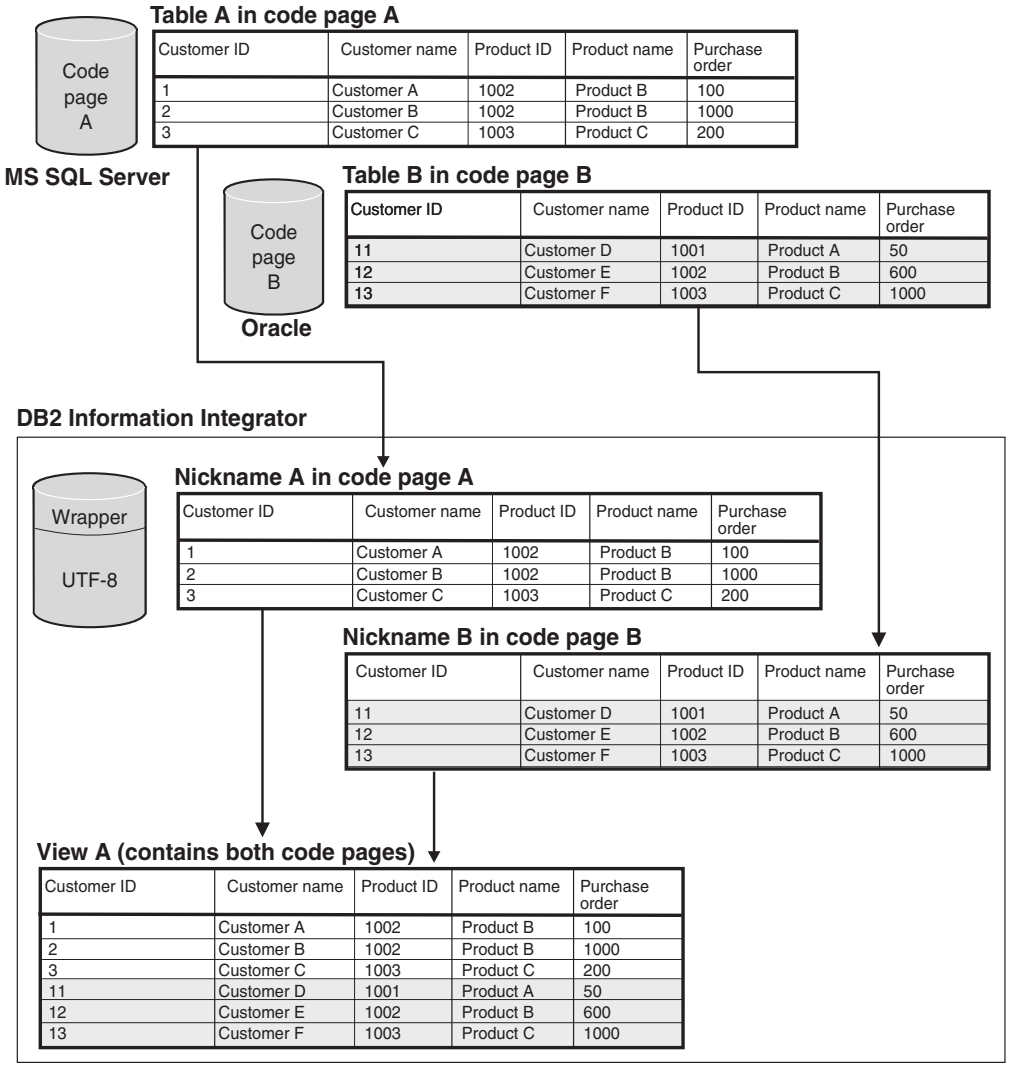


Figure 5. Unicode example

Related tasks:

- "Specifying the client code page for Unicode support of Microsoft SQL Server and ODBC data sources" on page 125
- "Specifying the file code page for Unicode support of table-structured file data sources" on page 126

Related reference:

- "Unicode default forward data type mappings - NET8 wrapper" on page 287
- "Supported Unicode code pages for the MSSQL and ODBC wrapper CODEPAGE option" on page 126

- "Unicode default reverse data type mappings - NET8 wrapper" on page 287
- "Unicode default forward data type mappings - Sybase wrapper" on page 288
- "Unicode default reverse data type mappings - Sybase wrapper" on page 288
- "Unicode default forward data type mappings - ODBC wrapper" on page 289
- "Unicode default reverse data type mappings - ODBC wrapper" on page 289
- "Unicode default forward data type mappings - Microsoft SQL Server wrapper" on page 290
- "Unicode default reverse data type mappings - Microsoft SQL Server wrapper" on page 290
- "Specifying the file code page for Unicode support of table-structured file data sources - example" on page 127

Specifying the client code page for Unicode support of Microsoft SQL Server and ODBC data sources

To ensure correct code page conversion for Microsoft SQL Server and ODBC data sources, you must specify the client code page if the code page differs from the federated database code page.

Procedure:

Issue a CREATE SERVER statement with the CODEPAGE option set to the value of the client code page.

For example, if the data source is Microsoft SQL Server and the federated server is on Windows and the default system locale of the operating system is set to Japanese (Shift-JIS), the CODEPAGE server option must be set to either 943 (Shift-JIS) or 1202 (UTF-16LE).

To specify the 1202 code page for the Microsoft SQL server data source named FEDSERVERW, issue the following statement:

```
CREATE SERVER FEDSERVERW TYPE MSSQLSERVER VERSION 2000 WRAPPER MSSQLODBC3
       OPTIONS(NODE 'SAMPLE', DBNAME 'TESTDB', CODEPAGE '1202');
```

If the data source is Microsoft SQL Server and the federated server is running on UNIX and the AppCodePage or IANAAppCodePage setting of the DataDirect Connect client is 6 (Shift-JIS), the CODEPAGE server option must be set to either 943 (Shift-JIS) or 1208 (UTF-8).

To specify the 1208 code page for the Microsoft SQL server data source named FEDSERVERU, issue the following statement:

```
CREATE SERVER FEDSERVERU TYPE MSSQLSERVER VERSION 2000 WRAPPER MSSQLODBC3
       OPTIONS(NODE 'SAMPLE', DBNAME 'TESTDB', CODEPAGE '1208');
```

Related concepts:

- "Unicode support for federated systems" on page 123

Related reference:

- "CREATE SERVER statement" in the *SQL Reference, Volume 2*
- Chapter 21, "Server options for federated systems," on page 219
- "Supported Unicode code pages for the MSSQL and ODBC wrapper CODEPAGE option" on page 126

Supported Unicode code pages for the MSSQL and ODBC wrapper CODEPAGE option

Valid code page values are those that DB2 Universal Database supports plus those shown in Table 16.

Table 16. Supported Unicode code pages for the MSSQL and ODBC wrapper CODEPAGE option

CODEPAGE option value	Description
1200	Codepage1200 - UCS-2 (big-endian)
1202	Codepage1202 - UCS-2 (little-endian)
1208	Codepage1208 - UTF-8
1232	Codepage1232 - UTF-32 (big-endian)
1234	Codepage1234 - UTF-32 (little-endian)

Related tasks:

- "Specifying the client code page for Unicode support of Microsoft SQL Server and ODBC data sources" on page 125

Related reference:

- "Supported territory codes and code pages" in the *Administration Guide: Planning*

Specifying the file code page for Unicode support of table-structured file data sources

To ensure correct code page conversion for table-structured file data sources data sources, you must specify the file code page if the code page differs from the federated database code page.

Restrictions:

You can use the CODEPAGE option only in a Unicode federated database.

Procedure:

Issue the CREATE NICKNAME statement with the CODEPAGE option set to the code page of the data in the table-structured file. Valid values are those DB2 Universal Database supports. The default value is the code page of the DB2 Universal database federated database.

Related concepts:

- "Unicode support for federated systems" on page 123

Related reference:

- "CREATE NICKNAME statement" in the *SQL Reference, Volume 2*
- "Supported territory codes and code pages" in the *Administration Guide: Planning*
- "Specifying the file code page for Unicode support of table-structured file data sources - example" on page 127
- "CREATE NICKNAME statement syntax - Table-structured file wrapper" in the *IBM DB2 Information Integrator Data Source Configuration Guide*

Specifying the file code page for Unicode support of table-structured file data sources - example

The code page of the data in a file named DRUGDATA1.TXT is 943. To specify the code page of a table-structured file as 943, issue the following CREATE NICKNAME statement:

```
CREATE NICKNAME DRUGDATA1(Dcode Integer NOT NULL, Drug CHAR(20),
         Manufacutuer CHAR(20))
       FOR SERVER biochem_lab
       OPTIONS(FILE_PATH '/usr/pat/DRUGDATA1.TXT',CODEPAGE '943',
      COLUMN_DELIMITER '.',
       SORTED 'Y', KEY_COLUMN 'DCODE', VALIDATE_DATA_FILE 'Y');
```

Related tasks:

- "Specifying the file code page for Unicode support of table-structured file data sources" on page 126

Errors when remote and federated code point sizes are different

When the code point size differs between the federated database and the remote data source, you can get incorrect data returned or insertion failures.

If the federated database has a larger code point size than the remote data source, the federated server can truncate data that you select from the remote data source. Data is truncated if the character string conversion results in a larger number of bytes than the size of the nickname column. Remaining bytes are blank. Also, you can insert data that is larger than the nickname column size. This type of insertion succeeds if conversion results in a number of bytes that is smaller than, or equal to, the remote column size.

If the federated database has a smaller code point size than the remote data source, insertion of data can fail. Insertions fail if the character string conversion results in a larger number of bytes than the size of the remote data source column.

Ensure that the code point sizes between the federated database and the remote data source do not differ enough to cause the previously mentioned errors.

Part 3. Performance

Chapter 11. Tuning the performance of a federated system

Performance problems can originate at either the 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 might require tuning queries, applications, configuration parameters, and network usage to solve these problems.

This chapter includes discussions on:

Publications about federated performance

The following IBM documents contain detailed information about performance tuning:

- *Data Federation with IBM DB2 Information Integrator V8.1*, at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg247052.html?Open
- "Using the federated database technology of IBM DB2 Information Integrator", at ftp://ftp.software.ibm.com/software/data/pubs/papers/iifed.pdf
- "DB2 Information Integrator XML Wrapper Performance", at ftp://ftp.software.ibm.com/software/data/pubs/papers/db2iixmlwrapper.pdf

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 nickname 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 UDB 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 UDB 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.

Pushdown analysis is only performed on relational data sources. Nonrelational data sources use the request-reply-compensate protocol.

The following figure illustrates the steps performed by the SQL Compiler when it processes a query.

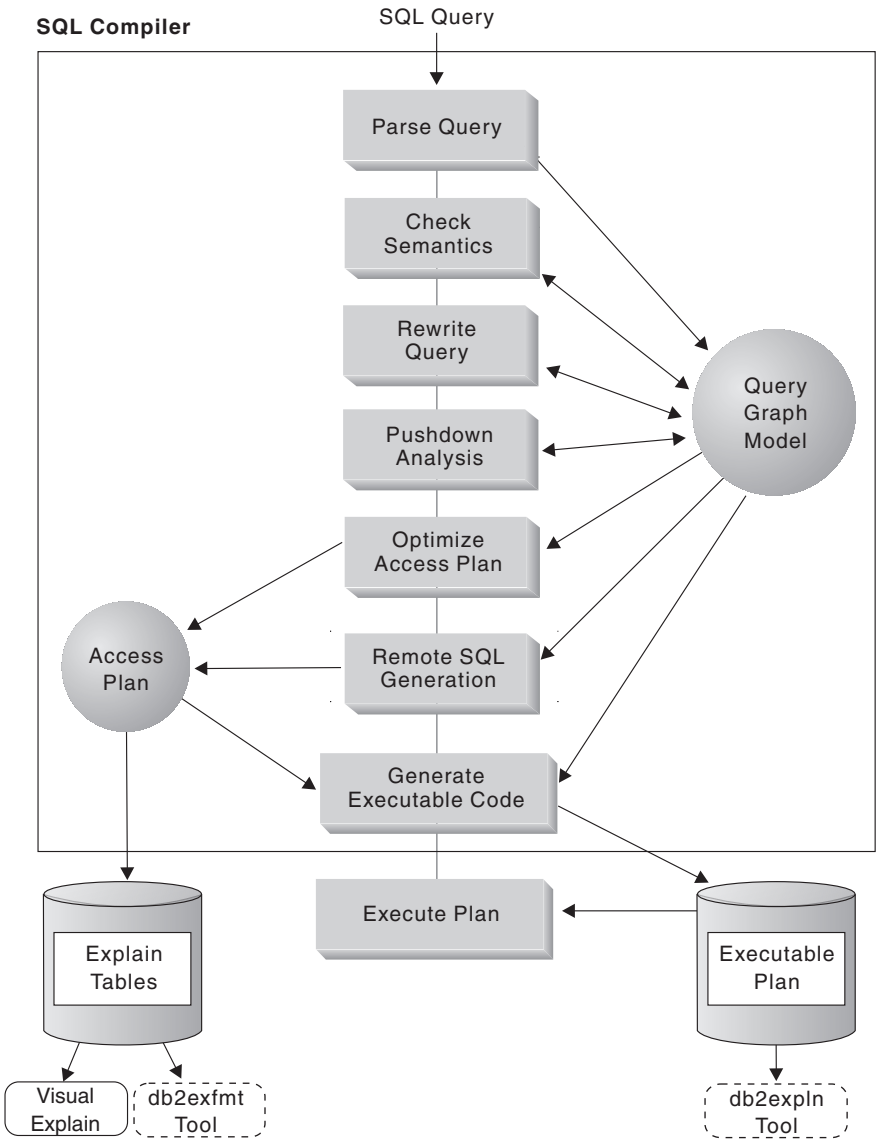


Figure 6. SQL Compiler query analysis flowchart

The query optimizer generates local and remote access plans for processing a query fragment, based on resource cost. DB2 UDB 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 UDB submits these fragments to the data sources. After the data sources process the fragments,

the results are retrieved and returned to DB2 UDB. If DB2 UDB performed any part of the processing, it combines its results with the results retrieved from the data source. DB2 UDB then returns all results to the client.

The primary task of pushdown analysis is to determine which operations can be remotely evaluated. Pushdown analysis does this based on the SQL statement it receives and the capabilities of the remote data source. Based on this analysis, the query optimizer evaluates the alternatives and chooses the access plan based on cost. 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 nonrelational sources, the wrappers use the request-reply-compensate protocol.

Related concepts:

- "Pushdown analysis" on page 133

Related tasks:

- "Request-reply-compensate protocol" in the *IBM DB2 Information Integrator Wrapper Developer's Guide*
- "Global optimization" on page 144

Pushdown analysis

Pushdown analysis is performed on relational data sources. Nonrelational use the request-reply-compensate protocol. 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 using 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 LIKE '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 LIKE 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:

- "Server characteristics affecting pushdown opportunities" on page 134
- "Nickname characteristics affecting pushdown opportunities" on page 138
- "Query characteristics affecting pushdown opportunities" on page 140

Related tasks:

- "Request-reply-compensate protocol" in the *IBM DB2 Information Integrator Wrapper Developer's Guide*

Pushdown analysis-details

Server characteristics affecting pushdown opportunities

The following sections contain data source-specific factors that can affect pushdown opportunities.

For relational data sources, these factors exist because you use the DB2® SQL dialect to submit queries and the DB2 dialect might 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 might require that the operation take place at the federated server.

The factors that affect pushdown opportunities for nonrelational data sources are different than the factors that affect pushdown opportunities for relational data sources. The SQL dialect is not a factor for most nonrelational data sources, since they do not use SQL.

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 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.

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 might 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 UDB 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 UDB 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 can 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 statements 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 such as DB2 UDB.

Administrators can create federated databases with a particular collating sequence that matches the data source collating sequence. This approach can speed performance if all data sources use the same collating sequence or if most or all column functions are directed to 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 Linux, 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 on 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, do not apply blank-padded comparison semantics like DB2 for Linux, UNIX, and Windows does. This padding difference can cause unexpected results. 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.
- DB2_MAXIMAL_PUSHDOWN. This option specifies the primary criteria that the query optimizer uses when choosing an access plan. The query optimizer can choose access plans based on cost or based on the user requirement that as much query processing as possible be performed by the remote data sources. With DB2_MAXIMAL_PUSHDOWN set to 'Y', reducing network traffic becomes the overriding criteria for the query optimizer. The query optimizer uses the access plan that performs the fewest number of "sends" to the data sources. Setting this server option to 'Y' forces the federated server to use an access plan that might not be the lowest cost plan. Using an access plan other than the lowest cost plan can decrease performance. If a materialized query table (MQT) on the federated server can process part or all of the query, then an access plan that includes the materialized query table might be used. Using a materialized query table instead of pushing down operations to the data sources reduces network traffic. When the DB2_MAXIMAL_PUSHDOWN server option is set to 'Y' a query that will result in a Cartesian product will not be pushed down the remote data sources. Queries that will result in a Cartesian product will be processed by the federated database. The DB2_MAXIMAL_PUSHDOWN server option does not need to be set to 'Y' for the federated server to pushdown query processing to the remote data sources. When this server option is set to 'N' (the default), the query optimizer will pushdown query processing to the data sources. However, the primary criteria the optimizer uses when the option is set to 'N' is cost instead of network traffic.

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 Linux, UNIX, and Windows TIMESTAMP data type. You can change the local data type to the DB2 for Linux, 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, can return a different result. An example is the '>' (greater than) operator. For those data sources with different collating sequences, the greater than operator can return different results than if it is evaluated locally by DB2 for Linux, UNIX, and Windows.

Related concepts:

- "Collating Sequences" in the *Application Development Guide: Programming Client Applications*
- "Data type mappings in a federated system" on page 45
- "Tuning query processing" on page 131
- "Pushdown analysis" on page 133
- "Nickname characteristics affecting pushdown opportunities" on page 138
- "Query characteristics affecting pushdown opportunities" on page 140

Related reference:

- Chapter 21, "Server options for federated systems," on page 219

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® UDB 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 UDB 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 can 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 on the data source, but the net result should be equivalent.

Materialized query tables

A materialized query table is a summary table that is created from the result set of a query. The query used to create a materialized query table uses a fullselect that contains a GROUP BY clause to summarize data from the tables referenced in the query.

Unlike a view, a materialized query table stores the actual data from the result set in a table. In a federated system, you can create a materialized query table that references one or more nicknames, or a combination of nicknames and local tables.

A materialized query table defined by a query that references a nickname is a local table on the federated server. The materialized query table contains a copy of the data stored at the remote data sources that was returned in the result set of the query. Users of the federated system need not be aware of the existence of the materialized query table. When a query is sent to the federated server that references the nicknames on which the materialized query table is based, the query

optimizer can transparently use the local materialized query table instead of accessing the remote data source. Accessing local data instead of remote data can improve performance.

Materialized query tables can be created with the REFRESH DEFERRED option only.

Related concepts:

- "Cache tables" on page 175
- "Tuning query processing" on page 131
- "Pushdown analysis" on page 133
- "Server characteristics affecting pushdown opportunities" on page 134
- "Query characteristics affecting pushdown opportunities" on page 140

Related tasks:

- "Altering a nickname" on page 32

Related reference:

- Chapter 24, "Nickname column options for federated systems," on page 247

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 134
- "Nickname characteristics affecting pushdown opportunities" on page 138

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 two 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.
- You can use the **db2expln** and **dynexpln** tools to understand the access plan chosen for a particular SQL statement. You can also use the integrated Explain Facility in the DB2® Control Center in conjunction with Visual Explain to understand the access plan chosen for a particular SQL statement. Both dynamic and static SQL statements can be explained using the Explain Facility. One difference from the Explain tools is that with Visual Explain the Explain information is presented in a graphical format. Otherwise the level of detail provided in the two methods is equivalent. To fully use the output of **db2expln** , and **dynexpln** you must understand:
  - The different SQL statements supported and the terminology related to those statements (such as predicates in a SELECT statement)
  - The purpose of a package (access plan)
  - The purpose and contents of the system catalog tables
  - General application tuning concepts

Consider using the Explain tools with the DB2_MAXIMAL_PUSHDOWN server option. Run the Explain tools on a query with the DB2_MAXIMAL_PUSHDOWN server option set to 'N'. This is the default setting for this option. Pushdown analysis determines which parts of the SQL can be pushed down. Then the query optimizer generates all the alternative plans that do not violate the criteria set by pushdown analysis. The query optimizer estimates the cost of each plan, and will select the plan with the lowest estimated cost. Then set the DB2_MAXIMAL_PUSHDOWN server option to 'Y'. Use the Explain tools on the same SQL statement. The plan displayed in the Explain output shows all of the SQL operations that can be pushed down to the data source. If you see a difference between the two plans, the difference is the result of the information used by the query optimizer for cost optimization. This information includes the nickname index information, the nickname statistics, and the server attributes.

Related concepts:

- "Explain tools" in the *Administration Guide: Performance*
- "SQL explain tools" 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 131
- "Pushdown analysis" on page 133
- "Understanding access plan evaluation decisions" on page 142
- "Data source upgrades and customization" on page 144

Related tasks:

- "Global optimization" on page 144

Related reference:

- "db2exfmt - Explain Table Format Command" in the *Command Reference*
- "db2expln - SQL Explain Command" in the *Command Reference*

Understanding access plan evaluation 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 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 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® UDB 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 133
- "Analyzing where a query is evaluated" on page 140

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 ALTER SERVER, to update the catalog.

Related concepts:
- "Tuning query processing" on page 131
- "Pushdown analysis" on page 133

Related tasks:
- "Global optimization" on page 144

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.

Relational and nonrelational wrappers differ in the details of how an access plan is produced, but the concept and final effect are the same.

Related concepts:

- "Tuning query processing" on page 131
- "Pushdown analysis" on page 133
- "Server characteristics affecting global optimization" on page 145

Related tasks:

- "Nickname characteristics affecting global optimization" on page 147

Global optimization-details

While both server characteristics and nickname characteristics affect global optimization, nickname characteristics have a greater impact on the cost estimates generated by the query optimizer.

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 location options (such as the data source computer 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

Use caution when tuning the CPU_RATIO, IO_RATIO, or COMM_RATE server options as you can get unexpected errors if the cost calculation for a query causes overflows or underflows.

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. A low ratio is a value that is less than 1.

A CPU_RATIO server option setting of 1 indicates that the DB2 federated CPU and the data source CPU have the same CPU speed, a 1:1 ratio. If the DB2 federated CPU speed is 50% slower than the data source CPU speed, then .5 should be value for the CPU_RATIO server option setting. If the DB2 federated CPU speed is twice as fast as the data source CPU speed, then 2 should be value for the CPU_RATIO server option setting.

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. A low ratio is a value that is less than 1.

An IO_RATIO server option setting of 1 indicates that the DB2 federated I/O speed and the data source I/O speed have the same I/O speed, a 1:1 ratio. If the DB2 federated I/O speed is 50% slower than the data source I/O speed, then .5 should be value for the IO_RATIO server option setting. If the DB2 federated I/O speed is twice as fast as the data source I/O speed, then 2 should be value for the IO_RATIO server option setting.

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 UDB 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 UDB 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 generate remote plan hints. 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.

You should run some tests to determine if this server option will improve the performance of your queries.

You cannot code your own plan hints in a query.

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, t1index)*/`

Related concepts:

- "Tuning query processing" on page 131
- "Server characteristics affecting pushdown opportunities" on page 134

Related tasks:

- "Altering server definitions and server options" on page 26
- "Global optimization" on page 144
- "Nickname characteristics affecting global optimization" on page 147

Related reference:

- Chapter 21, "Server options for federated systems," on page 219

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 data source table is only acquired when the nickname is created for that table. After the nickname is created, the index information for that data source table is not updated on the federated server. When the remote index information changes, you can update the index information stored on the federated server by dropping the nickname for the table and creating the nickname again. Alternatively, if a new index is added for the data source table, you can define an index specification for the table on the federated server.

Index information is not gathered for nicknames on objects that do not have indexes such as views, synonyms, or nonrelational 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 table acquires a new index.
- You create a nickname for a data source object that does not contain indexes such as a view or a synonym.

Consider your needs before issuing CREATE INDEX...SPECIFICATION ONLY statements on a nickname for a data source view:

- If the remote view is a simple SELECT statement on a data source table with an index, creating an index specification on the nickname that matches the index on the data source table can significantly improve query performance.
- If an index specification is created for a remote view that is not simple SELECT statements (for example, a view created by joining two tables), query performance might suffer.

Suppose that an index specification is created for a remote 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 might 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 retrieved from the data source 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 describe the overall size of tables and views, and the range of values in associated columns. The information retrieved includes:

- The number of rows in a nickname object
- The number of pages that a nickname occupies
- The number of distinct values in each column of a table
- The number of distinct values in columns of an index
- The Highest/lowest values of a column

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, the 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 characteristics for a remote object on which a nickname is defined change, you have three choices for updating the statistics:

- Use the nickname statistics update facility in the DB2 Control Center.
- Run the equivalent of RUNSTATS at the data source. Then drop the current nickname and create the nickname again. This is the recommended method for updating statistics.

- Manually update the statistics in the SYSSTAT.TABLES catalog view. Use this method only when you know that the statistical information on the remote data source is incorrect or incomplete.

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 that the optimizer has accurate statistics when it develops access plans for processing queries on 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 statistics update facility - overview" on page 187
- "Nickname characteristics affecting pushdown opportunities" on page 138
- "Server characteristics affecting global optimization" on page 145

Related tasks:

- "Dropping a nickname" on page 42
- "Creating index specifications for data source objects" on page 72

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 two 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.
- You can use the **db2expln** and **dynexpln** tools to understand the access plan chosen for a particular SQL statement. You can also use the integrated Explain Facility in the DB2® Control Center in conjunction with Visual Explain to understand the access plan chosen for a particular SQL statement. Both dynamic and static SQL statements can be explained using the Explain Facility. One difference from the Explain tools is that with Visual Explain the Explain information is presented in a graphical format. Otherwise the level of detail provided in the two methods is equivalent. To fully use the output of **db2expln** , and **dynexpln** you must understand:
  - The different SQL statements supported and the terminology related to those statements (such as predicates in a SELECT statement)
  - The purpose of a package (access plan)
  - The purpose and contents of the system catalog tables
  - General application tuning concepts

Related concepts:

- "Explain tools" in the *Administration Guide: Performance*
- "SQL explain tools" 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 131
- "Understanding access plan optimization decisions" on page 150

Related tasks:

- "Global optimization" on page 144

Related reference:

- "db2exfmt - Explain Table Format Command" in the *Command Reference*
- "db2expln - SQL Explain Command" in the *Command Reference*

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 might 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 will 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 UDB 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 will 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 more functions than the original query? Some of the extra functions might be generated to convert data types. Ensure that they are necessary.

Related concepts:

- "Pushdown analysis" on page 133
- "Understanding access plan evaluation decisions" on page 142
- "Analyzing global optimization" on page 149

Related tasks:

- "Global optimization" on page 144

System monitor elements affecting performance

The DB2 database system monitor gathers statistical information regarding the current state of the database manager, and activity information such as counters and other measurements of database processing.

In a federated system, you can use the DB2 database system monitor to gather information about database activity, system performance, and application performance.

The Timestamp monitor switch is used to track the response times of interactions that the federated database has with a data source. The federated data elements tracked by the Timestamp switch are:

- Create nickname response time
- Delete response time
- Insert response time
- Pass-through time
- Query response time
- Remote lock time
- Update response time

The default setting for the Timestamp monitor switch is ON.

Recommendation: You can increase performance by changing the setting for the Timestamp monitor switch to OFF for all applications. If one application has the Timestamp switch set to ON, the system will continue to collect the response times. Therefore, you will not increase performance by turning off the timestamp switch for only some of your applications.

Turning off the switch does have other implications.

- Turning off the Timestamp monitor switch for all applications requires that you stop and restart the DB2 instance to implement the change.
- Turning off the Timestamp monitor switch disables the gathering of timestamp information for both federated and non-federated applications. The local database will not receive timestamp information either.

If you need timestamp information for local non-federated applications, then you should not turn off the Timestamp monitor switch.

You can set the Timestamp switch to OFF for all applications by using this command:

```
update dbm cfg using dft_mon_timestamp off
```

Then issue:

```
db2stop
db2start
```

Stopping and starting DB2 UDB will ensure that the switch is off for all applications.

Specific information about each of the elements tracked by the Timestamp switch is discussed in a separate topic.

Related reference:

- Chapter 31, "Federated database systems monitor elements," on page 297

Chapter 12. Parallelism with queries that reference nicknames

This chapter describes how parallelism works with queries that reference nicknames.

This chapter contains:

Parallelism with queries that reference nicknames

Queries that contain nicknames can participate in three types of intraquery parallelism:

- Intrapartition query parallelism on single partition, multiprocessor configurations
- Interpartition query parallelism on multiple partition configurations
- Mixed query parallelism that consists of both intrapartition and interpartition parallelism where each partition runs on an SMP computer

Related concepts:

- "Parallelism" in the *Administration Guide: Planning*
- "Partition and processor environments" in the *Administration Guide: Planning*
- "The DB2 Process Model" in the *Administration Guide: Performance*
- "Computational partition groups" on page 160
- "Parallel access plans of queries that reference nicknames" on page 163
- "Intrapartition parallelism with queries that reference nicknames" on page 156
- "Interpartition parallelism with queries that reference nicknames" on page 157
- "Mixed parallelism with queries that reference nicknames" on page 162

Related reference:

- Chapter 20, "Wrapper options for federated systems," on page 217

Intrapartition parallelism with queries that reference nicknames

Intrapartition parallelism refers to the process of dividing a query into multiple concurrent parts that run in parallel by multiple processes on a single database partition. In federated queries, the part of a query that involves local data can run in parallel while the part that involves nicknames runs serially.

Queries that reference local tables and nicknames can run faster than in previous releases of DB2® Information Integrator because multiple processors can work on the local portions of the query.

The DFT_DEGREE database configuration parameter and the CURRENT DEGREE special register control the degree of intrapartition parallelism.

Related concepts:

- "Parallelism" in the *Administration Guide: Planning*
- "Partition and processor environments" in the *Administration Guide: Planning*

Related tasks:

- "Enabling intra-partition parallelism for queries" in the *Administration Guide: Implementation*
- "Enabling intrapartition parallelism with queries that reference nicknames" on page 156

Related reference:

- "Intrapartition parallelism with queries that reference nicknames - examples of access plans" on page 163

Enabling intrapartition parallelism with queries that reference nicknames

For queries that reference local tables and nicknames in a multiprocessor environment, you can enable intrapartitition parallelism. The federated server can then process the local tables in parallel.

Restrictions:

The federated system can process only the local portion of a query in parallel. The coordinator partition processes all operations on the remote portion of a query in serial.

Procedure:

1. Set the INTRA_PARALLEL database configuration parameter to YES.
2. Set the MAX_QUERYDEGREE database configuration parameter to a value greater than 1.
3. Set the DFT_DEGREE database configuration parameter to a value greater than 1, or set the special register CURRENT DEGREE.

   If you set the DFT_DEGREE parameter to ANY, the default level of intrapartition parallelism equals the number of processors on the computer.

Related concepts:

- "Parallelism with queries that reference nicknames" on page 155

Related reference:

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

Interpartition parallelism with queries that reference nicknames

Interpartition parallelism refers to the process of dividing a single query into multiple parts that run in parallel on different partitions of a partitioned database.

In queries that reference local and remote data, the federated server can distribute the remote data to each of the local partitions. Figure 7 on page 158 shows the concept of interpartition parallelism that involves local and remote data sources. The top portion of the figure shows how this type of query was processed in previous releases of DB2® Information Integrator. The remote nickname data and the local partitioned data were processed serially at a single coordinator partition. The bottom portion of the figure shows how DB2 Information Integrator processes these types of queries in this release. The database distributes the nickname data to the partitions of the local system for parallel processing.

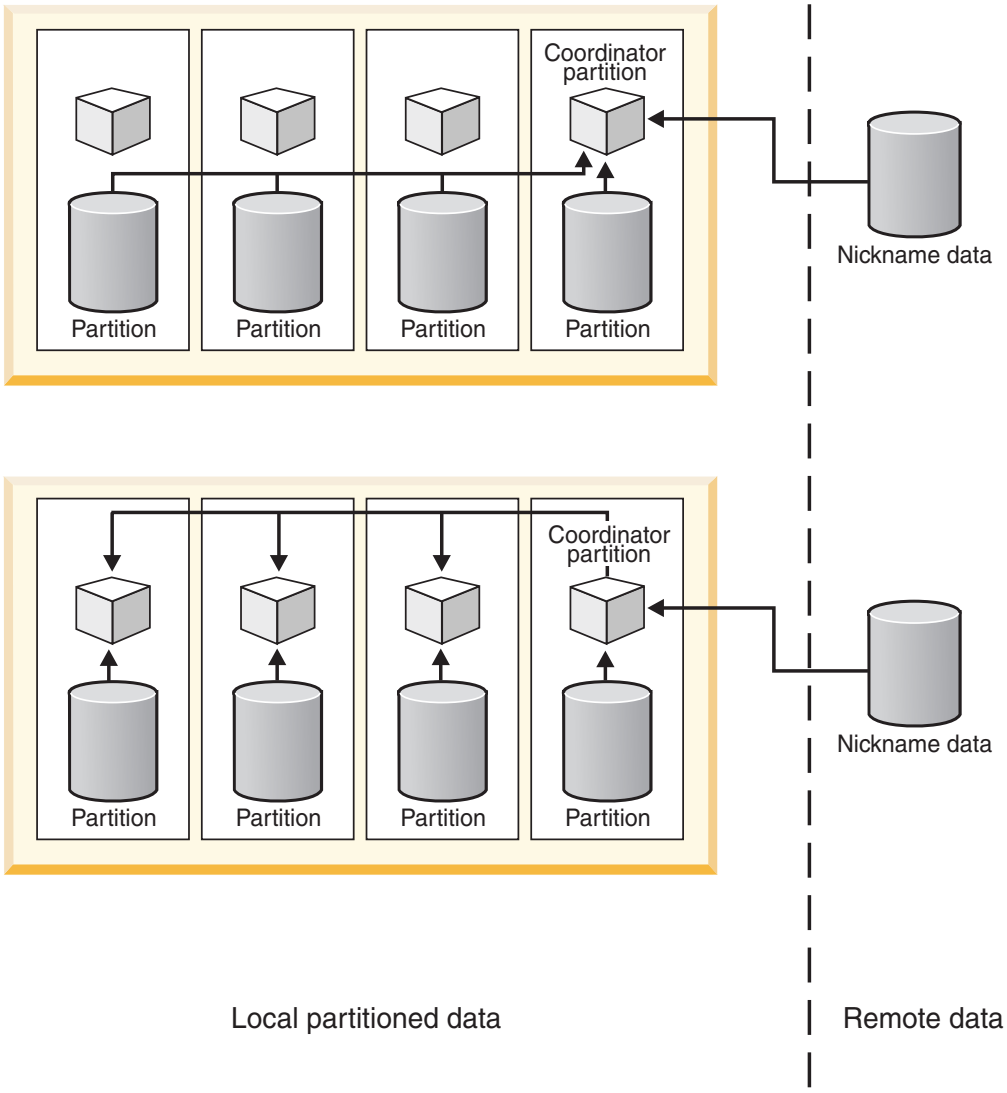


Figure 7. Interpartition parallelism of local and remote data sources

Figure 8 on page 159 shows the concept of interpartition parallelism that involves only remote data sources. The top portion of the figure shows serial processing of the remote nickname data at a single coordinator partition. The bottom portion of the figure shows the coordinator partition distributing the data across a computational partition group.

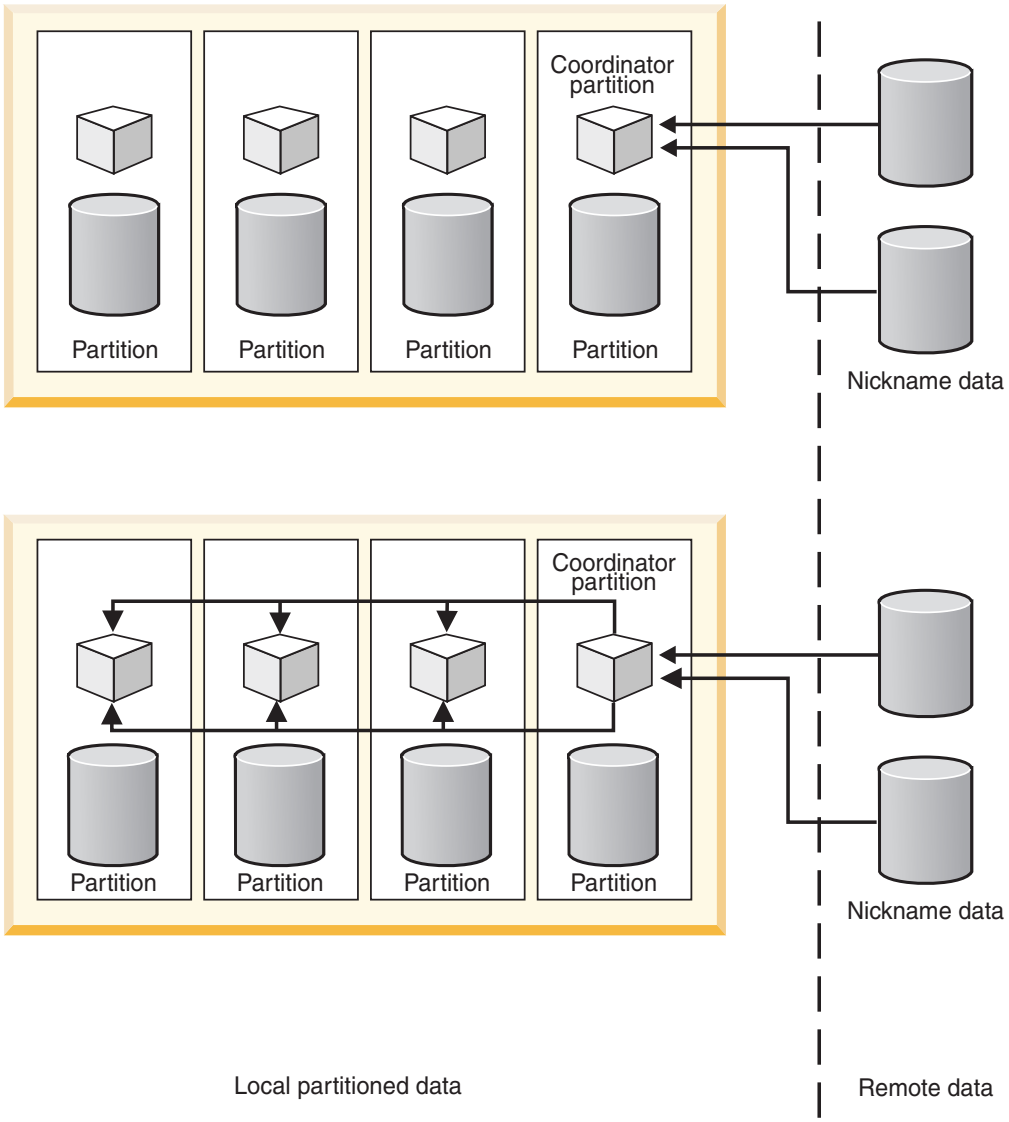


Figure 8. Interpartition parallelism of a query that references only remote data sources

Related concepts:

- "Parallelism with queries that reference nicknames" on page 155
- "Computational partition groups" on page 160

Related tasks:

- "Enabling interpartition parallelism with queries that reference nicknames" on page 159

Enabling interpartition parallelism with queries that reference nicknames

Queries that reference nicknames that can run across multiple partitions in parallel can run faster.

In a partitioned database environment, federated queries can run in parallel under the following conditions:

- They involve a combination of nicknames and local partitioned tables.
- They involve only nicknames that use a computational partition group.

You do not need to set any database parameters or database configuration parameters in a partitioned environment to achieve interpartition parallelism.

Restrictions:

Only parts of a query that reference nicknames that use fenced wrappers can run in parallel. Any parts of a query that reference nicknames that use trusted wrappers cannot run in parallel.

Procedure:

1. Set up a computational partition group.
2. Issue the CREATE WRAPPER or ALTER WRAPPER statement with the DB2_FENCED option set to Y.

Related concepts:

- "Computational partition groups" on page 160
- "Interpartition parallelism with queries that reference nicknames" on page 157

Computational partition groups

A computational partition group defines a set of partitions for the optimizer to use for performing a dynamic redistribution operation for join operations. A computational partition group is a database partition group, other than IBMCATNODEGROUP, that is specified in the system catalog, SYSCAT.DBPARTITIONGROUPS.

You can use computational partition groups for federated interpartition query parallelism. By using a computational partition group, you can run the nickname parts of queries in parallel. In a query plan that involves a computational partition group, the federated server redistributes nickname data across the partitions to create a parallel join. This type of plan can make queries run faster if the amount of nickname data that participates in the join is large.

You use the DB2_COMPPARTITIONGROUP registry variable to specify the computational partition group.

Related concepts:

- "Parallelism with queries that reference nicknames" on page 155

Related tasks:

- "Defining a computational partition group" on page 160

Defining a computational partition group

Defining a computational partition group enables the optimizer to use a plan that distributes nickname data to the partitions of the computational partition group. You define a computational partition group to enable interpartition query parallelism for queries or parts of queries that reference only nicknames.

Prerequisites:

All partition groups used to represent the computational partition group on all the databases in the instance must have the same name. You can define these partition groups differently in each database, but they must have the same name.

Restrictions:

The optimizer uses computational partition groups for only the parts of a query that reference nicknames without referencing local data.

Procedure:

To define a computational partition group, issue the following command at the DB2 command line:

```
db2set DB2_COMPPARTITIONGROUP=partitiongroup_name
```

partitiongroup_name is the name of the partition group that you want to define as the computational partition group.

The following example shows how to define the computational partition group, FINANCE3, using the DB2_COMPPARTITIONGROUP registry variable.

```
db2set DB2_COMPPARTITIONGROUP=FINANCE3
```

Related concepts:

- "Computational partition groups" on page 160

Related reference:

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

Interpartition parallelism with queries that reference nicknames - performance expectations

For queries that reference a combination of local partitioned tables and nicknames, the optimizer can choose an execution plan that redistributes nickname data across appropriate partitions. Redistribution plans can make queries run faster if the amount of nickname data in the join is smaller than the amount of local partitioned data. If the amount of nickname data in the join is considerably larger than the local data, then a parallel plan with redistribution of the nickname data is unlikely to be used. If the optimizer does not choose a parallel plan, the federated server performs the joins serially between nicknames and local tables at the coordinator partition.

For joins between two nicknames, an execution plan that distributes the data among all partitions of a computational partition group can be beneficial if it involves a large amount of data. The advantage of processing the large join in parallel offsets the additional cost of redistributing the data across multiple partitions. If the amount of nickname data is relatively small the join is not expensive enough to merit the extra cost of redistributing the data across partitions. In general, the optimizer chooses computational partition group plans if the nicknames involved are large; otherwise, the federated server joins the nicknames serially at the coordinator partition.

Related concepts:

- "Computational partition groups" on page 160
- "Interpartition parallelism with queries that reference nicknames" on page 157

Related reference:

- "Interpartition parallelism with queries that reference nicknames - examples of access plans" on page 165

Mixed parallelism with queries that reference nicknames

For queries that contain local tables and nicknames in a partitioned environment, you can use both intrapartition and interpartition parallelism. The federated server can distribute remote data among partitions and process data in parallel within each partition.

Related concepts:

- "Parallelism with queries that reference nicknames" on page 155

Related tasks:

- "Enabling mixed parallelism with queries that reference nicknames" on page 162

Enabling mixed parallelism with queries that reference nicknames

Federated queries that run in parallel can run faster. You can improve the performance of queries that reference local and remote data by the use of intrapartition and interpartition parallelism. The federated queries reference a combination of nicknames and local partitioned tables.

Procedure:

1. Set the MAX_QUERYDEGREE database configuration parameter to a value greater than 1.
2. Set the DFT_DEGREE database configuration parameter to a value greater than 1, or you must set the special register CURRENT DEGREE. If you set the DFT_DEGREE parameter to ANY, the default level of intrapartition parallelism equals the number of SMP processors on the computer.
3. Set up a computational partition group.
4. Issue the CREATE WRAPPER or ALTER WRAPPER statement with the DB2_FENCED option set to Y.

Related concepts:

- "Computational partition groups" on page 160
- "Mixed parallelism with queries that reference nicknames" on page 162

Related tasks:

- "Trusted and fenced mode process environments" in the *IBM DB2 Information Integrator Wrapper Developer's Guide*

Related reference:

- "Mixed parallelism with queries that reference nicknames - examples of access plans" on page 167

Parallel access plans of queries that reference nicknames

The SQL Explain facility captures information about the access plans that the optimizer uses when it processes queries.

Related concepts:

- "SQL explain facility" in the *Administration Guide: Performance*
- "Explain tools" in the *Administration Guide: Performance*
- "SQL explain tools" in the *Administration Guide: Performance*

Related reference:

- "Intrapartition parallelism with queries that reference nicknames - examples of access plans" on page 163
- "Interpartition parallelism with queries that reference nicknames - examples of access plans" on page 165
- "Mixed parallelism with queries that reference nicknames - examples of access plans" on page 167

Intrapartition parallelism with queries that reference nicknames - examples of access plans

The DB2 UDB Explain facility can generate the access plan the optimizer uses during query processing. The following examples show how the optimizer accesses nickname data in an intrapartition parallelism environment.

Example 1: Before DB2 Information Integrator Version 8.2 parallelism support

In this example, the federated server processes the join of the local table, ORDERS, and nickname, ITEMS, serially. No intrapartition parallelism is used.

```
SELECT *
FROM ORDERS A, ITEMS B
WHERE A.ID1 = B.ID1 AND B.ITEM = 3
```

```
            RETURN
           (   1)
              |
           HSJOIN
           (   2)
         /----+---\
     TBSCAN         SHIP
     (   3)        (   4)
       |             |
   TABLE: NEWTON  NICKNM: NEWTON
     ORDERS         ITEMS
```

Example 2: With DB2 Information Integrator Version 8.2 parallelism support

In this example of a join, the query can run faster by having the local table read in parallel before the serial join with the nickname.

```
SELECT *
FROM ORDERS A, ITEMS B
WHERE A.ID1 = B.ID1 AND B.ITEM = 3
```

```
            RETURN
```

```
                 (   1)
                   |
                 HSJOIN
                 (   2)
                /----+---\
          LTQ             SHIP
         (   3)          (   5)
           |               |
         TBSCAN     NICKNM: NEWTON
         (   4)          ITEMS
           |
      TABLE: NEWTON
       ORDERS
```

Example 3: Intrapartition parallelism with aggregation

In this example, the database aggregates the local table data in parallel in the partition, improving the execution of the aggregation. The join of the local table and the nickname occurs serially at the coordinator node.

```
SELECT *
FROM ITEMS A
WHERE ID =
   (SELECT MAX(ID)
   FROM ORDERS
   WHERE NUMBER = 10)
```

```
                 RETURN
                 (   1)
                   |
                 NLJOIN
                 (   2)
                /----+---\
          GRPBY           SHIP
         (   3)          (   7)
           |               |
          LTQ       NICKNM: NEWTON
         (   4)          ITEMS
           |
           1
         GRPBY
         (   5)
           |
          TBSCAN
         (   6)
           |
      TABLE: NEWTON
        ORDERS
```

Related concepts:

- "Example two: single-partition plan with intra-partition parallelism" in the *Administration Guide: Performance*
- "Parallel access plans of queries that reference nicknames" on page 163

Related tasks:

- "Enabling intrapartition parallelism with queries that reference nicknames" on page 156

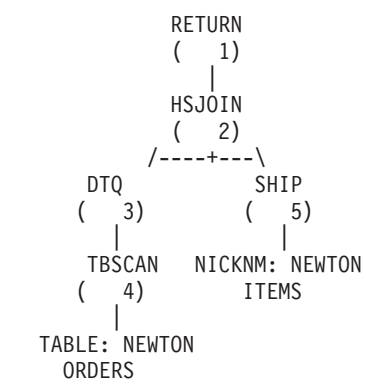
Interpartition parallelism with queries that reference nicknames - examples of access plans

The DB2 UDB Explain facility can generate the access plan the optimizer uses during query processing. The following examples shows how the optimizer accesses nickname data in an interpartition parallelism environment.

Example 1: Trusted mode

In this example, the nickname uses a trusted wrapper. The database serially performs the join between the local table and the nickname at the coordinator partition. The database brings the local data, which is distributed over two partitions, to the coordinator partition. The federated server then joins the local data with the nickname data. The database serially joins nicknames that are defined by using a trusted wrapper at the coordinator partition. The database cannot distribute the data across multiple partitions to create a parallel join.

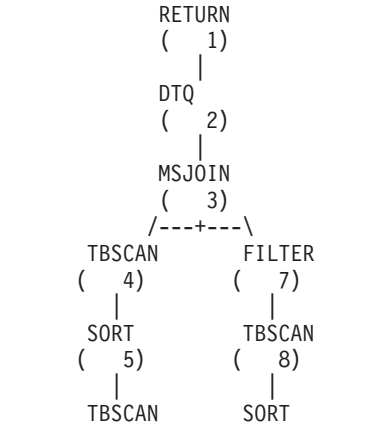
```
SELECT *
FROM ORDERS A, ITEMS B
WHERE A.ID1 = B.ID1 AND B.ITEM = 3
```



Example 2: Fenced mode

In this example, the nickname uses a fenced wrapper. The federated server distributes the nickname data to the other partitions and performs the join with the local data in parallel.

```
SELECT *
FROM ORDERS A, ITEMS B
WHERE A.ID1 = B.ID1 AND B.ITEM = 3
```



```
    (   6)       (   9)
      |            |
TABLE: NEWTON     DTQ
ORDERS           (  10)
                   |
                  SHIP
                 (  11)
                   |
              NICKNM: NEWTON
                  ITEMS
```

Example 3: Fenced mode without a computational partition group

In this example, the two nicknames use a fenced wrapper, and a computational partition group is not defined. The federated server performs the join at the coordinator partition. The federated server does not distribute the data to the other partitions for processing.

```
SELECT *
FROM ITEMS A, LOCATIONS B
WHERE A.ID1 = B.ID1
```

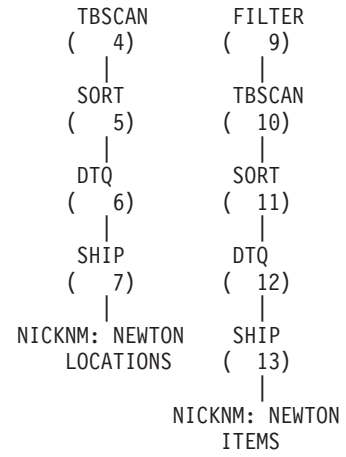
```
            RETURN
            (   1)
              |
            MSJOIN
            (   2)
           /---+---\
       TBSCAN       FILTER
      (   3)       (   7)
        |            |
       SORT         TBSCAN
      (   4)       (   8)
        |            |
       SHIP         SORT
      (   5)       (   9)
        |            |
  NICKNM: NEWTON    SHIP
    LOCATIONS      (  10)
                     |
                NICKNM: NEWTON
                    ITEMS
```

Example 4: Fenced mode with a computational partition group

In this example, the nicknames use fenced wrappers, and a computational partition group is defined. In this case, the optimizer selects a plan that distributes the data from the coordinator partition to the other partitions in the computational partition group.

```
SELECT *
FROM ITEMS A, LOCATIONS B
WHERE A.ID = B.ID
```

```
            RETURN
            (   1)
              |
            DTQ
            (   2)
              |
            MSJOIN
            (   3)
           /---+---\
```



Related concepts:

- "Example three: multipartition plan with inter-partition parallelism" in the *Administration Guide: Performance*
- "Parallel access plans of queries that reference nicknames" on page 163

Related tasks:

- "Trusted and fenced mode process environments" in the *IBM DB2 Information Integrator Wrapper Developer's Guide*
- "Enabling interpartition parallelism with queries that reference nicknames" on page 159

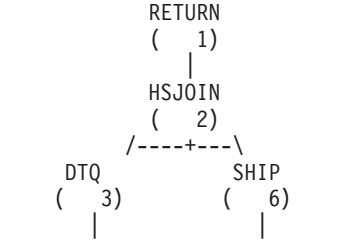
Mixed parallelism with queries that reference nicknames - examples of access plans

The DB2 UDB Explain facility can create the access plan that the optimizer uses during query processing. The following examples shows how the optimizer accesses nickname data in an environment that uses both intrapartition parallelism and interpartition parallelism.

Example 1: Trusted mode

The following example shows a join between a local table and a nickname in trusted mode. The federated server processes the local data in parallel in each partition before it joins the local data with the nickname data at the coordinator partition. The federated server does not process the nickname data in parallel across the partitions or the processors on any given partition.

```
SELECT *
FROM ORDERS A, ITEMS B
WHERE A.ID1 = B.ID1 AND B.ITEM = 3
```

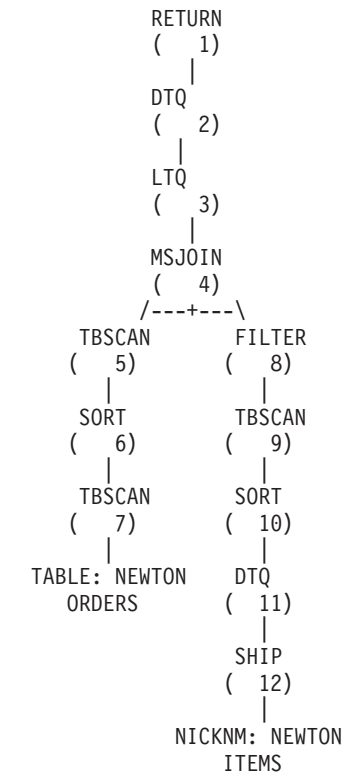


```
      LTQ      NICKNM: NEWTON
     (   4)        ITEMS
       |
      TBSCAN
     (   5)
       |
  TABLE: NEWTON
    ORDERS
```

Example 2: Fenced mode

The following example shows a join between a local table and a nickname in fenced mode. The federated server distributes the nickname data from the coordinator partition to the other partitions in the system. The federated server processes both the local table data and the nickname data in parallel among the partitions and processors.

```
SELECT *
FROM ORDERS A, ITEMS B
WHERE A.ID1 = B.ID1 AND B.ITEM = 3
```



Related concepts:

- "Example four: multipartition plan with inter-partition and intra-partition parallelism" in the *Administration Guide: Performance*
- "Parallel access plans of queries that reference nicknames" on page 163

Related tasks:

- "Trusted and fenced mode process environments" in the *IBM DB2 Information Integrator Wrapper Developer's Guide*
- "Enabling mixed parallelism with queries that reference nicknames" on page 162

Chapter 13. Materialized query tables and federated systems

This chapter describes how you can use materialized query tables that reference nicknames in your federated system.

This chapter contains:

Materialized query tables and federated systems – overview

A materialized query table is a table that caches the results of a query. When you submit the query again, the database engine can return the data from the materialized query table. You can use materialized query tables with nicknames to improve the performance of a query and to encapsulate a part of logic. You also use materialized query tables when you create cache tables.

The SQL optimizer determines if a query will run more efficiently with a materialized query table than the base tables or nicknames. The optimizer uses the following factors to select a materialized query table:

- The materialized query table must match part or all of the query.
- The refresh age criterion must be met.
- The access plan that uses a materialized query table must be cheaper than the access plan that uses the base tables or nicknames.

The following data sources support materialized query tables:

- Relational data sources
  - DRDA®
  - Informix®
  - ODBC
  - Oracle
  - Sybase
  - MS SQL Server
  - Teradata
- Nonrelational data sources
  - BioRS
  - BLAST
  - Documentum
  - Entrez
  - Excel
  - HMMER
  - IBM® Lotus® Extended Search
  - Table-structured files
  - Web Services
  - WebSphere® Business Integration
  - XML

Related concepts:

- "Cache tables" on page 175
- "Performance and tuning planning— materialized query tables in a federated system" in the *IBM DB2 Information Integrator Application Developer's Guide*

Related tasks:

- "Creating a materialized query table" in the *Administration Guide: Implementation*
- "Altering materialized query table properties" in the *Administration Guide: Implementation*
- "Creating a federated materialized query table" on page 170

Related reference:

- "Data source specific restrictions for materialized query tables" on page 170

Creating a federated materialized query table

You use materialized query tables to cache data locally and to improve the performance of your queries. You can use nicknames from relational and nonrelational data sources to create materialized query tables.

Restrictions:

- Data source specific restrictions for materialized query tables.
- If a query has a function template in a predicate or a select list, the function template must be part of the materialized query table.

Procedure:

To create a materialized query table, issue a CREATE TABLE statement that includes the nickname that represents your remote data source object.

Related concepts:

- "Materialized query tables and federated systems – overview" on page 169

Related tasks:

- "Creating a materialized query table" in the *Administration Guide: Implementation*
- "Adding a materialized query table to a cache table" on page 178

Related reference:

- "Data source specific restrictions for materialized query tables" on page 170
- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

Data source specific restrictions for materialized query tables

This topic describes the restrictions on creating materialized query tables for the following data sources:

- Bio-RS
- BLAST
- Entrez
- HMMER
- IBM Lotus Extended Search
- Table-structured files

- Web services
- XML

Bio-RS, Entrez, and IBM Lotus Extended Search restrictions

These wrappers require at least one predicate in the WHERE clause. You must create a materialized query table that satisfies the predicate requirements of the wrappers. If you do not specify a predicate, a refresh of the materialized query table fails.

BLAST and HMMER restrictions

If you use the BLAST and HMMER wrappers, the data source requires values for some predicates. The wrapper provides the default value. For other predicates, you must specify a value. If you do not specify a predicate, a refresh of the materialized query table fails.

In the following example, the materialized query table is successfully created because the wrapper did not access the data source.

```
CREATE TABLE MY_MQT AS (SELECT * FROM BLAST_NICK)
   DATA INITIALLY DEFERRED REFRESH DEFERRED ENABLE QUERY OPTIMIZATION;
```

However, when you refresh the table, the BLAST data source is contacted to retrieve data. The BLAST data source issues an error because it needs a predicate on the column blast_seq.

For optional predicates, if you issue a query with non-default values, you must provide values when you create the materialized query table. If you do not specify a predicate, a refresh of the materialized query table fails.

In the following example, when you create the materialized query table, the wrapper provides the default values for the optional parameters. These optional parameters are in_arg1 = 2 and in_arg2 < 30.

```
CREATE TABLE MY_MQT AS (SELECT * FROM BLAST_NICK WHERE BLAST_SEQ = '12345')
   DATA INITIALLY DEFERRED REFRESH DEFERRED ENABLE QUERY OPTIMIZATION;
```

When you issue the following query, the optimizer matches the query to the materialized query table. If the optimizer chooses the materialized query table plan, this query yields incorrect results. The query returns incorrect results because the materialized query table contains data with in_arg1 = 2, but the query requested data with in_arg1 = 3.

```
SELECT * FROM BLAST_NICK WHERE BLAST_SEQ = '12345' AND IN_ARG1 = 3;
```

When you create a materialized query table, you must explicitly specify the values for some of the predicates even if the wrapper supplies default values. Otherwise, the optimizer might not route the query to a matching materialized query table. This can happen because you did not explicitly specify these predicate values when you created the materialized query table.

The following example shows how the optimizer would fail to select the materialized query table when the query matches the materialized query table. If the materialized query table explicitly contains a fixed input column with a default value and the query does not have this predicate specified, the query will not be routed to the materialized query table.

```
CREATE TABLE K55ADMIN.BLAST_NICK1_M1 AS (
   SELECT SCORE, E_VALUE, QUERYSTRANDS
   FROM K55ADMIN.BLAST_NICK1
   WHERE BLASTSEQ='ATGATCGGATCGAATTCGAT'
   AND E_VALUE < 10) DATA INITIALLY DEFERRED REFRESH DEFERRED;
```

If you issue the following query, the query is not routed to the materialized query table. The query is not routed because the optimizer determines that the materialized query table did not specify E_VALUE < 10.

```
SELECT SCORE, E_VALUE, QUERYSTRANDS
FROM K55ADMIN.BLAST_NICK1 WHERE BLASTSEQ='ATGATCGGATCGAATTCGAT';
```

Table-structured file restrictions

If you define a nickname for a table-structured file with the DOCUMENT option, the materialized query table must have a predicate that specifies the file path. If you do not specify a predicate, a refresh of the materialized query table fails.

Web services restrictions

You can only create a materialized query table over a flattened view of a hierarchy of nicknames. You cannot create a materialized query table for each nickname of a hierarchy.

XML restrictions

You cannot create a materialized table on a child table.

If you define a nickname for an XML table with the DOCUMENT option, the materialized query table requires a predicate that specifies the file path. If you do not specify a predicate, a refresh of the materialized query table fails.

Related tasks:

- "Creating a federated materialized query table" on page 170

Related reference:

- "Restrictions on using materialized query tables with nicknames" on page 172

Restrictions on using materialized query tables with nicknames

DB2 Information Integrator does not support system-maintained materialized query tables that reference nicknames in a partitioned database environment.

To work around this restriction, you can use a user-maintained materialized query tables.

For example, for a nonrelational nickname named DEPART, you can issue the following commands to simulate a system-maintained materialized query table.

```
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION ALL;

CREATE TABLE AST1(C1, C2)
 AS (SELECT EMPNO, FIRSTNME FROM DEPART WHERE EMPNO>'000000')
 DATA INITIALLY DEFERRED REFRESH DEFERRED
 ENABLE QUERY OPTIMIZATION MAINTAINED BY USER;

SET INTEGRITY FOR AST1 ALL IMMEDIATE UNCHECKED;
```

```
INSERT INTO AST1 (SELECT EMPNO, FIRSTNME FROM DEPART WHERE EMPNO>'000000');

SET CURRENT REFRESH AGE ANY;
```

The following select statement can be answered by the previous defined materialized query table:

```
SELECT EMPNO, FIRSTNME FROM DEPART
 WHERE EMPNO > '000000' AND FIRSTNME LIKE 'AN%';
```

Chapter 14. Cache tables in a federated system

This chapter describes what a cache table is and how to use one in your federated system.

This chapter contains:

Cache tables

A cache table can improve query performance by accessing a local subset of data instead of accessing data directly from the remote relational data source. A cache table consists of these components:

- A nickname on your federated database system with the same column definitions and same data access as the remote relational data source table.
- One or more user-maintained materialized query tables that you define on the nickname. The nickname can contain a subset of high-use data from your remote data source.
- A user-defined replication schedule that is associated with each materialized query table to keep the local materialized query tables current with your remote data source.

Figure 9 on page 176 illustrates the cache table concept.

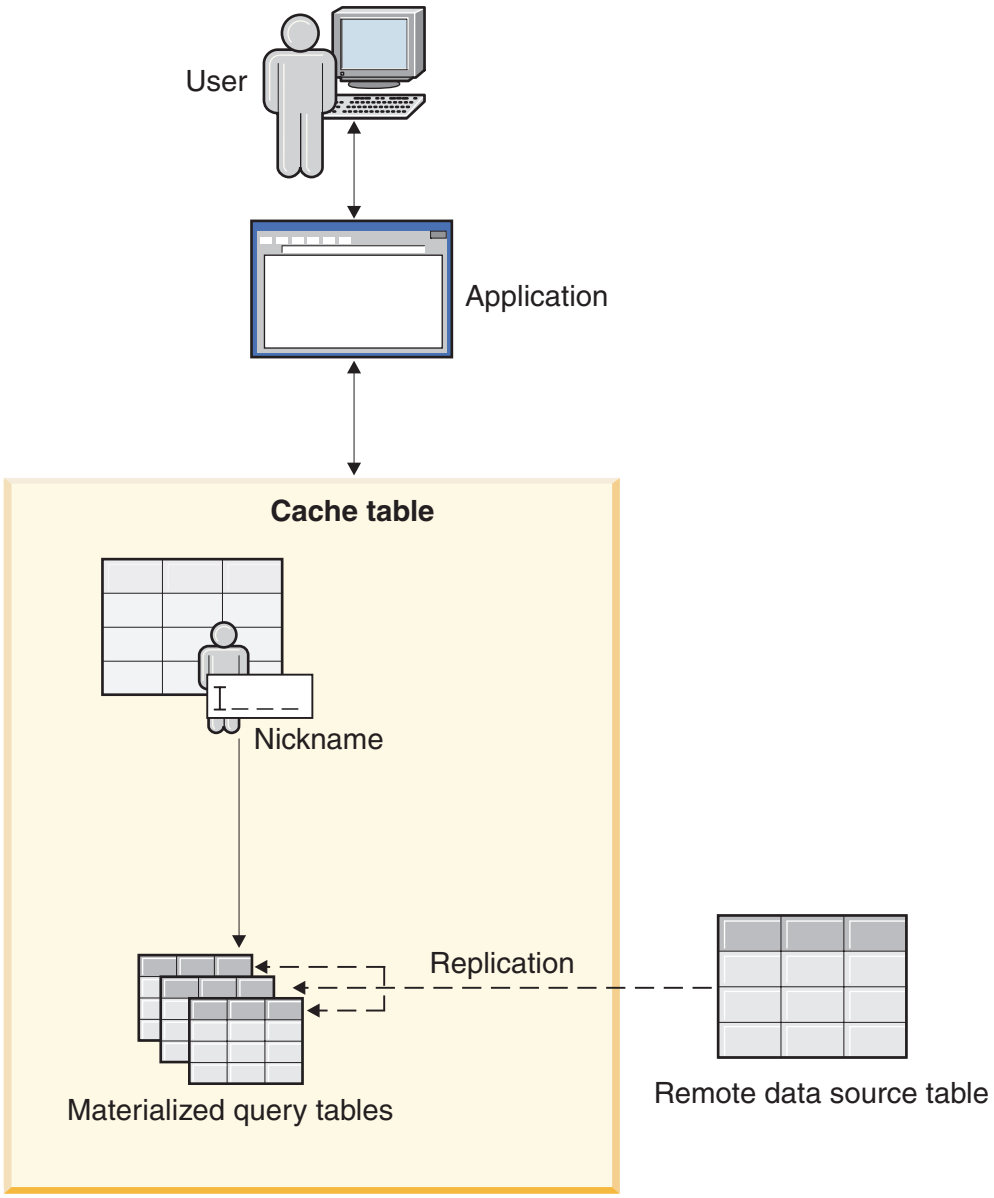


Figure 9. Cache tables consist of a nickname, materialized query tables, and a replication schedule.

The cache table has the same name as the nickname component. You can associate a cache table with only one remote table. The cache table can be a full replica or partial subset of rows from your remote data source. The cache table contains local data that is defined by the materialized query tables associated with it. The main difference between a cache table and a materialized query table is that a cache table is configured from the DB2® Control Center.

Applications that query a remote data source can query the cache table with minimal change to the application. During query processing, the optimizer directs the query to the cache table or the remote relational data source table.

Related concepts:

- "Replicated materialized query tables" in the *Administration Guide: Planning*
- "Materialized query tables and federated systems – overview" on page 169

Related tasks:

- "Creating a cache table" on page 177
- "Creating a federated materialized query table" on page 170

Creating a cache table

Use a cache table to store data that you access frequently but that does not change often. You can improve query performance by using cache tables.

You can cache data from data sources from which DB2 UDB supports heterogeneous replication. These data sources are DB2 family, Informix, Microsoft SQL Server, Oracle, and Sybase data sources.

Prerequisites:

- Set the FEDERATED database manager configuration parameter for the DB2 UDB server to YES.
- To access Informix data sources, install and configure the Informix Client SDK software in the federated server.
- To cache data from a DB2 UDB for Linux, UNIX, and Windows tables, you must configure the database that the table is stored in for *log retention logging*. Log retention logging is a type of archive logging. To configure the database for log retention logging, you set the LOGRETAIN value to RECOVER.
- The federated database or the source database must be on the computer from which you are creating the cache tables. If the federated database or the source database are not local, you must catalog the databases on the local computer. The alias name that you use when you catalog the database must be the same name as the database name.
- The user ID in the user mapping between the databases must have the authority to create tables in the source database.

Procedure:

To create a cache table:

1. In the DB2 Control Center, expand the **Cache Objects** folder.
2. Right-click the **Cache Tables** folder and click **Create**. The Cache Table wizard opens.
3. Complete the steps in the wizard.

The Cache Table wizard creates one materialized query table when you create the cache table. You can create additional materialized query tables to store other data from the same data source.

Related concepts:

- "Cache tables" on page 175

Related tasks:

- "Enabling a cache" on page 178
- "Adding a materialized query table to a cache table" on page 178

Related reference:

- "logretain - Log retain enable configuration parameter" in the *Administration Guide: Performance*

Enabling a cache

When you enable a cache, you activate the Capture and Apply programs. These programs replicate data from the data source to the materialized query table.

Generally, when you create a cache table, you enable the cache. However, you might need to manually enable a cache for replication in the following situations:

- When you create a cache table, if both of the following conditions are true. the database did not enable the cache for replication:
  - The data source is a DB2 UDB for Linux, UNIX, and Windows table.
  - You did not configure the database that the table resides in for log retention logging.
- After you create a cache table, a cache is disabled if the replication Capture or Apply program stops. When you restart the Capture or Apply program in the DB2 Replication Center, you must enable the cache to start replicating the data.

The DB2 Control Center sometimes refers to the term *log retention logging* as *archive logging*.

Prerequisites:

If the cache is for a DB2 UDB for Linux, UNIX, and Windows table, the following prerequisites apply:

- You must configure the database that the table is in for log retention logging.
- The Database Administration Server (DAS) must be running.

Procedure:

To enable a cache:

1. In the DB2 Control Center, expand the **Cache Objects** folder and the **Cache Tables** folder in the object tree.
2. Right-click the cache table that contains the cache that you want to enable and click **Enable Caching**. The Enable Caching window opens.
3. Select the cache that you want to enable and click **OK**.

Related concepts:

- "Cache tables" on page 175

Related tasks:

- "Adding a materialized query table to a cache table" on page 178

Adding a materialized query table to a cache table

When you create a cache table, the federated server stores data locally from the data source in a materialized query table. The criteria that you specify in the Cache Table wizard determines which data to store in the materialized query table.

You can create additional materialized query tables for the same cache table to store other data from the same data source object.

Prerequisites:

A cache table for the data source must exist.

Procedure:

To add a materialized query table to an existing cache table:

1. In the DB2 Control Center, expand the **Cache Objects** folder and the **Cache Tables** folder in the object tree.
2. Right-click the cache table that you want to add a materialized query table to and click **Properties**. The Cache Table Properties window opens.
3. Click **Add**. The Cache Table wizard opens.
4. Complete the wizard.

Related concepts:

- "Cache tables" on page 175
- "Materialized query tables and federated systems – overview" on page 169

Related tasks:

- "Enabling a cache" on page 178
- "Dropping a materialized query table from a cache table" on page 179

Dropping a materialized query table from a cache table

When you no longer want to store data locally in a materialized query table, you can drop the materialized query table from the cache table. If a cache table has only one materialized query table, dropping the materialized query table also drops the cache table.

Restrictions:

Use the DB2 Control Center to drop a materialized query table from a cache table to ensure complete removal of the materialized query table from your system.

Procedure:

To drop a materialized query table to an existing cache table:

1. In the DB2 Control Center, expand the **Cache Objects** folder and the **Cache Tables** folder in the object tree.
2. Right-click the cache table that you want to drop a materialized query table from and click **Properties**. The Cache Table Properties window opens.
3. Select the materialized query table that you want to drop and click **Remove**.

Related tasks:

- "Adding a materialized query table to a cache table" on page 178
- "Dropping a cache table" on page 180

Dropping a cache table

When you no longer want to store data locally in a cache table, you can drop the cache table.

Dropping a cache table results in the following actions:

- The materialized query tables that were created for the cache table are dropped.
- The replication schedule between the data sources and the materialized query tables is removed.
- The nickname for the data source is dropped, if the nickname was created when you created the cache table. If you used an existing nickname when you created the cache table, the federated server does not drop the nickname.

Procedure:

To drop a cache table:

1. In the DB2 Control Center, expand the **Cache Objects** folder and the **Cache Tables** folder in the object tree.
2. Right-click the cache table that you want to drop and click **Drop**.

Related tasks:

- "Creating a cache table" on page 177
- "Dropping a materialized query table from a cache table" on page 179

Chapter 15. Informational constraints on nicknames in a federated system

This chapter describes how to use informational constraints on nicknames.

This chapter contains:

- "Informational constraints on nicknames"
- "Specifying informational constraints on nicknames"
- "Specifying informational constraints on nicknames - examples" on page 182

Informational constraints on nicknames

Informational constraints are rules that the optimizer can use to improve performance but that the database manager does not enforce. You can use informational constraints on nicknames to improve the performance of queries on remote data sources.

You can specify the following types of informational constraints for nicknames:

- Referential constraints
- Check constraints
- Functional dependency constraints
- Primary key constraints
- Unique constraints

Related concepts:

- "Constraints" in the *SQL Reference, Volume 1*

Related tasks:

- "Specifying informational constraints on nicknames" on page 181

Related reference:

- "Specifying informational constraints on nicknames - examples" on page 182

Specifying informational constraints on nicknames

To improve the performance of some queries on remote data sources, you can add informational constraints to nicknames. However, the federated server does not enforce or check the constraints.

For relational data sources, you can specify informational constraints when you alter a nickname.

For nonrelational data sources, you can specify informational constraints when you create a nickname or alter a nickname.

Restrictions:

After you define informational constraints on a nickname, you cannot alter the column names for that nickname until you remove the informational constraints.

Procedure:

You can specify informational constraints on a nickname from the DB2 Control Center or the DB2 command line.

To do this task from the DB2 Control Center:

1. Select the **Nicknames** folder:
   - If you are creating a nickname, in the object details pane of the DB2 Control Center, click **Create Nicknames** from the list of actions. The Nicknames wizard opens.

     From the Create Nicknames window, open the Add Nickname notebook or the Properties notebook for a nickname:
     - If you plan to create a single nickname, click **Add**. The Add Nickname notebook opens.
     - If you used the discover feature to generate a list of nicknames, select the nickname that you want to add informational constraints to. Then click **Properties**. The Properties notebook opens.
   - If you are altering a nickname, click on the nickname that you want to alter. In the object details pane of the DB2 Control Center, click **Alter** from the list of actions. The Alter Nickname notebook opens.
2. On the Keys page, set the referential integrity constraints for the nickname. You can set a primary key, unique key, or foreign key constraint.
3. On the Check Constraints page, set the check constraints or functional dependency constraints for the nickname.
4. Click **OK** to set the informational constraints and close the notebook.

To do this task from the DB2 command line:

Issue the CREATE NICKNAME statement or the ALTER NICKNAME statement with the appropriate constraint attributes set.

Related concepts:

- "Primary keys" in the *Administration Guide: Planning*
- "Informational constraints on nicknames" on page 181

Related reference:

- "Specifying informational constraints on nicknames - examples" on page 182

Specifying informational constraints on nicknames - examples

The following examples illustrate the use of informational constraints on nicknames.

Informational check constraint

In the following remote table, the data in the salary column is always greater than 10000.

```
CREATE TABLE account.salary (
    empno INTEGER NOT NULL PRIMARY KEY,
    salary  INTEGER NOT NULL
);
```

Create a nickname for this table:

```
CREATE NICKNAME account.salary FOR myserv.account.salary;
```

Then add informational check constraints for the nickname by issuing the following statement:

```
ALTER NICKNAME account.salary ADD CONSTRAINT cons1 CHECK( salary > 10000 )
NOT ENFORCED
ENABLE QUERY OPTIMIZATION;
```

Informational referential constraint: nickname to nickname

In this example, there are two nicknames N1 and N2. Column F1 of nickname N2 contains the key value in column P1 of nickname N1. You can define the referential constraint on nickname N2 by issuing the following statement:

```
ALTER NICKNAME SCHEMA1.N2 ADD CONSTRAINT ref1
                          FOREIGN KEY (F1) REFERENCES SCHEMA1.N1 (P1)
                          NOT ENFORCED;
```

Informational referential constraint: nickname to table

In this example, nickname N3 with column F1 contains the key value in column P1 of table T1. You can define the referential constraint on nickname N3 by issuing the following statement:

```
ALTER NICKNAME SCHEMA1.N3 ADD CONSTRAINT ref1
                          FOREIGN KEY (F1) REFERENCES SCHEMA1.T1 (P1)
                          NOT ENFORCED;
```

Informational referential constraint: table to nickname

In this example, table T2 with column F1 contains the key value in column P1 of nickname N4. You can define the referential constraint on table T2 by issuing the following statement:

```
ALTER TABLE SCHEMA1.T2 ADD CONSTRAINT ref1
                          FOREIGN KEY (F1) REFERENCES SCHEMA1.N4 (P1)
                          NOT ENFORCED;
```

Functional dependency

In this example, the column pair C1 and C2 uniquely determine the value in the column P1. You can define functional dependency by issuing the following statement:

```
ALTER NICKNAME SCHEMA1.NICK1 ADD CONSTRAINT FD1 CHECK( P1 DETERMINED BY (C1,C2) )
                         NOT ENFORCED ENABLE QUERY OPTIMIZATION;
```

Table-structured file

This example shows how to define a primary key for a table-structured file.

```
CREATE NICKNAME MY_FILE (
       X INTEGER NOT NULL,
       Y INTEGER,
       PRIMARY KEY (X) NOT ENFORCED
)  FOR SERVER MY_SERVER OPTIONS(FILE_PATH '/usr/pat/DRUGDATA1.TXT');
```

Star schema

The following example shows four dimension tables and one fact table.

```
CREATE TABLE SCHEMA.FACT (
       LOCATION_CODE   INTEGER NOT NULL,
       PRODUCT_CODE    INTEGER NOT NULL,
       CUSTOMER_CODE   INTEGER NOT NULL,
       SDATE           DATE NOT NULL,
       SALES           INTEGER NOT NULL
);

CREATE TABLE SCHEMA.LOCATION (
       LOCATION_CODE   INTEGER NOT NULL PRIMARY KEY,
       STATE           CHAR(2) NOT NULL,
       SHOP_ID         INTEGER NOT NULL,
       ...
);

CREATE TABLE SCHEMA.PRODUCT (
       PRODUCT_CODE    INTEGER NOT NULL PRIMARY KEY,
       PRODUCT_CAT     INTEGER NOT NULL,
       PRODUCT_NAME    VARCHAR(20) NOT NULL,
       ...
);

CREATE TABLE SCHEMA.CUSTOMER (
       CUSTOMER_CODE   INTEGER NOT NULL PRIMARY KEY,
       NAME            VARCHAR(20) NOT NULL,
       TEL             VARCHAR(10) NOT NULL,
       ...
);

CREATE TABLE SCHEMA.TIMEDIM (
       SDATE           DATE NOT NULL UNIQUE,
       YEAR            INTEGER NOT NULL,
       QUARTER         INTEGER NOT NULL,
       ...
);
```

The federated server creates the following nicknames for the fact table and the four dimension tables:

```
CREATE NICKNAME SCHEMA.FACT FOR SERVER.SCHEMA.FACT;
CREATE NICKNAME SCHEMA.LOCATION FOR SERVER.SCHEMA.LOCATION;
CREATE NICKNAME SCHEMA.PRODUCT FOR SERVER.SCHEMA.PRODUCT;
CREATE NICKNAME SCHEMA.CUSTOMER FOR SERVER.SCHEMA.CUSTOMER;
CREATE NICKNAME SCHEMA.TIMEDIM FOR SERVER.SCHEMA.TIMEDIM;
```

You can define the following foreign key relationship:

```
ALTER NICKNAME SCHEMA.FACT ADD CONSTRAINT L1 FOREIGN KEY (LOCATION_CODE)
                               REFERENCES SCHEMA.LOCATION(LOCATION_CODE)
                               NOT ENFORCED ENABLE QUERY OPTIMIZATION;

ALTER NICKNAME SCHEMA.FACT ADD CONSTRAINT P1 FOREIGN KEY (PRODUCT_CODE)
                               REFERENCES SCHEMA.PRODUCT(PRODUCT_CODE)
                               NOT ENFORCED ENABLE QUERY OPTIMIZATION;

ALTER NICKNAME SCHEMA.FACT ADD CONSTRAINT C1 FOREIGN KEY (CUSTOMER_CODE)
                               REFERENCES SCHEMA.CUSTOMER(CUSTOMER_CODE)
                               NOT ENFORCED ENABLE QUERY OPTIMIZATION;

ALTER NICKNAME SCHEMA.FACT ADD CONSTRAINT S1 FOREIGN KEY (SDATE)
                               REFERENCES SCHEMA.TIMEDIM(SDATE)
                               NOT ENFORCED ENABLE QUERY OPTIMIZATION;
```

When the value of the TEL column in the CUSTOMER nickname is unique, you can add the following informational unique constraint:

```
ALTER NICKNAME SCHEMA.CUSTOMER ADD CONSTRAINT U1 UNIQUE( TEL )
                                    NOT ENFORCED ENABLE QUERY OPTIMIZATION;
```

When the value of the SHOP_ID column in the LOCATION nickname uniquely determines the value of the LOCATION_ID column, you can define the following functional dependency:

```
ALTER NICKNAME SCHEMA.LOCATION
ADD CONSTRAINT F1 CHECK( LOCATION_ID DETERMINED BY SHOP_ID )
                         NOT ENFORCED ENABLE QUERY OPTIMIZATION;
```

Because the value of the QUARTER column in the TIMEDIM nickname is between 1 and 4, you can define the following informational check constraint:

```
ALTER NICKNAME SCHEMA.TIMEDIM
ADD CONSTRAINT Q1 CHECK( QUARTER BETWEEN 1 AND 4 )
                         NOT ENFORCED ENABLE QUERY OPTIMIZATION;
```

The statements in this example create nicknames for remote tables. Nicknames have primary keys when remote tables have primary keys. When you create nickname on views, nicknames do not have primary keys. In this case, you can change the nickname to add an informational primary key constraint. For example:

```
CREATE NICKNAME SCHEMA.LOCATION FOR SERVER.SH.V_LOCATION;
ALTER NICKNAME SCHEMA.LOCATION
      ADD CONSTRAINT P1 PRIMARY KEY ( LOCATION_CODE ) NOT ENFORCED;
```

Related tasks:

- "Specifying informational constraints on nicknames" on page 181

Chapter 16. Nickname statistics

This chapter describes how to retrieve nickname statistics.

This chapter contains:

Nickname statistics update facility - overview

When you register a nickname for a data source object, the federated server adds information about that data source object to the system catalog on the federated database. The DB2® UDB query optimizer uses this information to plan how to retrieve data from the data source object. The federated database does not automatically detect changes to the data source objects, so the information in the system catalog can become outdated. Retrieving the statistics for a nickname ensures that the query optimizer is using the most recent information about the nickname when it generates the query access plans.

You can retrieve the currently available statistics on database nicknames, columns, and indexes from the remote data source.

You can retrieve the statistics of a single nickname or all nicknames in a DB2 schema on a specific DB2 server definition. If any part of the retrieval fails, the database rolls back the changes.

For relational data sources and ODBC data sources where the statistics are not set to their defaults, you can retrieve the following statistics if they are available on the remote source:

- CARD
- FPAGES
- NPAGES
- OVERFLOW
- COLCARD
- HIGH2KEY
- LOW2KEY
- NLEAF
- NLEVELS
- CLUSTERFACTOR
- CLUSTERRATIO
- FULLKEYCARD
- FIRSTKEYCARD

For nonrelational data sources and ODBC data sources where the statistics are set to their defaults, you can retrieve the following statistics if they are available on the remote source:

- CARD
- COLCARD
- HIGH2KEY
- LOW2KEY
- FULLKEYCARD
- FIRSTKEYCARD

You can retrieve nickname statistics by using the DB2 Control Center or the DB2 Command Line Processor.

You can retrieve nickname statistics for the following data sources:

- BioRS
- DB2 family (DRDA)
- Documentum (except pseudo columns)
- Informix®
- Microsoft® SQL Server
- ODBC
- Oracle (NET8)
- Sybase (CTLIB)
- Table-structured files
- Teradata
- XML (on the root nickname)

You can retrieve nickname statistics on federated servers that use Microsoft Windows® 2000 and Windows NT®, AIX®, Solaris, HP-UX, and Linux operating systems.

Related tasks:

- "Retrieving nickname statistics" on page 188
- "Viewing the status of the updates to nickname statistics" on page 191

Related reference:

- "Retrieving nickname statistics from the command line - examples" on page 190
- Chapter 32, "SYSPROC.NNSTAT stored procedure," on page 299

Retrieving nickname statistics

Retrieving the statistics for a nickname ensures that the query optimizer uses the information about the nickname currently available at the data source. You can improve query performance by keeping the statistical information about a nickname updated. However, the statistics are only as accurate and up-to-date as the information currently on the remote source.

Prerequisites:

The following prerequisites apply when you use the command line prompt to update statistics:

- The federated server creates the log file on the server. The directories that you list in the path must exist.
- On Windows, use two backslashes to specify the log path. For example: `c:\\temp\\nnstat.log`.

Restrictions:

If the local column name or type is different than the remote column name or type, the nickname statistics update utility will not retrieve column statistics.

The user ID that you use to connect to the federated database must be mapped to the remote data source table.

Procedure:

You can update nickname statistics from the DB2 Control Center or the DB2 command line.

To do this task from the DB2 Control Center:

1. Select the nicknames that you want to retrieve current statistics for:
   - To update the statistics for all of the nicknames that are associated with a server definition:
     a. Expand the **Federated Database Objects** folder.
     b. Expand the **Server definitions** folder that contains the nicknames that you want to update.
     c. Right-click on the name of the server definition and select **Statistics**.
     d. Select **Update**. The Statistics Update notebook opens.
   - To update the statistics for a single nickname, expand the **Nicknames** folder.
     a. Right-click on the nickname that you want to update and select **Statistics**.
     b. Select **Update**. The Statistics Update window opens.
2. If a DB2 tools catalog does not exist, a window appears from which you can create the DB2 tools catalog.
3. Specify the settings for the update:
   - To update the statistics for all nicknames that are associated with a server definition:
     a. On the Nicknames page:
        – Select the schema that is associated with the nicknames that you want to update statistics for. If you do not select a schema, the federated server updates the statistics for all of the nicknames that are associated with the local schema.
        – Select an existing log file or type the fully qualified path for a new log file.
     b. On the Schedule page, specify when you want the update for the nickname statistics to run.
   - To update the statistics for a single nickname, specify when you want the update for the nickname statistics to run.

To do this task from the command line or within your application call the stored procedure SYSPROC.NNSTAT.

Related concepts:

- "Nickname statistics update facility - overview" on page 187

Related tasks:

- "Viewing the status of the updates to nickname statistics" on page 191

Related reference:

- "Retrieving nickname statistics from the command line - examples" on page 190
- Chapter 32, "SYSPROC.NNSTAT stored procedure," on page 299

Retrieving nickname statistics from the command line - examples

In this example, the federated server retrieves the statistics for all the nicknames on the DB2SERV server and does not create a log.

```
CALL SYSPROC.NNSTAT('DB2SERV','NULL','NULL','NULL',?,?)
```

In this example, the federated server retrieves the statistics for the nickname STAFF in the ADMIN schema. The federated server writes the log to the /home/iiuser/reportlogs/log1.txt file.

```
CALL SYSPROC.NNSTAT(
 'NULL', 'ADMIN','STAFF','/home/iiuser/reportlogs/log1.txt',?,?)
```

In this example, the federated server retrieves the statistics for all the nicknames on the DB2Serv server in the admin schema. The federated server writes the log to the /home/iiuser/stats/recent.log file.

```
CALL SYSPROC.NNSTAT(
 'DB2Serv', 'admin', 'NULL', '/home/iiuser/stats/recent.log', ?,?)
```

Related concepts:

- "Nickname statistics update facility - overview" on page 187

Related tasks:

- "Retrieving nickname statistics" on page 188

Creating a DB2 tools catalog

When you update the statistics for a nickname you can use a DB2 tools catalog in order to schedule the update. Scheduling the update is only available through the DB2 Control Center. If you do not have a DB2 tools catalog, you will be prompted to create a catalog.

Prerequisites:

The DB2 administration server must be installed.

Procedure:

You can create a DB2 tools catalog from DB2 Control Center or the command line prompt.

To do this task from the DB2 Control Center:

1. The Statistics Update window appears when you update nickname statistics
2. Select the system that you want to create a database on for the DB2 tools catalog.

The database must be on a cataloged system that currently has no metadata storage. If the system you want is not cataloged, you must catalog the system before you create the database for the DB2 tools catalog.

Related tasks:

- "Retrieving nickname statistics" on page 188
- "Tools catalog database and DAS scheduler setup and configuration" in the *Administration Guide: Implementation*

Related reference:

- "CREATE TOOLS CATALOG Command" in the *Command Reference*

Viewing the status of the updates to nickname statistics

After you request an update to the statistics for a nickname, you can view the status of the update. You can use this information to evaluate the performance of your queries.

Procedure:

You can view the status of updates to nickname statistics from the DB2 Control Center or the DB2 command line.

To do this task from the DB2 Control Center:

1. Select the nicknames that you want to view the status of the updates for:
   - To view the status of the updates for all nicknames that are associated with a specific server definition:
     a. Expand the **Federated Database Objects** folder.
     b. Expand the **Server definitions** folder that contains the nicknames that you want to update.
     c. Right-click on the name of the server definition and select **Statistics**.
   - If you want to view the status of the updates for a specific nickname, expand the **Nicknames** folder.
     a. Select the nickname that you want to update.
     b. Right-click on the nickname and select **Statistics**.
2. Select **View Results**. The Statistics View Results window opens.

To do this task from the command line, look in the SYSPROC.FED_STATS table to see a history of the updates.

Related concepts:

- "Nickname statistics update facility - overview" on page 187

Related tasks:

- "Retrieving nickname statistics" on page 188

Related reference:

- "Retrieving nickname statistics from the command line - examples" on page 190

Part 4. Application programming

Chapter 17. Application programming scenario

You can use the following references to learn how to use DB2 Information Integrator with IBM Websphere Portal to develop applications:

- *IBM Websphere Portal and DB2 Information Integrator*, at http://www.redbooks.ibm.com/redpieces/pdfs/sg246433.pdf
- Sample code from the scenario, at ftp://www.redbooks.ibm.com/redbooks/SG246433/

Chapter 18. Application programming for federated systems

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*

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 UDB 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 nonrelational 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 UDB recognizes.

The federated database system catalog contains information about the 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*.

Related concepts:

- "Querying data sources directly with pass-through" on page 208

Related tasks:

- "Reference data source objects by nicknames in SQL statements" on page 198
- "Nicknames in DDL statements" on page 198
- "Data source statistics impact applications" on page 199
- "Nicknames that invoke stored procedures" on page 200
- "Defining column options on nicknames" on page 200
- "Accessing data sources using pass-through sessions" on page 104

Working with nicknames in your applications

Reference 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.

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 table called PERSON.DEPT, where:

- 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 PERSON.DEPT is not allowed (except in a pass-through session). The federated server does not have PERSON.DEPT registered as a nickname.

Using nicknames in the CREATE TABLE statement:

Suppose that you want to create a local materialized query table (MQT) based on table for which you have defined a nickname. An example of the CREATE TABLE statement is:

```
CREATE TABLE table_name LIKE nickname
```

Related concepts:

- "Querying data sources directly with pass-through" on page 208

Related tasks:

- "Federated pass-through considerations and restrictions" on page 209
- "Pass-through sessions to Oracle data sources" on page 210

Nicknames in DDL statements

Data source objects must have nicknames registered in the federated database before you can include them in your DDL statements. Some examples of DDL statements you use with federated systems are:

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 with 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 with a nickname for certain privileges and for all users and groups. However, DB2 UDB does not issue a corresponding GRANT or REVOKE statement on 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 tasks:

- "Creating index specifications for data source objects" on page 72
- "The SQL statements you can use with nicknames" on page 98

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*

Data source statistics impact applications

When a nickname is created for a data source object, the federated database global catalog is updated with information about that object. DB2 query optimizer 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.

Catalog statistics stored in the global catalog:

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 UDB obtains the latest statistics. Run the data source equivalent of the RUNSTATS command on 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 and referencing the nickname for the table. You use the SPECIFICATION ONLY clause with the CREATE INDEX statement to produce only an index specification. You do not want to create an actual index.

To determine what data source information is stored in the global catalog, query the SYSCAT.TABLES and SYSCAT.COLUMNS catalog views. To determine what data source index information is stored in the catalog, or what a particular index specification contains, query the SYSCAT.INDEXES catalog view.

Changing applications to reference the SYSSTAT view instead of the SYSCAT view:

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 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 72
- "Nickname characteristics affecting global optimization" on page 147

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*
- Chapter 19, "Views in the global catalog table containing federated information," on page 213

Nicknames that invoke stored procedures

If you are migrating applications from DataJoiner that invoke stored procedures through nicknames, you need to modify your applications. DB2 Information Integrator does not currently support the ability to invoke a stored procedure using a nickname.

Defining column options on nicknames

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 that you provide through the column options is stored in the global catalog.

Nonrelational data sources

Column options are unique for each nonrelational wrapper. These options are typically set when you issue the CREATE NICKNAME statement.

Relational data sources

There are two column options you can use for relational data sources: NUMERIC_STRING and VARCHAR_NO_TRAILING_BLANKS.

Setting the NUMERIC_STRING column option

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 that use this column to be optimized for sorting operations and comparison operations. For example:

```
ALTER NICKNAME nickname
   ALTER COLUMN local_column_name
   OPTIONS (SET NUMERIC_STRING 'Y')
```

Setting the VARCHAR_NO_TRAILING_BLANKS column option

Some data sources, such as Oracle, do not use the same blank-padded string comparison logic as is used in DB2 for Linux, UNIX, and Windows. This applies to data types such as VARCHAR and VARCHAR2. As a result, predicates that involve these data types must be rewritten by the query optimizer to ensure consistent query results. Rewriting query statements can impact performance. Setting this option for a specific column provides the query optimizer with information about these columns so that it can generate more efficient SQL statements.

For example:

```
ALTER NICKNAME nickname
   ALTER COLUMN local_column_name
   OPTIONS (SET VARCHAR_NO_TRAILING_BLANKS'Y')
```

Related reference:

- Chapter 24, "Nickname column options for federated systems," on page 247

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 using nicknames, instead of using 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 that contain confidential or sensitive data from a view

Restrictions:

Federated views that are created from multiple data source objects are read-only views and cannot be updated.

Federated views that are created from only one data source object might or might not be read-only views.

- A federated view created from a single nonrelational 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 VIEW statement.

Procedure:

You create a federated view from 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.

Example: Creating a federated view that merges similar data from several data source objects:

Suppose that 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 VIEW FV1
   AS SELECT * FROM ORA_EU_CUST
   UNION
   SELECT * FROM SYB_AS_CUST
   UNION
   SELECT * FROM INFMX_SA_CUST
```

Example: Joining data to create a federated view:

Suppose that 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 made by those customers. Each table has a column containing the customer number (CUST_NO). The syntax to create a federated view from these two nicknames that joins this data is:

```
CREATE 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
```

Related tasks:

- "Accessing heterogeneous data through federated views" on page 105

Related reference:

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

Use 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 the application processes that use the package. Isolation levels are used when you prepare 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 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 that you request to a corresponding one at the data source. At each connection to the data source, the wrapper determines the DB2 for Linux, UNIX, and Windows isolation level. The remote data source isolation level is set to the equivalent level. If there is no exact equivalent, then the federated server sets the isolation level to the next stronger level. Once a connection is made to a data source, the isolation level for duration of the connection cannot be changed.

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 17. Comparable isolation levels between the federated server and supported data sources.

DB2 federated server	Cursor stability	Repeatable read	Read stability	Uncommitted read
DB2 family of products	Cursor stability	Repeatable read	Read stability	Uncommitted read
Informix	Cursor stability	Repeatable read	Repeatable read	Dirty read
Microsoft SQL Server	Read commited	Serializable	Repeatable read	Read Uncommited
ODBC	Read commited	Serializable	Repeatable read	Read Uncommited
Oracle	Read commited	Serializable	Serializable	Read commited
Sybase	Level 1	Level 3	Level 3	Level 0

OLE DB, Teradata, and nonrelational data sources do not have a concept similar to isolation levels as is used by DB2 for Linux, UNIX, and Windows. There is no mapping between the DB2 isolation levels and the OLE DB, Teradata, and nonrelational data sources.

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 197

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 data at DB2 UDB.

A federated system supports SELECT operations on LOBs at DRDA®, Informix®, Microsoft® SQL Server, Oracle, and Sybase data sources. For example:

```
SELECT empname, picture FROM infmx_emp_table
   WHERE empno = '01192345'
```

Where *picture* represents a LOB column and *infmx_emp_table* represents a nickname referencing an Informix table containing employee data.

A federated system supports SELECT, INSERT, UPDATE, and DELETE operations on LOBs at Oracle data sources (Version 8 or later).

The read and write operations supported by DB2 for Linux, UNIX®, and Windows®, Version 8 are listed in the following table:

Table 18. Read and write support for LOBs

Data source	Type of operations
DB2 family of products[1]	read only
BioRS	read only
BLAST	read and bind-in
Entrez	read only
HMMER	read and bind-in
Informix	read only
Microsoft SQL Server	read only
Oracle (NET8 wrapper)	read and write
ODBC	read only
Sybase	read only
Teradata	read only
Web services	read only and bind-out for CLOB only
WebSphere® Business Integration	read only and bind-out for CLOB only
XML	read only

Table 18. Read and write support for LOBs (continued)

Data source	Type of operations

Notes:

1. DB2 UDB for iSeries™ Version 5 (or later) is required for LOB support. DB2 Information Integrator Version 8 cannot access DB2 UDB for Linux, UNIX, and Windows Version 7 LOB data.

Teradata LOBs

Teradata LOBs are slightly different than DB2 LOBs. Teradata does not have any data types as large as the LOBs supported in DB2 UDB. However, there are some Teradata data types that can be up to 64000 bytes long. These data types are CHAR, VARCHAR, BYTE, VARBYTE, GRAPHIC, and VARGRAPHIC. These Teradata data types are mapped to DB2 LOB data types when the length of the Teradata data type exceeds the limits of the corresponding DB2 data type.

LOB lengths

Some data sources, such as Oracle and Informix, do not store the lengths of LOB columns in their system catalogs. When you create a nickname on a table, information from the data source system catalog is retrieved including column length. Since no length exists for the LOB columns, the federated database assumes that the length is the maximum length of a LOB column in DB2 for Linux, UNIX, and Windows. The federated database stores the DB2 for Linux, UNIX, and Windows maximum length in the federated database catalog as the length of the nickname column.

Related concepts:

- "LOB locators" on page 205
- "Restrictions on LOBs" on page 206

Related reference:

- Chapter 27, "Default forward data type mappings," on page 261

Federated LOB support-details

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® UDB can retrieve LOBs from remote data sources, store them at the federated server, and then issue a LOB locator on 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 204
- "Restrictions on LOBs" on page 206

Restrictions on LOBs

Federated systems have the following 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 204
- "LOB locators" on page 205

Distributed requests for querying data sources

Queries submitted to the federated database can request results from a single data source, but typically are requests that include 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 to 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
```

Optimizing distributed requests with server options

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 UDB 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. This server option is for Oracle data source only. Use this option only when you are certain that all VARCHAR2 columns for every object that is referenced by a nickname on the server has no trailing blanks. Otherwise,

use a column option to specify the columns for individual objects on the server that have no trailing blanks. The column option is also named VARCHAR_NO_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*. Plan hints can help a data source optimizer 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 server options that help optimize queries. For example, suppose that for data sources ORACLE1 and ORACLE2, the PLAN_HINTS server option is set to the default, 'N' (no, do not furnish this data source with plan hints). You write a distributed request that selects 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 to the data sources is terminated, the setting will automatically revert back to 'N'.

Procedure:

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.

Recommendation: Prepare the statement dynamically. The SET SERVER OPTION statement affects only dynamic SQL statements.

Related reference:

- "SET SERVER OPTION statement" in the *SQL Reference, Volume 2*
- Chapter 21, "Server options for federated systems," on page 219

Using pass-through 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.

Querying data sources directly with pass-through

Pass-through sessions are useful when:

- Applications must create objects at the data source or perform INSERT, UPDATE, or DELETE operations.
- DB2® UDB does not support a unique data source operation.

Procedure:

Use the SET PASSTHRU statement to start a pass-through session and access a server directly. This statement can be issued dynamically. An example of this statement is:

```
SET PASSTHRU ORACLE1
```

This SET PASSTHRU statement opens a pass-through session to the data source using the server name ORACLE1. ORACLE1 is the name you registered for the data source server when you created the server definition.

Once the pass-through session is opened, ensure that you use the true name of the object and not the nickname when you reference objects in a pass-through session. You must use the SQL dialect of the data source, unless DB2 UDB is the data source that is being referenced.

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 prepare statements dynamically in a pass-through session:

- 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, but does not end the pass-through session.

Federated pass-through considerations and restrictions

There are a number of considerations and restrictions to keep in mind when you use a pass-through session. The following considerations and restrictions apply 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 issue multiple SET PASSTHRU statements, however only the last session is active. When a new SET PASSTHRU statement is invoked, it terminates the previous SET PASSTHRU statement. You cannot pass through to more than one data source in the same pass-through session.
- 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.
- Parameter markers are not supported in pass-through sessions. Use host variables instead of parameter markers.
- You can use the WITH HOLD semantics on a cursor defined in a pass-through session. However, you will receive an error if you try to use the semantics (with a COMMIT) and the data source does not support the WITH HOLD semantics.
- 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 LOBs.
- Pass-through does not support stored procedure calls.
- Pass-through does not support the SELECT INTO statement.

Related concepts:

- "Pass-through sessions" on page 10
- "Querying data sources directly with pass-through" on page 208

Related tasks:

- "Pass-through sessions to Oracle data sources" on page 210
- "Accessing data sources using pass-through sessions" on page 104

Related reference:

- "SET PASSTHRU statement" in the *SQL Reference, Volume 2*

Pass-through sessions to Oracle data sources

Before submitting SQL statements to Oracle data sources in a pass-through session, consider the following:

- When a remote client issues a SELECT statement from a command line processor (CLP) in pass-through mode and the client code is an SDK before 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 higher.
- Any DDL statement issued on 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.

Part 5. Reference

Chapter 19. Views in the global catalog table containing federated information

Most of the catalog views in a federated database are the same as the catalog views in any other DB2 for Linux, UNIX, and Windows database. There are several unique views which contain information pertinent to a federated system, such as the SYSCAT.WRAPPERS view.

As noted in the DB2 for Linux, UNIX, and Windows Version 6 and Version 7 SQL Reference manuals, the DB2 Version 8 SYSCAT views are now read-only. If you issue an UPDATE or INSERT operation on a view in the SYSCAT schema, it will fail. Using the SYSSTAT views is the recommended way to update the system catalog. Change applications that reference the SYSCAT view to reference the updatable SYSSTAT view instead.

The following table lists the SYSCAT views which contain federated information. These are read-only views.

Table 19. Catalog views typically used with a federated system

Catalog views	Description
SYSCAT.CHECKS	Contains check constraint information that you defined.
SYSCAT.COLCHECKS	Contains columns referenced by a check constraint.
SYSCAT.COLUMNS	Contains column information about the data source objects (tables and views) that you created nicknames for.
SYSCAT.COLOPTIONS	Contains information about column option values that you set for a nickname.
SYSCAT.CONSTDEP	Contains the dependency of an informational constraint that you defined.
SYSCAT.DATATYPES	Contains data type information about local built-in and user-defined DB2 data types.
SYSCAT.DBAUTH	Contains the database authorities held by individual users and groups.
SYSCAT.FUNCMAPOPTIONS	Contains information about option values that you have set for a function mapping.
SYSCAT.FUNCMAPPINGS	Contains the function mappings between the federated database and the data source objects.
SYSCAT.INDEXCOLUSE	Contains columns that participate in an index.
SYSCAT.INDEXES	Contains index specifications for data source objects.
SYSCAT.KEYCOLUSE	Contains columns that participate in a key defined by a unique key, primary key, or foreign key constraint.
SYSCAT.REFERENCES	Contains information about referential constraints defined by you.

Table 19. Catalog views typically used with a federated system (continued)

Catalog views	Description
SYSCAT.ROUTINES	Contains local DB2 user-defined functions, or function templates. Function templates are used to map to a data source function.
SYSCAT.REVTYPEMAPPINGS	This view is not used. All data type mappings are recorded in the SYSCAT.TYPEMAPPINGS view.
SYSCAT.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.TABCONST	Each row represents a table and nickname constraints of type CHECK, UNIQUE, PRIMARY KEY, or FOREIGN KEY.
SYSCAT.TABLES	Contains information about each local DB2 table, federated view, and nickname that you create.
SYSCAT.TYPEMAPPINGS	Contains forward data type mappings and reverse data type mappings. The mapping is to local DB2 data types from data source data types. These mappings are used when you create a nickname on a data source object.
SYSCAT.USEROPTIONS	Contains user authorization information that you set when you create user mappings between the federated database and the data source servers.
SYSCAT.VIEWS	Contains information about local federated views that you create.
SYSCAT.WRAPOPTIONS	Contains information about option values that you have set for a wrapper.
SYSCAT.WRAPPERS	Contains the name of the wrapper and library file for each data source that you create a wrapper for.

The following table lists the SYSSTAT views which contain federated information. These are read-write views that contain statistics you can update.

Table 20. Federated updatable global catalog views

Catalog views	Description
SYSSTAT.COLUMNS	Contains statistical information about each column in the data source objects (tables and views) that you have created nicknames for. Statistics are not recorded for inherited columns of typed tables.
SYSSTAT.INDEXES	Contains statistical information about each index specification for data source objects.
SYSSTAT.ROUTINES	Contains statistical information about each user-defined function. Does not include built-in functions. Statistics are not recorded for inherited columns of typed tables.

Table 20. Federated updatable global catalog views (continued)

Catalog views	Description
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.

Chapter 20. Wrapper options for federated systems

Wrapper options are used to configure the wrapper or to define how the federated server uses the wrapper. Wrapper options can be set when you create or alter the wrapper.

All relational and nonrelational data sources use the DB2_FENCED wrapper option. The ODBC data source uses the MODULE wrapper option. The Entrez data source uses the EMAIL wrapper option.

Table 21. Wrapper options and their settings

Option	Valid settings		Default setting
DB2_FENCED	Specifies whether the wrapper runs in fenced or trusted mode.		Relational wrappers: N. Nonrelational wrappers from IBM: N. Nonrelational wrappers from third parties: Y.
	Y	The wrapper runs in fenced mode.	
	N	The wrapper runs in trusted mode.	
EMAIL	Specifies an e-mail address when you register the Entrez wrapper. This e-mail address is included with all queries and allows NCBI to contact you if there are problems, such as too many queries overloading the NCBI servers. This option is required.		
MODULE	Specifies the full path of the library that contains the ODBC Driver Manager implementation or the SQL/CLI implementation. Required for the ODBC wrapper on UNIX federated servers.		On Windows, the default value is odbc32.dll

Related concepts:

- "Parallelism with queries that reference nicknames" on page 155

Related tasks:

- "Trusted and fenced mode process environments" in the *IBM DB2 Information Integrator Wrapper Developer's Guide*
- "Altering a wrapper" on page 25
- "Registering wrappers for a data source" in the *IBM DB2 Information Integrator Data Source Configuration Guide*

Chapter 21. Server options for federated systems

Server options are used to describe a data source server. Server options specify data integrity, location, security, and performance information. Some server options are available for all data sources, and other server options are data source specific.

The common federated server options for relational data sources are:

- Compatibility options. COLLATING_SEQUENCE, IGNORE_UDT
- Data integrity options. IUD_APP_SVPT_ENFORCE
- Data and time options. DATEFORMAT, TIMEFORMAT, TIMESTAMPFORMAT
- Location options. CONNECTSTRING, DBNAME, IFILE
- Security options. FOLD_ID, FOLD_PW, INFORMIX_LOCK_MODE
- Performance options. COMM_RATE, CPU_RATIO, DB2_MAXIMAL_PUSHDOWN, IO_RATIO, LOGIN_TIMEOUT, PACKET_SIZE, PLAN_HINTS, PUSHDOWN, TIMEOUT, VARCHAR_NO_TRAILING_BLANKS

The following table lists the server definition server options applicable for each relational data source.

Table 22. Server options for relational data sources

Data Source	CODEPAGE	COLLATING_SEQUENCE	COMM_RATE	CONNECTSTRING	CPU_RATIO	DATEFORMAT	DB2_MAXIMAL_PUSHDOWN	DBNAME	FOLD_ID	FOLD_PW	IFILE	INFORMIX_LOCK_MODE	IO_RATIO	IUD_APP_SVPT_ENFORCE	LOGIN_TIMEOUT	NODE	PACKET_SIZE	PASSWORD	PLAN_HINTS	PUSHDOWN	TIMEOUT	TIMEFORMAT	TIMESTAMPFORMAT	VARCHAR_NO_TRAILING_BLANKS
DB2 UDB for iSeries		X	X		X		X	X	X	X			X	X				X		X				X
DB2 UDB for z/OS and OS/390		X	X		X		X	X	X	X			X	X				X		X				X
DB2 for VM and VSE		X	X		X		X	X	X	X			X	X				X		X				X
DB2 UDB for Linux, UNIX, and Windows		X	X		X		X	X	X	X			X	X				X		X				X
Informix		X	X		X		X	X	X	X		X	X	X		X		X		X				
Microsoft SQL Server	X	X	X		X		X	X	X	X			X	X		X		X		X				
ODBC	X	X	X		X	X	X	X	X	X			X	X		X		X		X		X	X	X

Table 22. Server options for relational data sources (continued)

Data Source	CODEPAGE	COLLATING_SEQUENCE	COMM_RATE	CONNECTSTRING	CPU_RATIO	DATEFORMAT	DB2_MAXIMAL_PUSHDOWN	DBNAME	FOLD_ID	FOLD_PW	IFILE	INFORMIX_LOCK_MODE	IO_RATIO	IUD_APP_SVPT_ENFORCE	LOGIN_TIMEOUT	NODE	PACKET_SIZE	PASSWORD	PLAN_HINTS	PUSHDOWN	TIMEOUT	TIMEFORMAT	TIMESTAMPFORMAT	VARCHAR_NO_TRAILING_BLANKS
OLE DB		X		X																				
Oracle		X	X		X		X		X	X			X			X		X	X	X				X
Sybase		X	X		X		X	X	X	X	X		X		X	X	X	X	X	X	X			
Teradata		X	X		X		X						X	X		X				X				

The following table lists the server definition server options applicable for each nonrelational data source, except WebSphere Business Integration. The server definition server options for WebSphere Business Integration are listed in Table 24 on page 221.

Table 23. Server options for nonrelational data sources.

Data Source	CASE_SENSITIVE	CONTENT_DIR	DAEMON_PORT	ES_HOST	ES_PORT	ES_TRACING	ES_TRACELEVEL	ES_TRACEFILENAME	HMMPFAM_OPTIONS	HMMSEARCH_OPTIONS	MAX_ROWS	NODE	OS_TYPE	PORT	PROCESSORS	PROXU_AUTHID	PROXY_PASSWORD	PROXY_SERVER_NAME	PROXY_SERVER_PORT	PROXY_TYPE	RDBMS_TYPE	SOCKET_TIMEOUT	TIMEOUT	TRANSACTIONS	USE_CLOB_SEQUENCE
BioRS	X											X		X									X		
BLAST			X									X													X
Documentum		X										X	X								X			X	
Entrez											X					X	X	X	X	X		X			
Excel																									
Extended Search				X	X	X	X	X																	
HMMER			X						X	X		X			X										X
Table-structured files																									
Web services																									
XML																X	X	X	X	X		X			

The following table lists the server definition server options applicable for WebSphere Business Integration data sources.

Table 24. Server options for WebSphere Business Integration data sources.

Data Source	APP_TYPE	FAULT_QUEUE	MQ_CONN_NAME	MQ_MANAGER	MQ_RESPONSE_TIMEOUT	MQ_SVRCONN_CHANNELNAME	REQUEST_QUEUE	RESPONSE_QUEUE
WebSphere Business Integration	X	X	X	X	X	X	X	X

The following table describes each server option and lists the valid and default settings.

Table 25. Server options and their settings

Option	Description and valid settings	Default setting
APP_TYPE	The type of remote application. Valid values are 'PSOFT', 'SAP', and 'SIEBEL'. This option is required.	None.

Table 25. Server options and their settings (continued)

Option	Description and valid settings	Default setting
CASE_SENSITIVE	Specifies whether the BioRS server treats names in a case sensitive manner. Valid values are Y or N. 'Y' The BioRS server treats names in a case sensitive manner. 'N' The BioRS server does not treat names in a case sensitive manner In the BioRS product, a configuration parameter controls the case sensitivity of the data that is stored on the BioRS server. The CASE_SENSITIVE option is the DB2 Information Integrator counterpart to that BioRS system configuration parameter. You must synchronize the BioRS server case sensitivity configuration settings in your BioRS system and in DB2 Information Integrator. If you do not keep the case sensitivity configuration settings synchronized between BioRS and DB2 Information Integrator, errors will occur when you attempt to access BioRS data through DB2 Information Integrator. You cannot change or delete the CASE_SENSITIVE option after you create a new BioRS server in DB2 Information Integrator. If you need to change the CASE_SENSITIVE option, you must drop and then create the entire server again. If you drop the BioRS server, you must also create all of the corresponding BioRS nicknames again. DB2 Information Integrator automatically drops all nicknames that correspond to a dropped server.	Y
CODEPAGE	Specifies the DB2 code page identifier corresponding to the coded character set of the data source client configuration. You must specify the client's code page if the client's code page and the federated database code page do not match. For data sources that support Unicode, the CODEPAGE option can be set to the DB2 code page identifier corresponding to the supported Unicode encoding of the data source client.	On UNIX or Windows systems with a non-Unicode federated database: The federated database code page. On UNIX systems with a Unicode federated database: 1208 On Windows systems with a Unicode federated database: 1202

Table 25. Server options and their settings (continued)

Option	Description and valid settings	Default setting
COLLATING_ SEQUENCE	Specifies whether the data source uses the same default collating sequence as the federated database, based on the NLS code set and the country/region information. 'Y' The data source has the same collating sequence as the DB2 federated database. 'N' The data source has a different collating sequence than the DB2 federated database collating sequence. 'I' The data source has a different collating sequence than the DB2 federated database collating sequence, and the data source collating sequence is insensitive to case (for example, 'STEWART' and 'StewART' are considered equal).	'N'
COMM_RATE	Specifies the communication rate between the federated server and the data source server. Expressed in megabytes per second. Valid values are greater than 0 and less than $1x10^{23}$. Values can be expressed in any valid REAL notation.	'2'
CONTENT_DIR	Specifies the name of the locally-accessible root directory for storing content files retrieved by the GET_FILE, GET_FILE_DEL, GET_RENDITION, and GET_RENDITION_DEL pseudo columns. It must be writable by all users who can use these pseudo columns.	On UNIX systems: '/tmp' On Windows systems: 'C:\temp'
CONNECTSTRING	Specifies initialization properties needed to connect to an OLE DB provider.	None.
CPU_RATIO	Indicates how much faster or slower a data source CPU runs than the federated server CPU. Valid values are greater than 0 and less than $1x10^{23}$. Values can be expressed in any valid REAL notation. A setting of 1 indicates that the DB2 federated CPU speed and the data source CPU speed have the same CPU speed, a 1:1 ratio. A setting of .5 indicates that the DB2 federated CPU speed is 50% slower than the data source CPUO speed. A setting of 2 indicates that the DB2 federated CPU speed is twice as fast as the data source CPU speed.	'1.0'
DATEFORMAT	The date format used by the data source. Enter the format using 'DD', 'MM', and 'YY' or 'YYYY' to represent the numeric form of the date. You should also specify the delimiter such as a space or comma. For example, to represent the date format for '2003-01-01', use 'YYYY-MM-DD'. This field is nullable.	None.

Table 25. Server options and their settings (continued)

Option	Description and valid settings	Default setting
DAEMON_PORT	Specifies the port number on which the daemon will listen for BLAST or HMMER job requests. The port number must be the same number specified in the DAEMON_PORT option of the daemon configuration file.	BLAST: '4007'; HMMER: '4098'
DB2_MAXIMAL_ PUSHDOWN	Specifies the primary criteria that the query optimizer uses when choosing an access plan. The query optimizer can choose access plans based on cost or based on the user requirement that as much query processing as possible be performed by the remote data sources. 'Y' The query optimizer chooses an access plan that pushes down more query operations to the data source than other plans. When several access plans provide the same amount of pushdown, the query optimizer then chooses the plan with the lowest cost. If a materialized query table (MQT) on the federated server can process part or all of the query, then an access plan that includes the materialized query table is might be used. The federated database does not push down queries that result in a Cartesian product. 'N' The query optimizer chooses an access plan based on cost.	'N'
DBNAME	Name of the data source database that you want the federated server to access. For DB2 database, this value corresponds to a specific database for the initial remote DB2 database connection. This specific database is the database alias for the remote DB2 database that is cataloged at the federated server using the CATALOG DATABASE command or the DB2 Configuration Assistant. Does not apply to Oracle data sources because Oracle instances contain only one database.	None.
ES_HOST	Specifies the fully qualified host name or IP address of the Extended Search server that you want to search. This option is required.	None.
ES_PORT	Specifies the port number where this Extended Search server listens for requests. This option is optional.	'6001'
ES_TRACING	Specifies whether tracing should be enabled for error messages, warning messages, and informational messages that are produced by the remote Extended Search server. Valid values are: 'OFF' No trace messages will be logged. 'ON' Trace messages will be logged. This option is optional.	'OFF'

Table 25. Server options and their settings (continued)

Option	Description and valid settings	Default setting

ES_TRACELEVEL	If tracing is enabled, this option specifies the types of messages that will be written to the log file. You can enable and disable the following trace levels independently:
	'C' Critical error messages.
	'N' Noncritical messages.
	'W' Warning messages.
	'I' Informational messages.
	For example:
	`ES_TRACELEVEL 'W'`
	`ES_TRACELEVEL 'CN'`
	This option is optional.
		'C'

ES_TRACEFILENAME	If tracing is enabled, this option specifies the name of a directory and file where messages will be written. This option is optional.	For UNIX operating systems: $INSTHOME/sqllib/log/ESWrapper.log. For Windows operating systems: %DB2TEMPDIR%\ESWrapper.log.
FAULT_QUEUE	The name of the fault queue that delivers error messages from the adapter to the wrapper. The name must conform to the specifications for queue names for WebSphere MQ. This is a required option.	None.
FOLD_ID (See notes 1 and 4 at the end of this table.)	Applies to user IDs that the federated server sends to the data source server for authentication. Valid values are: 'U' The federated server folds the user ID to uppercase before sending it to the data source. This is a logical choice for DB2 family and Oracle data sources (See note 2 at end of this table.) 'N' The federated server does nothing to the user ID before sending it to the data source. (See note 2 at end of this table.) 'L' The federated server folds the user ID to lowercase before sending it to the data source. If 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.

Table 25. Server options and their settings (continued)

Option	Description and valid settings	Default setting
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.
HMMPFAM_OPTIONS	Specifies hmmpfam options such as --null2, --pvm, and --xnu that have no corresponding column name in a reference table that maps options to column names. For example: `HMMPFAM_OPTIONS '--xnu --pvm'` In this example, the daemon runs the HMMPFAM program with options from the WHERE clause of the query, plus the additional options --xnu --pvm.	
HMMSEARCH_ OPTIONS	Allows the user to provide additional command line options to the hmmsearch command. Only valid with type SEARCH. See the HMMER User's Guide for more information.	None.
IFILE	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.

Table 25. Server options and their settings (continued)

Option	Description and valid settings	Default setting
INFORMIX_LOCK_ MODE	Specifies the lock mode to be set for an Informix data source. The Informix wrapper issues the 'SET LOCK MODE' command immediately after establishing the connection to an Informix data source. Valid values are: **'W'** Sets the Informix lock mode to WAIT. If the wrapper tries to access a locked table or row, Informix waits until the lock is released. **'N'** Sets the Informix lock mode to NOWAIT. If the wrapper tries to access a locked table or row, Informix returns an error. **'n'** Sets the Informix lock mode to WAIT *n* seconds. If the wrapper tries to access a locked table or row and the lock is not released within the specified number of seconds, Informix returns an error.	'W'
IO_RATIO	Denotes how much faster or slower a data source I/O system runs than the federated server I/O system. Valid values are greater than 0 and less than $1x10^{23}$. Values can be expressed in any valid REAL notation. A setting of 1 indicates that the DB2 federated I/O speed and the data source I/O speed have the same I/O speed, a 1:1 ratio. A setting of .5 indicates that the DB2 federated I/O speed is 50% slower than the data source I/O speed. A setting of 2 indicates that the DB2 federated I/O speed is twice as fast as the data source I/O speed.	'1.0'
IUD_APP_SVPT_ ENFORCE	Specifies whether the DB2 federated system should enforce detecting or building of application savepoint statements. When set using the SET SERVER OPTION statement, this server option will have no effect with static SQL statements. 'Y' The federated server rolls back insert, update, or delete transactions if an error occurs in an insert, update, or delete operation and the data source does not enforce application savepoint statements. SQL error code SQL1476N is returned. 'N' The federated server will not roll back transactions when an error is encountered. Your application must handle the error recovery.	'Y'
LOGIN_TIMEOUT	Specifies the number of seconds for the DB2 federated server to wait for a response from Sybase Open Client to the login request. The default values are the same as for TIMEOUT.	'0'

Table 25. Server options and their settings (continued)

Option	Description and valid settings	Default setting
MAX_ROWS	Specifies the number of rows that the federated server returns for a query that uses the Entrez wrapper. You can specify only positive numbers and zero. When you set the option to be zero, you enable queries to retrieve an unlimited number of rows from the NCBI Web site. However, setting the MAX_ROWS server option to zero or to a very high number can impact your query performance. The MAX_ROWS server option is not required.	Microsoft Windows operating systems: 2000 rows. UNIX-based operating systems: 5000 rows.
MQ_CONN_NAME	The hostname or network address of the computer where the Websphere MQ server is running. An example of a connection name is: 9.30.76.151(1420) where 1420 is the port number. If the port number is excluded a default value of 1414 will be used. This option is optional. If it is omitted, the MQSERVER environment variable (if sepcified in db2dj.ini file) is used to select the channel definition. If MQSERVER is not set, the client channel table is used.	The wrapper uses the MQSERVER environment variable, if specified in the db2dj.ini file, to select the channel definition. If the MQSERVER environment variable is not set, the wrapper uses the client channel table.
MQ_MANAGER	The name of the WebSphere MQ manager. Any valid WebSphere MQ manager name. This option is required.	None.
MQ_RESPONSE_ TIMEOUT	The amount of time that the wrapper should wait for a response message from the response queue. The value is in milliseconds. You can specify a special value of -1 to indicate that there is no timeout period. This option is optional.	10000
MQ_SVRCONN_ CHANNELNAME	The name of the server-connection channel on the Webspehere MQ Manager that the wrapper should try to connect to. This parameter can be specified only if the MQ_CONN_NAME server option is specified. The default server-connection channel, SYSTEM.DEF.SVRCONN, is used if this option is omitted.	SYSTEM.DEF.SVRCONN
NODE	Relational data sources: Name by which a data source is defined as an instance to its RDBMS. Documentum: Specifies the actual name of the Documentum Docbase. This option is required. BLAST: Specifies the host name of the system on which the BLAST daemon process is running. This option is required. HMMER: Specifies the host name of the server on which the HMMER daemon process runs. This option is required. BioRS: Specifies the host name of the system on which the BioRS query tool is available. This option is optional.	BioRS: *localhost*

Table 25. Server options and their settings (continued)

Option	Description and valid settings	Default setting
OS_TYPE	Specifies the Docbase server's operating system. Valid values are AIX, SOLARIS, and WINDOWS. This option is required.	None.
PACKET_SIZE	Specifies the packet size of the Sybase interfaces file in bytes. If the data source does not support the specified packet size, the connection will fail. Increasing the packet size when each record is very large (for example, when inserting rows into large tables) significantly increases performance. The byte size is a numeric value.	
PASSWORD	Specifies whether passwords are sent to a data source. 'Y' Passwords are sent to the data source and validated. 'N' Passwords are not sent to the data source and not validated.	'Y'
PLAN_HINTS	Specifies whether *plan hints* are to be enabled. Plan hints are statement fragments that provide extra information for data source optimizers. This information can, for certain query types, improve query performance. The plan hints can help the data source optimizer decide whether to use an index, which index to use, or which table join sequence to use. 'Y' Plan hints are to be enabled at the data source if the data source supports plan hints. 'N' Plan hints are not to be enabled at the data source. This option is only available for Oracle and Sybase data sources.	'N'
PORT	Specifies the number of the port the wrapper uses to connect to the BioRS server. This option is optional.	'5014'
PROCESSORS	Specifies the number of processors that the HMMER program uses. This option is equivalent to the --cpu option of the `hmmpfam` command.	None.
PROXY_AUTHID	Specifies the user name to use when the value of PROXY_TYPE is 'SOCKS5'. This field is optional if the value of PROXY_TYPE is 'SOCKS5'. Contact your network administrator for the user name to use. This option is invalid if the PROXY_TYPE is not 'SOCKS5'.	None.
PROXY_PASSWORD	Specifies the password to use when the value of PROXY_TYPE is 'SOCKS5'. This field is optional if the value of PROXY_TYPE is 'SOCKS5'. Contact your network administrator for the password to use. This option is invalid if the PROXY_TYPE is not 'SOCKS5'.	None.

Table 25. Server options and their settings (continued)

Option	Description and valid settings	Default setting
PROXY_SERVER_ NAME	Specifies the proxy server name or the IP address. This field is required if the value of PROXY_TYPE is 'HTTP', 'SOCKS4', or 'SOCKS5'. Contact your network administrator for the proxy server name or the IP address.	None.
PROXY_SERVER_ PORT	Specifies the proxy server port number. This field is required if the value of PROXY_TYPE is 'HTTP', 'SOCKS4', or 'SOCKS5'. Contact your network administrator for the proxy server port number that should be used.	None.
PROXY_TYPE	Specifies the proxy type that is used to access the Internet when behind a firewall. The valid values are 'NONE', 'HTTP', 'SOCKS4', or 'SOCKS5'. The default value is 'NONE'. Contact your network administrator for the type of proxy that is used.	'NONE'
PUSHDOWN	'Y' DB2 UDB will consider letting the data source evaluate operations. 'N' DB2 UDB will send the data source SQL statements that include only SELECT with column names. Predicates (such as WHERE=) column and scalar functions (such as MAX and MIN), sorts (such as ORDER BY or GROUP BY), and joins will not be included in any SQL sent to the data source.	'Y'
RDBMS_TYPE	Specifies the RDBMS used by the Docbase. Valid values are DB2, INFORMIX, ORACLE, SQLSERVER or SYBASE. This option is required.	None.
RESPONSE_QUEUE	The name of the response queue that delivers query results from the adapter to the wrapper. The name must conform to the specifications for queue names for WebSphere MQ. This option is required.	None.
REQUEST_QUEUE	The name of the request queue that delivers query requests from the wrapper to the adapter. The name must conform to the specifications for queue names for WebSphere MQ. This option is required.	None.
SOCKET_TIMEOUT	Specifies the maximum time in minutes that the DB2 federated server will wait for results from the proxy server. A valid value is any number that is greater than or equal to zero. The default is zero '0'. A value of zero denotes an unlimited amount of time to wait.	0
TIMEFORMAT	The time format used by the data source. Enter the format using 'hh12', 'hh24', 'mm', 'ss', 'AM', or 'A.M'. For example, to represent the time format of '16:00:00', use 'hh24:mm:ss'. To represent the time format of '8:00:00 AM', use 'hh12:mm:ss AM'. This field is nullable.	None.

Table 25. Server options and their settings (continued)

Option	Description and valid settings	Default setting
TIMESTAMPFORMAT	The timestamp format used by the data source. The format follows that for date and time, plus 'n' for tenth of a second, 'nn' for hundredth of a second, 'nnn' for milliseconds, and so on, up to 'nnnnnn' for microseconds. For example, to represent the timestamp format of '2003-01-01-24:00:00.000000', use 'YYYY-MM-DD-hh24:mm:ss.nnnnnn'. This field is nullable.	None.
TIMEOUT	Sybase: 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 UDB to wait indefinitely for a response. BioRS: Specifies the time, in minutes, that the BioRS wrapper should wait for a response from the BioRS server. The default value is 10. This option is optional.	'0'; BioRS: '10'
TRANSACTIONS	Specifies the server transaction mode. The valid values are: 'NONE' No transactions are enabled. 'QUERY' Transactions are enabled only for Dctm_Query methods. **'ALL'** Transactions are enabled for the Dctm_Query method. ALL has the same function as QUERY in this release.	'QUERY'
USE_CLOB_ SEQUENCE	This option specifies the data type the federated server uses for the BlastSeq or HmmQSeq column. The values can be 'Y' or 'N'. You can use the CREATE NICKNAME or ALTER NICKNAME statement. to override the default data type for the BlastSeq or HmmQSeq column.	'Y'

Table 25. Server options and their settings (continued)

Option	Description and valid settings	Default setting
VARCHAR_NO_ TRAILING_BLANKS	This option applies to data sources which have variable character data types that do not pad the length with trailing blanks during comparison. Some data sources, such as Oracle, do not have blank-padded character comparison semantics that return the same results as the DB2 for Linux, UNIX, and Windows comparison semantics. Set this option when you want it to apply to all the VARCHAR and VARCHAR2 columns in the data source objects that will be accessed from the designated server. This includes views. Y — Trailing blanks are absent from these VARCHAR columns, or the data source has blank-padded character comparison semantics that are similar to the semantics on the federated server. The federated server pushes down character comparison operations to the data source for processing. N — Trailing blanks are present in these VARCHAR columns and the data source has blank-padded character comparison semantics that are different than the federated server. The federated server processes character comparison operations if it is not possible to compensate for equivalent semantics. For example, rewriting the predicate.	N for affected data sources.

Notes on this table:

1. This field is applied regardless of the value specified for authentication.
2. Because DB2 UDB stores user IDs in uppercase, the values 'N' and 'U' are logically equivalent to each other.
3. The setting for FOLD_PW has no effect when the setting for password is 'N'. Because no password is sent, case cannot be a factor.
4. Avoid null settings for either of these options. A null setting can seem attractive because DB2 UDB will make multiple attempts to resolve user IDs and passwords; however, performance might suffer (it is possible that DB2 UDB will send a user ID and password four times before successfully passing data source authentication).

Related concepts:

- "Server characteristics affecting pushdown opportunities" on page 134
- "Server characteristics affecting global optimization" on page 145

Related tasks:

- "Registering server definitions for a data source" in the *IBM DB2 Information Integrator Data Source Configuration Guide*

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*

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

Chapter 22. User mapping options for federated systems

These options are valid for all relational data sources. For nonrelational data sources, the REMOTE_AUTHID and REMOTE_PASSWORD options are valid for the following data sources: BioRS, Documentum, Extended Search, and Web services. The GUEST option is valid for the BioRS data source.

These options are used with the CREATE USER MAPPING and ALTER USER MAPPING statements.

Table 26. User mapping options and their settings

Option	Valid settings	Default setting
ACCOUNTING	DRDA: Used to specify a DRDA accounting string. Valid settings include any string of length 255 or less. This option is required only if accounting information needs to be passed. See the DB2 Connect Users Guide for more information.	None
GUEST	Specifies if the wrapper is to use the guest access mode to the BioRS server. **Y** The wrapper uses the guest access mode to the BioRS server. **N** The wrapper does not use the guest access mode to the BioRS server. When set to a value of Y, this option is mutually exclusive with the REMOTE_AUTHID option and the REMOTE_PASSWORD option.	N
REMOTE_AUTHID	Indicates the authorization ID used at the data source. Valid settings include any string of length 255 or less.	The authorization ID you use to connect to the DB2 Universal Database.
REMOTE_DOMAIN	Documentum: Indicates the Windows NT domain used to authenticate users connecting to a Documentum data source. Valid settings include any valid Windows NT domain name.	The default authentication domain for the Documentum database.
REMOTE_PASSWORD	Indicates the authorization password used at the data source. Valid settings include any string of length 32 or less. You do not need to set this option if the following conditions are met: • The database manager configuration parameter AUTHENTICATON is set to SERVER. • When you connected to the DB2 database, you specified an auth ID and password. If your server requires a password and you do not set this option, you must ensure both of the previous conditions are met or the connection will fail.	The password you use to connect to the DB2 Universal Database if both conditions listed in the valid settings column are met.

Related concepts:

- "DB2 Connect and DRDA" in the *DB2 Connect User's Guide*
- "DRDA and data access" in the *DB2 Connect User's Guide*

Related tasks:

- "Registering user mappings for a data source" in the *IBM DB2 Information Integrator Data Source Configuration Guide*

Chapter 23. Nickname options for federated systems

Table 27 and Table 28 list the nickname options for each data source. Table 29 on page 238 describes each nickname option and lists the valid and default settings.

Table 27. Available nickname options – A through P

Data source	ALL_VERSIONS	APPLICATIONID	BUSOBJ_NAME	CATEGORY	COLUMN_DELIMITER	DATASOURCE	DIRECTORY_PATH	FILE_PATH	FOLDERS	HMMTYPE	INSTANCE_PARSE_TIME	IS_REG_TABLE	KEY_COLUMN	MAXHIT	NAMESPACES	NEXT_TIME	PARENT	PROCESSORS
BioRS																		
BLAST						X												X
Documentum	X								X			X						
Entrez																	X	
Excel								X										
Extended Search		X		X		X								X				
HMMER						X				X								
Table-structured files					X			X					X					
Web services															X			
WebSphere Business Integration			X												X			
XML							X	X			X					X		

Table 28 lists the nickname options, R through X, for each data source.

Table 28. Available nickname options – R through X

Data Source	RANGE	REMOTE_OBJECT	SOAPACTION	SORTED	SORTFIELD	SORTORDER	STREAMING	TEMPLATE	TOTALMAXHIT	TIMEOUT	URL	VALIDATE	VALIDATE_DATA_FILE	VERTICAL_TABLE	XPATH	XPATH_EVAL_TIME
BioRS		X														
BLAST										X						
Documentum		X														
Entrez		X														
Excel	X															
Extended Search					X	X			X	X				X		
HMMER										X						

Table 28. Available nickname options – R through X (continued)

Data Source	RANGE	REMOTE_OBJECT	SOAPACTION	SORTED	SORTFIELD	SORTORDER	STREAMING	TEMPLATE	TOTALMAXHIT	TIMEOUT	URL	VALIDATE	VALIDATE_DATA_FILE	VERTICAL_TABLE	XPATH	XPATH_EVAL_TIME
Table-structured files				X									X			
Web services			X				X	X			X				X	
WebSphere Business Integration								X							X	
XML							X					X			X	X

Table 29 describes each nickname option and lists the valid and default settings.

Table 29. Nickname options and their settings

Option	Description and valid settings	Default setting
ALL_VERSIONS	Specifies whether all object versions will be searched. The valid values are y, Y, n, and N. The default value of N means that only the current object versions are included in query processing. This option is invalid when IS_REG_TABLE = 'Y'.	N
APPLICATIONID	Specifies the name of the Extended Search application that you want to search. This name must exist in the Extended Search configuration database. This option is required.	
BUSOBJ_NAME	The name of the XML schema definition file (.xsd) that represents the business object. For example sap_bapi_customer_get_detail2 . This option must be specified in a parent nickname.	
CATEGORY	Specifies one or more Extended Search categories that you want to search. If you omit this option, you must specify at least one data source name. To specify multiple categories, delimit the category names with a semicolon.	
COLUMN_DELIMITER	The delimiter that is used to separate columns of a table-structured file, enclosed in single quotation marks. The column delimiter can be more than one character in length. If no column delimiter is defined, the default delimiter is a comma. A single quotation mark cannot be used as a delimiter. The column delimiter must be consistent throughout the file. A null value is represented by two delimiters next to each other or a delimiter followed by a line terminator, if the NULL field is the last one on the line. The column delimiter cannot exist as valid data for a column.	The default delimiter is a comma.

Table 29. Nickname options and their settings (continued)

Option	Description and valid settings	Default setting
DATASOURCE	For Extended Search: Specifies one or more Extended Search data sources that you want to search. If you omit this option, you must specify at least one category name. To specify multiple data sources, delimit the data source names with a semicolon. For BLAST: The name of the data source on which the BLAST search will run. The same string that is used here must be present in the configuration file of the BLAST daemon. This option is required. For HMMER (type PFAM): The name of the HMM Profile database that is to be searched by HMMPFAM. The same string that is used here must be present in the configuration file of the HMMER daeamon. This option is required. For HMMER (type SEARCH): The name of the sequence file that is to be searched by HMMSEARCH. The same string that is used here must be present in the configuration file of the HMMER daeamon. This option is required.	
DIRECTORY_PATH	Specifies the path name of a directory that contains one or more XML files. Use this option to create a single nickname over multiple XML source files. The XML wrapper uses only the files with an .xml extension that are located in the directory that you specify. The XML wrapper ignores all other files in this directory. If you specify this nickname option, do not specify a DOCUMENT column. This option is accepted only for the root nickname (the nickname that identifies the elements at the top level of the XML document).	
FILE_PATH	For Microsoft Excel: Specifies the fully qualified directory path and file name of the Excel spreadsheet that you want to access. This option is required. For table-structured files: The fully qualified path to the table-structured file to be accessed, enclosed in single quotation marks. The data file must be a standard file or a symbolic link, rather then a pipe or another non-standard file type. Either the FILE_PATH or the DOCUMENT nickname column option must be specified. If the FILE_PATH nickname option is specified, then no DOCUMENT nickname column option can be specified. For XML: Specifies the file path of the XML document. If you specify this nickname option, do not specify a DOCUMENT column. This option is accepted only for the root nickname (the nickname that identifies the elements at the top level of the XML document).	

Table 29. Nickname options and their settings (continued)

Option	Description and valid settings	Default setting
FOLDERS	Specifies a string that contains one or more logically combined and syntactically correct Documentum FOLDER predicates. Specifying FOLDER predicates restricts the set of documents that are represented by this nickname to the documents in the designated folders. When you specify this option, enclose the entire value of the FOLDERS option in single quotation marks and use double quotation marks in place of the single quotation marks within the string. This option is not valid when IS_REG_TABLE = 'Y'.	
HMMTYPE	Optional: The alphabet that is used in both models and gene sequences. The value can be either NUCLEIC or PROTEIN and is not case sensitive.	PROTEIN
INSTANCE_PARSE_TIME	Specifies the time (in milliseconds) to parse the data in one row of the XML source document. You can modify the INSTANCE_PARSE_TIME, XPATH_EVAL_TIME, and NEXT_TIME options to optimize queries of large or complex XML source structures. This option is accepted only for columns of the root nickname (the nickname that identifies the elements at the top level of the XML document). The number that you specify can be an integer or a decimal value.	7
IS_REG_TABLE	Specifies whether the object that is specified by the REMOTE_OBJECT option is a Documentum registered table. The valid values are 'y', 'Y', 'n', and 'N'. You cannot change a nickname from a Documentum object to a registered table (or back) by changing this option with the ALTER NICKNAME statement. Instead, you must drop and recreate the nickname.	N

Table 29. Nickname options and their settings (continued)

Option	Description and valid settings	Default setting
KEY_COLUMN	The name of the column in the file that forms the key on which the file is sorted, enclosed in single quotation marks. Use this option for sorted files only. A column that is designated with the DOCUMENT nickname column option must not be specified as the key column. Only single-column keys are supported. Multi-column keys are not allowed. The value must be the name of a column that is defined in the CREATE NICKNAME statement. The column must be sorted in ascending order. The key column must be designated not nullable by adding the NOT NULL option to its definition in the nickname statement. This option is case-sensitive. However, DB2 UDB changes column names to uppercase unless the column is defined with double quotation marks.	If the value is not specified for a sorted nickname, the value is the name of the first column in the nicknamed file.
MAXHIT	An INTEGER that specifies the maximum number of results that can be returned from each source that is being searched.	50

NAMESPACES	The namespaces that are associated with the namespace prefixes that is used in the XPATH and TEMPLATE options for each column. The syntax is:
	`NAMESPACES 'prefix1= "actual_namespace1", prefix2="actual_namespace2" '`
	
	Separate each namespace with a comma. For example:
	`NAMESPACES ' c="http://www.myweb.com/cust", i="http://www.myweb.com/cust/id", n="http://www.myweb.com/cust/name"'`	

NEXT_TIME	Specifies the time (in milliseconds) that is required to locate subsequent source elements from the XPath expression. You can modify the NEXT_TIME, XPATH_EVAL_TIME, and INSTANCE_PARSE_TIME options to optimize queries of large or complex XML source structures. This option is accepted for root nicknames and non-root nicknames.	1
PARENT	Specified only for a child nickname whose parent was renamed through the REMOTE_OBJECT option. The PARENT option associates a child with a parent when multiple nickname families are defined within a DB2 schema. This name is case-sensitive.	
PROCESSORS	Specifies the number of processors to be used when a BLAST query is evaluated. This option corresponds to the blastall -a option.	1
RANGE	Specifies a range of cells to be used in the data source.	

Table 29. Nickname options and their settings (continued)

Option	Description and valid settings	Default setting
REMOTE_OBJECT	For BioRS: Specifies the name of the BioRS databank that is associated with the nickname. This name determines the schema and the BioRS databank for the nickname. This name also specifies the relationship of the nickname to other nicknames. The case sensitivity of this name depends on the case sensitivity of the BioRS server and on the value of the CASE_SENSITIVE server option. You cannot use the ALTER NICKNAME statement to change or delete this name. If the name of the BioRS databank that is used in this option changes, you must delete and then create the entire nickname again. For Documentum: Specifies the name of the Documentum object type that is associated with the nickname. The name can be any Documentum object type or registered table. The name of a registered table must be prefixed by the table owner's name. If the registered table belongs to the Docbase owner, the value dm_dbo can be used for the owner name. This option is required. Using the ALTER NICKNAME statement to change the value of the REMOTE_OBJECT option results in errors if the structure of the new object is not similar to that of the original object. For Entrez: Specifies the name of the Entrez object type that is associated with the nickname. This name determines the schema and NCBI database for the nickname and its relationship to other nicknames. This name is case insensitive.	
SOAPACTION	The URI SOAPACTION attribute from the Web Service Description Language (WSDL) format. This option is required for the root nickname. This option is not allowed with nonroot nicknames.	
SORTED	Specifies whether the data source file is sorted or unsorted. This option accepts either Y, y, n, or N. Sorted data sources must be sorted in ascending order according to the collation sequence for the current locale, as defined by the settings in the LC_COLLATE National Language Support category. If you specify that the data source is sorted, set the VALIDATE_DATA_FILE option to Y.	N
SORTFIELD	Specifies the name of a field on which search results should be sorted. The default value, DOC_RANK, is a field that Extended Search uses to determine the relevancy of a result document. If you specify a different field name, that name must exist in the sources that you search.	DOC_RANK

Table 29. Nickname options and their settings (continued)

Option	Description and valid settings	Default setting
SORTORDER	Specifies a sort order for the return of search results, either ascending (A) or descending (D).	A
STREAMING	Specifies whether the XML source document should be separated into logical fragments for processing. The fragments correspond to the node that matches the XPath expression of the nickname. The wrapper then parses and processes the XML source data fragment by fragment. This type of parsing minimizes memory usage. This option is specified on only the root nickname. You can specify streaming for any XML source document (FILE, DIRECTORY, URI, or COLUMN). This option is accepted only for columns of the root nickname (the nickname that identifies the elements at the top level of the XML document). Valid values are: Y The XML documents are parsed. N The XML documents are not parsed. Do not set the STREAMING parameter to YES if you set the VALIDATE parameter to YES. If you set both parameters to YES, you will receive an error message.	N
TEMPLATE	For WebSphere Business Integration: The nickname template fragment to use to construct an XML input document. The fragment must conform to the specified template syntax. For Web Services: The nickname template fragment to use to construct a SOAP request. The fragment must conform to the specified template syntax.	
TOTALMAXHIT	An INTEGER that specifies the maximum number of results that can be returned from all the sources that are being searched. The wrapper combines these results into a single result set.	50
TIMEOUT	For Extended Search: An INTEGER that specifies the number of seconds to wait for a response from a server before the request times out. For BLAST and HMMER: The maximum time, in minutes, that the wrapper waits for results from the daemon.	For Extended Search: 30. For BLAST and HMMER: 60.
URL	The URL for the Web service endpoint. This option is required for the root nickname. This option is not allowed with nonroot nicknames. Supported protocols are HTTP and HTTPS.	

Table 29. Nickname options and their settings (continued)

Option	Description and valid settings	Default setting
VALIDATE	Specifies whether the XML source document is validated before the XML data is extracted. If this option is set to YES, the nickname option verifies that the structure of the source document conforms to an XML schema or to a document type definition (DTD). This option is accepted only for columns of the root nickname (the nickname that identifies the elements at the top level of the XML document). The XML source document is not validated if the XML wrapper cannot locate the XML schema file or DTD file (.xsd or .dtd). DB2 UDB does not issue an error message if the validation does not occur. Therefore, ensure that the XML schema file or DTD file exists in the location that is specified in the XML source document. Do not set the VALIDATE parameter to YES if you set the STREAMING parameter to YES. If you set both parameters to YES, you will receive an error message.	NO
VALIDATE_DATA_FILE	For sorted files, this option specifies whether the wrapper verifies that the key column is sorted in ascending order and checks for null keys. The only valid values for this option are Y or N. The check is done once at registration time. This option is not allowed if the DOCUMENT nickname column option is used for the file path.	N
VERTICAL_TABLE	Specifies the presentation format for search results. If you specify YES, Extended Search returns all fields that are configured as returnable, in addition to the user-defined columns. The wrapper stores the results in the nickname table as a vertical list of column names.	NO
XPATH	Specifies the XPATH expression that identifies the elements that represent the individual tuples. The XPATH nickname option for a child nickname is evaluated in the context of the path that is specified by the XPATH nickname option of its parent. This XPATH expression is used as a context for evaluating column values that are identified by the XPATH nickname column options. For XML: Do not specify a namespace prefix in an XPATH expression. The XML wrapper does not support namespaces.	

Table 29. Nickname options and their settings (continued)

Option	Description and valid settings	Default setting
XPATH_EVAL_TIME	Specifies the time (in milliseconds) to evaluate the XPath expression of the nickname and to locate the first element. You can modify the XPATH_EVAL_TIME, INSTANCE_PARSE_TIME, and NEXT_TIME options to optimize queries of large or complex XML source structures. This option is accepted for root nicknames and nonroot nicknames. The number that you specify can be an integer or a decimal value.	1

Chapter 24. Nickname column options for federated systems

You can specify column information in the CREATE NICKNAME or ALTER NICKNAME statements using parameters called nickname column options.

The following table lists the nickname column options for each data source.

Table 30. Available nickname column options

Data source	ALL_VALUES	DEFAULT	DELIMITER	DOCUMENT	ESCAPE_INPUT	FOREIGN_KEY	INDEX	IS_REPEATING	NUMERIC_STRING	PRIMARY_KEY	REMOTE_NAME	SOAPACTIONCOLUMN	TEMPLATE	URLCOLUMN	VARCHAR_NO_TRAILING_BLANKS	XPATH
BLAST		X	X				X									
DB2 Universal Database for iSeries									X							
DB2 Universal Database for z/OS and OS/390									X							
DB2 Universal Database for VM and VSE									X							
DB2 Universal Database for Linux, UNIX, and Windows									X							
Documentum	X		X					X			X					
Informix									X							
Microsoft SQL Server									X							
ODBC									X							
OLE DB																
Oracle									X						X	
Sybase									X							
Table-structured files				X												
Teradata									X							
WebSphere Business Integration					X	X				X			X			X
Web services					X	X				X		X	X	X		X
XML				X		X				X						X

Table 31. Column options and their settings

Option	Description and valid settings	Default setting
ALL_VALUES	Specifies that all values of a repeating attribute will be returned, separated by the specified delimiter. If this option is missing or is N, then only the last value of a repeating attribute is returned. The ALL_VALUES option can be specified only for VARCHAR columns for which the IS_REPEATING option is 'Y' (and is invalid when IS_REG_TABLE = 'Y').	
DEFAULT	Specifies a new default value for the following input fixed columns: • E_value • QueryStrands • GapAlign • NMisMatchPenalty • NMatchReward • Matrix • FilterSequence • NumberOfAlignments • GapCost • ExtendedGapCost • WordSize • ThresholdEx This new value overrides the preset default values. The new default value must be of the same type as the indicated value for a given column.	
DELIMITER	For Documentum: Specifies the delimiter string to be used when multiple values of a repeating attribute are concatenated. The delimiter can be one or more characters. This option is valid only for attributes of objects with data type VARCHAR where the IS_REPEATING option is set to Y. For BLAST: The delimiter characters to be used to determine the end point of the definition line information for the column on which this option appears. If more than one character appears in this option's value, then the first occurrence of any one of the characters signals the end of this field's information. The default is end of line. This option is required, unless you want the last specified column to contain the remainder of the definition line.	For Documentum: The default delimiter is a comma. For BLAST: The default delimiter is end of line.

Table 31. Column options and their settings (continued)

Option	Description and valid settings	Default setting
DOCUMENT	For table-structured files: Specifies the kind of table-structured file. This wrapper supports only the value FILE for this option. Only one column can be specified with the DOCUMENT option per nickname. The column that is associated with the DOCUMENT option must be a data type of VARCHAR or CHAR. Using the DOCUMENT nickname column option instead of the FILE_PATH nickname option implies that the file that corresponds to this nickname will be supplied when the query runs. If the DOCUMENT option has the FILE value, the value that is supplied when the query runs is the full path of the file whose schema matches the nickname definition for this nickname. For XML: Specifies that this column is a DOCUMENT column. The value of the DOCUMENT column indicates the type of XML source data that is supplied to the nickname when the query runs. This option is accepted only for columns of the root nickname (the nickname that identifies the elements at the top level of the XML document). Only one column can be specified with the DOCUMENT option per nickname. The column that is associated with the DOCUMENT option must be a VARCHAR data type. If you use a DOCUMENT column option instead of the FILE_PATH or DIRECTORY_PATH nickname option the document that corresponds to this nickname is supplied when the query runs. The valid values for the DOCUMENT option are: **FILE** Specifies that the value of the nickname column is bound to the path name of a file. The data from this file is supplied when the query runs. **DIRECTORY** Specifies that the value of the nickname column is bound to the path name of a directory that contains multiple XML data files. The XML data from multiple files is supplied when the query runs. The data is in XML files in the specified directory path. The XML wrapper uses only the files with an .xml extension that are located in the directory that you specify. The XML wrapper ignores all other files in this directory. **URI** Specifies that the value of the nickname column is bound to the path name of a remote XML file to which a URI refers. The URI address indicates the remote location of this XML file on the Web. **COLUMN** Specifies that the XML document is stored in a relational column.	

Table 31. Column options and their settings (continued)

Option	Description and valid settings	Default setting
ELEMENT_NAME	Specifies the BioRS element name. The case sensitivity of this name depends on the case sensitivity of the BioRS server and on the value of the CASE_SENSITIVE server option. You must specify the BioRS element name only if it is different from the column name.	
ESCAPE_INPUT	Specifies whether XML special characters are replaced in XML input values or not. Use this option to include XML fragments as input, such as XML fragments with repeating elements. The TEMPLATE column option must be defined on columns that use the ESCAPE_INPUT column option. The column data type must be VARCHAR or CHAR. Valid values are: Y If the XML input contains special characters these are replaced with their counterpart characters that XML uses to represent the input characters. N Input characters are preserved exactly as they appear.	Y
FOREIGN_KEY	Indicates that this nickname is a child nickname and specifies the name of the corresponding parent nickname. A nickname can have at most one FOREIGN_KEY column option. The value for this option is case sensitive. The table that is designated with this option holds a key that is generated by the wrapper. The XPATH option must not be specified for this column. The column can be used only to join parent nicknames and child nicknames. A CREATE NICKNAME statement with a FOREIGN_KEY option will fail if the parent nickname has a different schema name. Unless the nickname that is referred to in a FOREIGN_KEY clause was explicitly defined as lowercase or mixed case by enclosing it in quotation marks in the corresponding CREATE NICKNAME statement, then when you refer to this nickname in the FOREIGN_KEY clause, you must specify the nickname in uppercase. When this option is set on a column, no other option can be set on the column.	
INDEX	The ordinal number of the column on which this option appears in the group of definition line columns. This option is required.	
IS_INDEXED	Indicates whether the corresponding column is indexed (whether the column can be referenced in a predicate). The valid values are Y and N. The value Y can be specified only for columns whose corresponding element is indexed by the BioRS server.	When a nickname is created, this option is automatically added with the value Y to any columns that correspond to a BioRS indexed element.

Table 31. Column options and their settings (continued)

Option	Description and valid settings	Default setting
IS_REPEATING	Indicates whether the column is multi-valued. Valid values are Y and N. Only the last value is returned for: • Non-VARCHAR repeating attributes • VARCHAR columns when ALL_VALUES 'N' is specified To overcome this limitation, you can create a dual definition for the repeating attribute column.	N
NUMERIC_STRING	Specifies whether a column contains strings of numeric characters. Y — This column contains strings of numeric characters '0', '1', '2', ... '9'. It does not contain blanks. If this column contains only numeric strings followed by trailing blanks, do not specify Y. When you set NUMERIC_STRING to Y for a column, you are informing the optimizer that this column contains no blanks that could interfere with sorting of the column's data. Use this option when the collating sequence of a data source is different from the collating sequence that the federated server uses. Columns that use this option are not excluded from remote evaluation because of a different collating sequence. N — This column is either not a numeric string column or is a numeric string column that contains blanks.	N
PRIMARY_KEY	Indicates that this nickname is a parent nickname. The column data type must be VARCHAR(16). A nickname can have at most one PRIMARY_KEY column option. YES is the only valid value. The column that is designated with this option holds a key that is generated by the wrapper. The XPATH option must not be specified for this column. The column can be used only to join parent nicknames and child nicknames. When this option is set on a column, no other option can be set on the column.	
REFERENCED_OBJECT	This option is valid only for columns whose BioRS data type is Reference. This option specifies the name of the BioRS databank that is referenced by the current column. The case sensitivity of this name depends on the case sensitivity of the BioRS server and on the value of the CASE_SENSITIVE server option.	
REMOTE_NAME	Specifies the name of the corresponding Documentum attribute or column. This option maps remote attribute or column names to local DB2 UDB column names.	The DB2 UDB column name.
SOAPACTIONCOLUMN	A column to dynamically specify the URI SOAPACTION attribute from the Web Service Description Language (WSDL) format. This option is specified on only the root nickname. When this option is set on a column, no other option can be set on the column.	

Table 31. Column options and their settings (continued)

Option	Description and valid settings	Default setting
TEMPLATE	The column template fragment to use to construct the XML input document. The fragment must conform to the specified template syntax.	
URLCOLUMN	A column to dynamically specify the URL for the Web service endpoint when you run a query. This option is specified on only the root nickname. When this option is set on a column, no other option can be set on the column.	
VARCHAR_NO_ TRAILING_BLANKS	This option applies to data sources that have variable character data types that do not pad the length with trailing blanks during comparison. Some data sources, such as Oracle, do not have blank-padded character comparison semantics that return the same results as the DB2 UDB for Linux, UNIX, and Windows comparison semantics. Set this option when you want it to apply only to a specific VARCHAR or VARCHAR2 column in a data source object. Y Trailing blanks are absent from these VARCHAR columns, or the data source has blank-padded character comparison semantics that are similar to the semantics on the federated server. The federated server sends character comparison operations to the data source for processing. N Trailing blanks are present in these VARCHAR columns, and the data source has blank-padded character comparison semantics that are different than the federated server. The federated server processes character comparison operations if it is not possible to compensate for equivalent semantics. For example, rewriting the predicate.	N for affected data sources
XPATH	Specifies the XPath expression in the XML document that contains the data that corresponds to this column. The wrapper evaluates the XPath expression after the CREATE NICKNAME statement applies this XPath expression from this XPATH nickname option.	

Related concepts:

- "Pushdown analysis" on page 133

Related tasks:

- "Global optimization" on page 144

Chapter 25. Function mapping options for federated systems

DB2 Information Integrator 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 32. Function mapping options and their settings

Option	Valid settings	Default setting
DISABLE	Disable a default function mapping. Valid values are 'Y' and 'N'.	'N'
INITIAL_INSTS	Estimated number of instructions processed the first and last time that the data source function is invoked.	'0'
INITIAL_IOS	Estimated number of I/Os performed the first and last time that the data source function is invoked.	'0'
IOS_PER_ARGBYTE	Estimated number of I/Os expended for each byte of the argument set that's passed to the data source function.	'0'
IOS_PER_INVOC	Estimated number of I/Os per invocation of a data source function.	'0'
INSTS_PER_ARGBYTE	Estimated number of instructions processed for each byte of the argument set that's passed to the data source function.	'0'
INSTS_PER_INVOC	Estimated number of instructions processed per invocation of the data source function.	'450'
PERCENT_ARGBYTES	Estimated average percent of input argument bytes that the data source function will actually read.	'100'
REMOTE_NAME	Name of the data source function.	local name

Chapter 26. Valid server types in SQL statements

Server types indicate what kind of data source that the server definition represents. Server types vary by vendor, purpose, and operating system. Supported values depend on the wrapper being used.

For most data sources, you must specify a valid server type in the CREATE SERVER statement.

BioRS wrapper

BioRS data sources.

Server Type	Data Source
Not required in the CREATE SERVER statement.	BioRS

BLAST wrapper

BLAST data sources supported by the BLAST daemon.

Server Type	Data Source
BLASTN	BLAST searches in which a nucleotide sequence is compared with the contents of a nucleotide sequence database to find sequences with regions homologous to regions of the original sequence.
BLASTP	BLAST searches in which an amino acid sequence is compared with the contents of an amino acid sequence database to find sequences with regions homologous to regions of the original sequence.
BLASTX	BLAST searches in which a nucleotide sequence is compared with the contents of an amino acid sequence database to find sequences with regions homologous to regions of the original sequence.
TBLASTN	BLAST searches in which an amino acid sequence is compared with the contents of a nucleotide sequence database to find sequences with regions homologous to regions of the original sequence.
TBLASTX	BLAST searches in which a nucleotide sequence is compared with the contents of a nucleotide sequence database to find sequences with regions homologous to regions of the original sequence.

CTLIB wrapper

Sybase data sources supported by the CTLIB client software.

Server Type	Data Source
SYBASE	Sybase

Documentum wrapper

Documentum data sources supported by the Documentum Client API/Library.

Server Type	Data Source
DCTM	Documentum

DRDA wrapper

DB2 Family data sources

Table 33. DB2 for Linux, UNIX, and Windows

Server Type	Data Source
DB2/UDB	IBM DB2 Universal Database
DB2/6000	IBM DB2 for AIX
DB2/AIX	IBM DB2 for AIX
DB2/HPUX	IBM DB2 for HP-UX
DB2/HP	IBM DB2 for HP-UX
DB2/NT	IBM DB2 for Windows NT
DB2/EEE	IBM DB2 Enterprise-Extended Edition
DB2/SUN	IBM DB2 for Solaris
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 34. DB2 for iSeries (and AS/400)

Server Type	Data Source
DB2/400	IBM DB2 for iSeries and AS/400

Table 35. 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 36. 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

Entrez wrapper

Entrez data sources.

Server Type	Data Source
NUCLEOTIDE	Entrez
PUBMED	Entrez

Excel wrapper

Excel data sources supported by Microsoft Excel 97, 2000, and 2002.

Server Type	Data Source
Not required in the CREATE SERVER statement.	Microsoft Excel

Extended Search wrapper

Extended Search data sources supported by the Extended Search Client Library.

Server Type	Data Source
Not required in the CREATE SERVER statement.	IBM Lotus Extended Search

HMMER wrapper

HMMER data sources supported by the HMMER daemon.

Server Type	Data Source
PFAM	HMMER
SEARCH	HMMER

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 the DataDirect Connect ODBC 3.6 driver or the ODBC 3.0 (or later) driver

Server Type	Data Source
MSSQLSERVER	Microsoft SQL Server

NET8 wrapper

Oracle data sources supported by Oracle NET8 client software.

Server Type	Data Source
ORACLE	Oracle Version 8.0. or later

ODBC wrapper

ODBC data sources supported by the ODBC 3.x driver.

Server Type	Data Source
ODBC	ODBC

OLE DB wrapper

OLE DB providers compliant with Microsoft OLE DB 2.0 or later.

Server Type	Data Source
Not required in the CREATE SERVER statement.	Any OLE DB provider

Table-structured files wrapper

Table-structured file data sources.

Server Type	Data Source
Not required in the CREATE SERVER statement.	Table-structured files

Teradata wrapper

Teradata data sources supported by the Teradata V2R3, V2R4, and V2R5 client software.

Server Type	Data Source
TERADATA	Teradata

Web services wrapper

Web services data sources.

Server Type	Data Source
Not required in the CREATE SERVER statement.	Any Web services data source.

WebSphere Business Integration wrapper

Business application data sources supported by the WeSphere Business Integration wrapper.

Server Type	Data Source
WBI	WebSphere Business Integration 2.2 or 2.3

XML wrapper

XML data sources.

Server Type	Data Source
Not required in the CREATE SERVER statement.	XML

Chapter 27. Default forward data type mappings

The two kinds of mappings between data source data types and federated database data types are forward type mappings and reverse type mappings. In a *forward type mapping*, the mapping is from a remote type to a comparable local type.

You can override a default type mapping, or create a new type mapping with the CREATE TYPE MAPPING statement.

These mappings are valid with all the supported versions, unless otherwise noted.

For all default forward data types mapping from a data source to DB2 for Linux, UNIX, and Windows, the DB2 federated schema is SYSIBM.

The following tables show the default forward mappings between DB2 for Linux, UNIX, and Windows data types and data source data types.

DB2 for z/OS and OS/390 data sources

Table 37. DB2 for z/OS and OS/390 forward default data type mappings (Not all columns shown)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
ROWID	-	-	-	-	Y	-	VARCHAR	40	-	Y
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-

Table 37. DB2 for z/OS and OS/390 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
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

DB2 for iSeries data sources

Table 38. DB2 for iSeries forward default data type mappings (Not all columns shown)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N

Table 38. DB2 for iSeries 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
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
NUMERIC	-	-	-	-	-	-	DECIMAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

DB2 Server for VM and VSE data sources

Table 39. DB2 Server for VM and VSE forward default data type mappings (Not all columns shown)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBAHW	-	-	-	-	-	-	SMALLINT	-	0	-
DBAINT	-	-	-	-	-	-	INTEGER	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-

Table 39. DB2 Server for VM and VSE 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
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPH	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

DB2 for Linux, UNIX, and Windows data sources

Table 40. DB2 for Linux, UNIX, and Windows forward default data type mappings (Not all columns shown)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BIGINT	-	-	-	-	-	-	BIGINT	-	0	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	-	-	-	-	-	-	CHAR	-	0	N
CHAR	-	-	-	-	Y	-	CHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-

Table 40. DB2 for Linux, UNIX, and Windows forward default data type mappings (Not all columns shown) (continued)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	-	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
LONGVAR	-	-	-	-	N	-	CLOB	-	-	-
LONGVAR	-	-	-	-	Y	-	BLOB	-	-	-
LONGVARG	-	-	-	-	-	-	DBCLOB	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	0	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	0	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	0	N

Informix data sources

Table 41. Informix forward default data type mappings (Not all columns shown)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	2147483647	-	-
BOOLEAN	-	-	-	-	-	-	CHARACTER	1	-	-
BYTE	-	-	-	-	-	-	BLOB	2147483647	-	-
CHAR	1	254	-	-	-	-	CHARACTER	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CLOB	-	-	-	-	-	-	CLOB	2147483647	-	-
DATE	-	-	-	-	-	-	DATE	4	-	-
DATETIME	0	4	0	4	-	-	DATE	4	-	-
DATETIME	6	10	6	10	-	-	TIME	3	-	-
DATETIME	0	4	6	15	-	-	TIMESTAMP	10	-	-
DATETIME	6	10	11	15	-	-	TIMESTAMP	10	-	-
DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
DECIMAL	32	130	-	-	-	-	DOUBLE	8	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
INTERVAL	-	-	-	-	-	-	VARCHAR	25	-	-
INT8	-	-	-	-	-	-	BIGINT	19	0	-
LVARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
MONEY	1	31	0	31	-	-	DECIMAL	-	-	-
MONEY	32	32	-	-	-	-	DOUBLE	8	-	-
NCHAR	1	254	-	-	-	-	CHARACTER	-	-	-
NCHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
NVARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
REAL	-	-	-	-	-	-	REAL	4	-	-
SERIAL	-	-	-	-	-	-	INTEGER	4	-	-
SERIAL8	-	-	-	-	-	-	BIGINT	-	-	-
SMALLFLOAT	-	-	-	-	-	-	REAL	4	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
TEXT	-	-	-	-	-	-	CLOB	2147483647	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-

Table 41. 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

Notes:

- For the Informix DATETIME data type, the DB2 UNIX and Windows federated server uses the Informix high-level qualifer as the REMOTE_LENGTH and the Informix low-level qualifier as the REMOTE_SCALE.

  The Informix qualifiers are the "TU_" constants defined in the Informix Client SDK `datatime.h` file. The contstants are:

0 = YEAR	8 = MINUTE	13 = FRACTION(3)
2 = MONTH	10 = SECOND	14 = FRACTION(4)
4 = DAY	11 = FRACTION(1)	15 = FRACTION(5)
6 = HOUR	12 = FRACTION(2)	

Microsoft SQL Server data sources

Table 42. Microsoft SQL Server forward default data type mappings

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
bigint [4]	-	-	-	-	-	-	BIGINT	-	-	-
binary	1	254	-	-	-	-	CHARACTER	-	-	Y
binary	255	8000	-	-	-	-	VARCHAR	-	-	Y
bit	-	-	-	-	-	-	SMALLINT	2	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	8000	-	-	-	-	VARCHAR	-	-	N
datetime	-	-	-	-	-	-	TIMESTAMP	10	-	-
datetimen	-	-	-	-	-	-	TIMESTAMP	10	-	-
decimal	1	31	0	31	-	-	DECIMAL	-	-	-
decimal	32	38	0	38	-	-	DOUBLE	-	-	-
decimaln	1	31	0	31	-	-	DECIMAL	-	-	-

Table 42. Microsoft SQL Server forward default data type mappings (continued)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
decimaln	32	38	0	38	-	-	DOUBLE	-	-	-
DUMMY65 [1]	1	38	-84	127	-	-	DOUBLE	-	-	-
DUMMY2000 [3]	1	38	-84	127	-	-	DOUBLE	-	-	-
float	-	8	-	-	-	-	DOUBLE	8	-	-
floatn	-	8	-	-	-	-	DOUBLE	8	-	-
float	-	4	-	-	-	-	REAL	4	-	-
floatn	-	4	-	-	-	-	REAL	4	-	-
image	-	-	-	-	-	-	BLOB	2147483647	-	Y
int	-	-	-	-	-	-	INTEGER	4	-	-
intn	-	-	-	-	-	-	INTEGER	4	-	-
money	-	-	-	-	-	-	DECIMAL	19	4	-
moneyn	-	-	-	-	-	-	DECIMAL	19	4	-
nchar	1	127	-	-	-	-	CHAR	-	-	N
nchar	128	4000	-	-	-	-	VARCHAR	-	-	N
numeric	1	31	0	31	-	-	DECIMAL	-	-	-
numeric	32	38	0	38	-	-	DOUBLE	8	-	-
numericn	32	38	0	38	-	-	DOUBLE	-	-	-
numericn	1	31	0	31	-	-	DECIMAL	-	-	-
ntext [2]	-	-	-	-	-	-	CLOB	2147483647	-	Y
nvarchar	1	4000	-	-	-	-	VARCHAR	-	-	N
real	-	-	-	-	-	-	REAL	4	-	-
smallint	-	-	-	-	-	-	SMALLINT	2	-	-
smalldatetime	-	-	-	-	-	-	TIMESTAMP	10	-	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
smallmoneyn	-	-	-	-	-	-	DECIMAL	10	4	-
SQL_BIGINT	-	-	-	-	-	-	DECIMAL	-	-	-
SQL_BIGINT [4]	-	-	-	-	-	-	BIGINT	-	-	-
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	8000	-	-	-	-	VARCHAR	-	-	N

Table 42. Microsoft SQL Server forward default data type mappings (continued)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
SQL_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_DECIMAL	32	32	0	31	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_GUID [2]	1	4000	-	-	Y	-	VARCHAR	16	-	Y
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N
SQL_LONGVARBINARY	-	-	-	-	-	-	BLOB	-	-	Y
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_REAL	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	-	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_VARBINARY	1	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	8000	-	-	-	-	VARCHAR	-	-	N
text	-	-	-	-	-	-	CLOB	-	-	N
timestamp	-	-	-	-	-	-	VARCHAR	8		Y
tinyint	-	-	-	-	-	-	SMALLINT	2	-	-
uniqueidentifier [2]	1	4000	-	-	Y	-	VARCHAR	16	-	Y
varbinary	1	8000	-	-	-	-	VARCHAR	-	-	Y
varchar	1	8000	-	-	-	-	VARCHAR	-	-	N

Notes:

1. This type mapping is valid only with Microsoft SQL Server Version 6.5.
2. This type mapping is valid only with Microsoft SQL Server Version 7 and Version 2000.
3. This type mapping is valid only with Windows 2000 operating systems.
4. This type mapping is valid only with Microsoft SQL Server Version 2000.

ODBC data sources

Table 43. ODBC forward default data type mappings (Not all columns shown)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
SQL_BIGINT	-	-	-	-	-	-	BIGINT	8	-	-
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	32672	-	-	-	-	VARCHAR	-	-	N
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N
SQL_LONGVARBINARY	-	-	-	-	-	-	BLOB	-	-	Y
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_NUMERIC	32	32	0	31	-	-	DOUBLE	8	-	-
SQL_REAL	-	-	-	-	-	-	REAL	4	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TYPE_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_TYPE_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TYPE_TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	-	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_VARBINARY	1	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	N
SQL_WCHAR	1	127	-	-	-	-	CHAR	-	-	N
SQL_WCHAR	128	16336	-	-	-	-	VARCHAR	-	-	N
SQL_WVARCHAR	1	16336	-	-	-	-	VARCHAR	-	-	N
SQL_WLONGVARCHAR	-	1073741823	-	-	-	-	CLOB	2147483647	-	N

Oracle NET8 data sources

Table 44. Oracle NET8 forward default data type mappings

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BLOB	0	0	0	0	-	\0	BLOB	2147483647	0	Y
CHAR	1	254	0	0	-	\0	CHAR	0	0	N
CHAR	255	2000	0	0	-	\0	VARCHAR	0	0	N
CLOB	0	0	0	0	-	\0	CLOB	2147483647	0	N
DATE	0	0	0	0	-	\0	TIMESTAMP	0	0	N
FLOAT	1	126	0	0	-	\0	DOUBLE	0	0	N
LONG	0	0	0	0	-	\0	CLOB	2147483647	0	N
LONG RAW	0	0	0	0	-	\0	BLOB	2147483647	0	Y
MLSLABEL	0	0	0	0	-	\0	VARCHAR	255	0	N
NUMBER	1	38	-84	127	-	\0	DOUBLE	0	0	N
NUMBER	1	31	0	31	-	>=	DECIMAL	0	0	N
NUMBER	1	4	0	0	-	\0	SMALLINT	0	0	N
NUMBER	5	9	0	0	-	\0	INTEGER	0	0	N
NUMBER	-	10	0	0	-	\0	DECIMAL	0	0	N
RAW	1	2000	0	0	-	\0	VARCHAR	0	0	Y
ROWID	0	0	0	NULL	-	\0	CHAR	18	0	N
TIMESTAMP [1]	-	-	-	-	-	-	TIMESTAMP	10	-	-
VARCHAR2	1	4000	0	0	-	\0	VARCHAR	0	0	N

Notes:

1. This type mapping is valid only for Oracle 9i (or later) client and server configurations.

Sybase data sources

Table 45. Sybase CTLIB forward default data type mappings

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
binary	1	254	-	-	-	-	CHAR	-	-	Y
binary	255	16384	-	-	-	-	VARCHAR	-	-	Y
bit	-	-	-	-	-	-	SMALLINT	-	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	16384	-	-	-	-	VARCHAR	-	-	N
char null (see varchar)										
datetime	-	-	-	-	-	-	TIMESTAMP	-	-	-
datetimn	-	-	-	-	-	-	TIMESTAMP	-	-	-
decimal	1	31	0	31	-	-	DECIMAL	-	-	-
decimal	32	38	0	38	-	-	DOUBLE	-	-	-
decimaln	1	31	0	31	-	-	DECIMAL	-	-	-
decimaln	32	38	0	38	-	-	DOUBLE	-	-	-
float	-	4	-	-	-	-	REAL	-	-	-
float	-	8	-	-	-	-	DOUBLE	-	-	-
floatn	-	4	-	-	-	-	REAL	-	-	-
floatn	-	8	-	-	-	-	DOUBLE	-	-	-
image	-	-	-	-	-	-	BLOB	-	-	-
int	-	-	-	-	-	-	INTEGER	-	-	-
intn	-	-	-	-	-	-	INTEGER	-	-	-
money	-	-	-	-	-	-	DECIMAL	19	4	-
moneyn	-	-	-	-	-	-	DECIMAL	19	4	-
nchar	1	254	-	-	-	-	CHAR	-	-	N
nchar	255	16384	-	-	-	-	VARCHAR	-	-	N
nchar null (see nvarchar)										
numeric	1	31	0	31	-	-	DECIMAL	-	-	-
numeric	32	38	0	38	-	-	DOUBLE	-	-	-
numericn	1	31	0	31	-	-	DECIMAL	-	-	-
numericn	32	38	0	38	-	-	DOUBLE	-	-	-
nvarchar	1	16384	-	-	-	-	VARCHAR	-	-	N
real	-	-	-	-	-	-	REAL	-	-	-

Table 45. Sybase CTLIB forward default data type mappings (continued)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
smalldatetime	-	-	-	-	-	-	TIMESTAMP	-	-	-
smallint	-	-	-	-	-	-	SMALLINT	-	-	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
sysname	1	254	-	-	-	-	CHAR	-	-	N
text	-	-	-	-	-	-	CLOB	-	-	-
timestamp	-	-	-	-	-	-	VARCHAR	8	-	Y
tinyint	-	-	-	-	-	-	SMALLINT	-	-	-
unichar[1]	1	254	-	-	-	-	CHAR	-	-	N
unichar[1]	255	16384	-	-	-	-	VARCHAR	-	-	N
unichar null (see univarchar)										
univarchar[1]	1	16384	-	-	-	-	VARCHAR	-	-	N
varbinary	1	16384	-	-	-	-	VARCHAR	-	-	Y
varchar	1	16384	-	-	-	-	VARCHAR	-	-	N

Notes:

1. Valid for non-Unicode federated databases.

Teradata data sources

Table 46. Teradata forward default data type mappings (Not all columns shown)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BYTE	1	254	-	-	-	-	CHAR	-	-	Y
BYTE	255	32672	-	-	-	-	VARCHAR	-	-	Y
BYTE	32673	64000	-	-	-	-	BLOB	-	-	-

Table 46. Teradata forward default data type mappings (Not all columns shown) (continued)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BYTEINT	-	-	-	-	-	-	SMALLINT	-	-	-
CHAR	1	254	-	-	-	-	CHARACTER	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CHAR	32673	64000	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-
DECIMAL	1	18	0	18	-	-	DECIMAL	-	-	-
DOUBLE PRECISION	-	-	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	-	-
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	-	-
GRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
INTERVAL	-	-	-	-	-	-	CHAR	-	-	-
NUMERIC	1	18	0	18	-	-	DECIMAL	-	-	-
REAL	-	-	-	-	-	-	DOUBLE	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	-	-
VARBYTE	1	32762	-	-	-	-	VARCHAR	-	-	Y
VARBYTE	32763	64000	-	-	-	-	BLOB	-	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
VARCHAR	32673	64000	-	-	-	-	CLOB	-	-	-
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	-	-
VARGRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-

Related concepts:

- "Forward and reverse data type mappings" on page 48

Related reference:

- "Altering long data types to varchar data types" on page 55
- "Unicode default forward data type mappings - NET8 wrapper" on page 287
- "Unicode default forward data type mappings - Sybase wrapper" on page 288
- "Unicode default forward data type mappings - ODBC wrapper" on page 289

- "Unicode default forward data type mappings - Microsoft SQL Server wrapper" on page 290

Chapter 28. Default reverse data type mappings

The two kinds of mappings between data source data types and federated database data types are forward type mappings and reverse type mappings. In a *forward type mapping*, the mapping is from a remote type to a comparable local type. The other type of mapping is a *reverse type mapping*, which is used with transparent DDL to create or modify remote tables.

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

When you define a remote table or view to the DB2 federated database, the definition includes a reverse type mapping. The mapping is from a *local* DB2 for Linux, UNIX, and Windows data type for each column, and the corresponding *remote* data type. For example, there is a default reverse type mapping in which the local type REAL points to the Informix type SMALLFLOAT.

DB2 for Linux, UNIX, and Windows federated servers do not support mappings for LONG VARCHAR, LONG VARGRAPHIC, DATALINK, and user-defined types.

When you use the CREATE TABLE statement to create a remote table, you specify the local data types you want to include in the remote table. These default reverse type mappings will assign corresponding remote types to these columns. For example, suppose that you use the CREATE TABLE statement to define an Informix table with a column C2. You specify BIGINT as the data type for C2 in the statement. The default reverse type mapping of BIGINT depends on which version of Informix you are creating the table on. The mapping for C2 in the Informix table will be to DECIMAL in Informix Version 8 and to INT8 in Informix Version 9.

You can override a default reverse type mapping, or create a new reverse type mapping with the CREATE TYPE MAPPING statement.

The following tables show the default reverse mappings between DB2 for Linux, UNIX, and Windows local data types and remote data source data types.

These mappings are valid with all the supported versions, unless otherwise noted.

DB2 for z/OS and OS/390 data sources

Table 47. DB2 for z/OS and OS/390 reverse default data type mappings (Not all columns shown)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	–
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	N

DB2 for iSeries data sources

Table 48. DB2 for iSeries reverse default data type mappings (Not all columns shown)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	N
CHARACTER	-	-	-	-	Y	-	CHARACTER	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	NUMERIC	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	FLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	VARG	-	-	N

DB2 for VM and VSE data sources

Table 49. DB2 for VM and VSE reverse default data type mappings (Not all columns shown)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	-
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPH	-	-	N

DB2 for Linux, UNIX, and Windows data sources

Table 50. DB2 for Linux, UNIX, and Windows reverse default data type mappings (Not all columns shown)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	FEDERATED_BIT_DATA
BIGINT	-	8	-	-	-	-	BIGINT	-	-	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	-	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

Informix data sources

Table 51. Informix reverse default data type mappings

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BIGINT [1]	-	-	-	-	-	-	DECIMAL	19	-	-
BIGINT [2]	-	-	-	-	-	-	INT8	-	-	-
BLOB	1	2147483647	-	-	-	-	BYTE	-	-	-
CHARACTER	-	-	-	-	N	-	CHAR	-	-	-
CHARACTER	-	-	-	-	Y	-	BYTE	-	-	-
CLOB	1	2147483647	-	-	-	-	TEXT	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	SMALLFLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	DATETIME	6	10	-
TIMESTAMP	-	10	-	-	-	-	DATETIME	0	15	-
VARCHAR	1	254	-	-	N	-	VARCHAR	-	-	-
VARCHAR	255	32672	-	-	N	-	TEXT	-	-	-
VARCHAR	-	-	-	-	Y	-	BYTE	-	-	-
VARCHAR [2]	255	2048	-	-	N	-	LVARCHAR	-	-	-
VARCHAR [2]	2049	32672	-	-	N	-	TEXT	-	-	-

Notes:

1. This type mapping is valid only with Informix server Version 8 (or lower).
2. This type mapping is valid only with Informix server Version 9.

For the Informix DATETIME data type, the DB2 UNIX and Windows federated server uses the Informix high-level qualifer as the REMOTE_LENGTH and the Informix low-level qualifier as the REMOTE_SCALE.

The Informix qualifiers are the ″TU_″ constants defined in the Informix Client SDK `datatime.h` file. The contstants are:

0 = YEAR	8 = MINUTE	13 = FRACTION(3)
2 = MONTH	10 = SECOND	14 = FRACTION(4)
4 = DAY	11 = FRACTION(1)	15 = FRACTION(5)
6 = HOUR	12 = FRACTION(2)	

Microsoft SQL Server data sources

Table 52. Microsoft SQL Server reverse default data type mappings (Not all columns shown)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BIGINT [1]	-	-	-	-	-	-	bigint	-	-	-
BLOB	-	-	-	-	-	-	image	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
DATE	-	4	-	-	-	-	datetime	-	-	-
DECIMAL	-	-	-	-	-	-	decimal	-	-	-
DOUBLE	-	8	-	-	-	-	float	-	-	-
INTEGER	-	-	-	-	-	-	int	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
REAL	-	4	-	-	-	-	real	-	-	-
TIME	-	3	-	-	-	-	datetime	-	-	-
TIMESTAMP	-	10	-	-	-	-	datetime	-	-	-
VARCHAR	1	8000	-	-	N	-	varchar	-	-	-
VARCHAR	8001	32672	-	-	N	-	text	-	-	-
VARCHAR	1	8000	-	-	Y	-	varbinary	-	-	-
VARCHAR	8001	32672	-	-	Y	-	image	-	-	-

Notes:

1. This type mapping is valid only with Microsoft SQL Server Version 2000.

Oracle NET8 data sources

Table 53. Oracle NET8 reverse default data type mappings

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BLOB	0	2147483647	0	0	Y	\0	BLOB	0	0	Y
CHARACTER	1	254	0	0	N	\0	CHAR	0	0	N
CHARACTER	1	254	0	0	Y	\0	RAW	0	0	Y
CLOB	0	2147483647	0	0	N	\0	CLOB	0	0	N
DATE	0	4	0	0	N	\0	DATE	0	0	N
DECIMAL	0	0	0	0	N	\0	NUMBER	0	0	N
DOUBLE	0	8	0	0	N	\0	FLOAT	126	0	N
FLOAT	0	8	0	0	N	\0	FLOAT	126	0	N
INTEGER	0	4	0	0	N	\0	NUMBER	9	0	N
REAL	0	4	0	0	N	\0	FLOAT	63	0	N
SMALLINT	0	2	0	0	N	\0	NUMBER	4	0	N
TIME	0	3	0	0	N	\0	DATE	0	0	N
TIMESTAMP	0	10	0	0	N	\0	DATE	0	0	N
VARCHAR	1	4000	0	0	N	\0	VARCHAR2	0	0	N
VARCHAR	1	2000	0	0	Y	\0	RAW	0	0	Y

Note: The DB2 Universal Database for Linux, UNIX, and Windows BIGINT data type is not available for transparent DDL. You cannot specify the BIGINT data type in a CREATE TABLE statement when you create a remote Oracle table.

Sybase data sources

Table 54. Sybase CTLIB default reverse data type mappings

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BIGINT	-	-	-	-	-	-	decimal	19	0	-
BLOB	-	-	-	-	-	-	image	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
DATE	-	-	-	-	-	-	datetime	-	-	-
DECIMAL	-	-	-	-	-	-	decimal	-	-	-
DOUBLE	-	-	-	-	-	-	float	-	-	-
GRAPHIC	-	-	-	-	-	-	unichar	-	-	-
VARGRAPHIC	-	-	-	-	-	-	univarchar	-	-	-
INTEGER	-	-	-	-	-	-	integer	-	-	-
REAL	-	-	-	-	-	-	real	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
TIME	-	-	-	-	-	-	datetime	-	-	-
TIMESTAMP	-	-	-	-	-	-	datetime	-	-	-
VARCHAR[1]	1	255	-	-	N	-	varchar	-	-	-
VARCHAR[1]	256	32672	-	-	N	-	text	-	-	-
VARCHAR [2]	1	16384	-	-	N	-	varchar	-	-	-
VARCHAR [2]	16385	32672	-	-	N	-	text	-	-	-
VARCHAR[1]	1	255	-	-	Y	-	varbinary	-	-	-
VARCHAR[1]	256	32672	-	-	Y	-	image	-	-	-
VARCHAR [2]	1	16384	-	-	Y	-	varbinary	-	-	-
VARCHAR [2]	16385	32672	-	-	Y	-	image	-	-	-

Notes:

1. This type mapping is valid only for CTLIB with Sybase server version 12.0 (or earlier).
2. This type mapping is valid only for CTLIB with Sybase server version 12.5 (or later).

Teradata data sources

Table 55. Teradata reverse default data type mappings (Not all columns shown)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	FEDERATED_BIT_DATA
BLOB [1]	1	64000	-	-	-	-	VARBYTE	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	-
CHARACTER	-	-	-	-	Y	-	BYTE	-	-	-
CLOB [2]	1	64000	-	-		-	VARCHAR	-	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-
DBCLOB [3]	1	32000	-	-	-	-	VARGRAPHIC	-	-	-
DECIMAL	1	18	0	18	-	-	DECIMAL	-	-	-
DECIMAL	19	31	0	31	-	-	FLOAT	-	-	-
DOUBLE	-	-	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	FLOAT	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARBYTE	-	-	-
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

Notes:

1. The Teradata VARBYTE data type can contain only the specified length (1 to 64000) of a DB2 BLOB data type.
2. The Teradata VARCHAR data type can contain only the specified length (1 to 64000) of a DB2 CLOB data type.
3. The Teradata VARGRAPHIC data type can contain only the specified length (1 to 32000) of a DB2 DBCLOB data type.

Related concepts:

• "Forward and reverse data type mappings" on page 48

Chapter 29. Unicode default data type mappings

Unicode default forward data type mappings - NET8 wrapper

The following table lists the default forward data type mapping for the NET8 wrapper when the federated database is a Unicode database.

Table 56. Unicode default forward data type mappings for the NET8 wrapper

UTF-8	Oracle	
Data type	**Data type**	**Length**
CHAR	CHAR	1 to 254 bytes
VARCHAR	CHAR	255 to 2000 bytes
	VARCHAR2	1 to 4000 bytes
DBCLOB	NCLOB	
GRAPHIC	NCHAR	1 to 127 characters
VARGRAPHIC	NCHAR	128 to 1000 characters
	NVARCHAR2	1 to 2000 characters

Related concepts:

- "Unicode support for federated systems" on page 123

Unicode default reverse data type mappings - NET8 wrapper

The following table lists the default reverse data type mapping for the NET8 wrapper when the federated database is a Unicode database.

Table 57. Unicode default reverse data type mappings for the NET8 wrapper

UTF-8		Oracle
Data type	**Length**	**Data type**
CHAR	1 to 254 bytes	CHAR
VARCHAR	1 to 4000 bytes	VARCHAR2
CLOB	1 to 2 147 483 647 bytes	CLOB
GRAPHIC	1 to 127 characters	NCHAR
VARGRAPHIC	1 to 2000 characters	NVARCHAR2
DBCLOB	1 to 1 073 741 823 characters	NCLOB

Related concepts:

- "Unicode support for federated systems" on page 123

Unicode default forward data type mappings - Sybase wrapper

The following table lists the default forward data type mapping for the CTLIB wrapper when the federated database is a Unicode database.

Table 58. Unicode default forward data type mappings for the Sybase CTLIB wrapper

UTF-8	Sybase	
Data type	**Data type**	**Length**
CHAR	char	1 to 254 bytes
	nchar	1 to 127 characters
VARCHAR	char	255 to 32672 bytes
	varchar	1 to 32672 bytes
	nchar	128 to 16336 characters
	nvarchar	1 to 16336 characters
CLOB	text	
GRAPHIC	unichar	1 to 127 characters
VARGRAPHIC	unichar	128 to 16336 characters
	univarchar	1 to 16336 characters

Related concepts:

- "Unicode support for federated systems" on page 123

Unicode default reverse data type mappings - Sybase wrapper

The following table lists the default reverse data type mapping for the CTLIB wrapper when the federated database is a Unicode database.

Table 59. Unicode default reverse data type mappings for the Sybase CTLIB wrapper

UTF-8		Sybase
Data type	**Length**	**Data type**
CHAR	1 to 254 bytes	char
VARCHAR	1 to 32672 bytes	varchar
CLOB	1 to 2 147 483 647 bytes	text
GRAPHIC	1 to 127 characters	unichar
VARGRAPHIC	1 to 16336 characters	univarchar

Related concepts:

- "Unicode support for federated systems" on page 123

Unicode default forward data type mappings - ODBC wrapper

The following table lists the default forward data type mapping for the ODBC wrapper when the federated database is a Unicode database.

Table 60. Unicode default forward data type mappings for the ODBC wrapper

UTF-8	ODBC	
Data type	**Data type**	**Length**
CHAR	SQL_CHAR	1 to 254 bytes
VARCHAR	SQL_CHAR	255 to 32672 bytes
	SQL_VARCHAR	1 to 32672 bytes
CLOB	SQL_LONGVARCHAR	-
GRAPHIC	SQL_WCHAR	1 to 127 characters
VARGRAPHIC	SQL_WVARCHAR	128 to 16336 characters
	SQL_WVARCHAR	1 to 16336 characters
DBCLOB	SQL_WLONGVARCHAR	-

Related concepts:

- "Unicode support for federated systems" on page 123

Unicode default reverse data type mappings - ODBC wrapper

The following table lists the default reverse data type mapping for the ODBC wrapper when the federated database is a Unicode database.

Table 61. Unicode default reverse data type mappings for the ODBC wrapper

UTF-8		ODBC
Data type	**Length**	**Data type**
CHAR	1 to 254 bytes	SQL_CHAR
VARCHAR	1 to 32672 bytes	SQL_VARCHAR
CLOB	1 to 2 147 483 647 bytes	SQL_LONGVARCHAR
GRAPHIC	1 to 127 characters	SQL_WCHAR
VARGRAPHIC	1 to 16336 characters	SQL_WVARCHAR
DBCLOB	1 to 1 073 741 823 characters	SQL_WLONGVARCHAR

Related concepts:

- "Unicode support for federated systems" on page 123

Unicode default forward data type mappings - Microsoft SQL Server wrapper

The following table lists the default forward data type mapping for the Microsoft SQL Server wrapper when the federated database is a Unicode database.

Table 62. Unicode default forward data type mappings for the Microsoft SQL Server wrapper

UTF-8	Microsoft SQL Server	
Data type	**Data type**	**Length**
CHAR	CHAR	1 to 254 bytes
VARCHAR	CHAR	255 to 8000 bytes
	VARCHAR	1 to 8000 bytes
CLOB	TEXT	-
GRAPHIC	NCHAR	1 to 127 characters
VARGRAPHIC	NCHAR	128 to 16336 characters
	NVARCHAR	1 to 16336 characters
DBCLOB	NTEXT	-

Related concepts:

- "Unicode support for federated systems" on page 123

Unicode default reverse data type mappings - Microsoft SQL Server wrapper

The following table lists the default reverse data type mapping for the Microsoft SQL Server wrapper when the federated database is a Unicode database.

Table 63. Unicode default reverse data type mappings for the Microsoft SQL Server wrapper

UTF-8		Microsoft SQL Server
Data type	**Length**	**Data type**
CHAR	1 to 254 bytes	CHAR
VARCHAR	1 to 32672 bytes	VARCHAR
CLOB	1 to 2 147 483 647 bytes	TEXT
GRAPHIC	1 to 127 characters	NCHAR
VARGRAPHIC	1 to 16336 characters	NVARCHAR
DBCLOB	1 to 1 073 741 823 characters	NTEXT

Related concepts:

- "Unicode support for federated systems" on page 123

Chapter 30. Data types supported for nonrelational data sources

For most of the nonrelational data sources, you must specify the column information, including data type, when you create the nicknames to access the data source.

Some of the nonrelational wrappers create all of the columns required to access a data source. These are called *fixed columns*. Other wrappers let you specify some or all of the data types for the columns in the CREATE NICKNAME statement.

The following sections list the wrappers that you can specify the data types for, and the data types that are supported by the wrapper.

Data types supported by the BioRS wrapper

The following table lists the DB2 data types that are supported by the BioRS wrapper.

Table 64. BioRS data types that map to DB2 data types

BioRS data types	DB2 data type
AUTHOR	CHARACTER, CLOB, VARCHAR
DATE	CHARACTER, CLOB, VARCHAR
NUMBER	CHARACTER, CLOB, VARCHAR
REFERENCE	CHARACTER, CLOB, VARCHAR
TEXT	CHARACTER, CLOB, VARCHAR

The maximum length allowed for the CLOB data type is 5 megabytes.

Data types supported by the BLAST wrapper

Some of the data types are automatically set for the fixed columns that the BLAST wrapper creates.

For the definition line fields, you can assign when you create a nickname. If the data in the definition line column is not compatible with the local column data type, you will get an error. For example, if you define a definition line column of type INTEGER and there are values in the column that are not numeric, an error is returned.

The following table lists the DB2 data types that are supported by the BLAST wrapper.

Table 65. BLAST data types that map to DB2 data types

BLAST data types	DB2 data type
definition line	CLOB The maximum length allowed for the CLOB data type is 5 megabytes.
definition line	DOUBLE

Table 65. BLAST data types that map to DB2 data types (continued)

BLAST data types	DB2 data type
definition line	FLOAT
definition line	INTEGER
definition line	VARCHAR

Data types supported by the Documentum wrapper

The following table lists the DB2 data types that are supported by the Documentum wrapper.

Table 66. Documentum data types that map to DB2 data types

Documentum data types	DB2 data type
DOUBLE	DOUBLE, FLOAT, INTEGER, SMALLINT
ID	CHARACTER (16)
INTEGER	DOUBLE, FLOAT, INTEGER, SMALLINT
STRING (up to 255 characters)	CHAR, VARCHAR
TIME	CHAR, DATE, TIMESTAMP, VARCHAR

Data types supported by the Entrez wrapper

The following table lists the DB2 data types that are supported by the Entrez wrapper.

Table 67. Entrez data types that map to DB2 data types

Entrez data types	DB2 data type
character	CHARACTER
character	CLOB The maximum length allowed for the CLOB data type is 5 megabytes.
date	DATE
number	DECIMAL
number	DOUBLE
integer	INTEGER
number	REAL
integer	SMALLINT
time	TIMESTAMP
character	VARCHAR

Data types supported by the Excel wrapper

The following table lists the DB2 data types that are supported by the Excel wrapper.

Table 68. Excel data types that map to DB2 data types

Excel data types	DB2 data type
date	DATE
number	DOUBLE
number	FLOAT (n) where n is >= 25 and ≤= 53
integer	INTEGER
character	VARCHAR

Data types supported by the Extended Search wrapper

The following table lists the DB2 data types that are supported by the Extended Search wrapper.

Table 69. Extended Search data types that map to DB2 data types

Extended Search data types	DB2 data type
Date	DATE
Double	DOUBLE
Integer	INTEGER
String	VARCHAR

Data types supported by the HMMER wrapper

The following table lists the DB2 data types that are supported by the HMMER wrapper.

Table 70. HMMER data types that map to DB2 data types

HMMER data types	DB2 data type
character	CLOB The maximum length allowed for the CLOB data type is 5 megabytes.
character	DOUBLE
character	FLOAT
character	INTEGER
character	VARCHAR

Data types supported by the table-structured file wrapper

The following table lists the DB2 data types that are supported by the table-structured file wrapper.

Table 71. Table-structured file data types that map to DB2 data types

Table-structured file data types	DB2 data type
character	CHARACTER
character	CLOB The maximum length allowed for the CLOB data type is 5 megabytes.
number	DECIMAL
number	DOUBLE
number	FLOAT
integer	INTEGER
number	REAL
integer	SMALLINT
character	VARCHAR

Data types supported by the Web services wrapper

The following table lists the DB2 data types that are supported by the Web services wrapper. The Web services wrapper uses XML data types.

Table 72. XML data types that map to DB2 data types for the Web services wrapper

XML data types	DB2 data type
character	CHARACTER
character	CHARACTER FOR BIT DATA
character	CLOB
date	DATE
number	DECIMAL
number	DOUBLE
number	FLOAT
integer	INTEGER
number	REAL
integer	SMALLINT
character	VARCHAR
character	VARCHAR FOR BIT DATA

Data types supported by the WebSphere Business Integration wrapper

The following table lists the DB2 data types that are supported by the WebSphere Business Integration wrapper. The WebSphere Business Integration wrapper uses XML data types.

Table 73. XML data types that map to DB2 data types for the WebSphere Business Integration wrapper

XML data types	DB2 data type
character	CHARACTER
character	CHARACTER FOR BIT DATA
character	CLOB
date	DATE
number	DECIMAL
number	DOUBLE
number	FLOAT
integer	INTEGER
number	REAL
integer	SMALLINT
character	VARCHAR
character	VARCHAR FOR BIT DATA

Data types supported by the XML wrapper

The following table lists the DB2 data types that are supported by the XML wrapper

Table 74. XML data types that map to DB2 data types for the XML wrapper

XML data types	DB2 data type
character	CHARACTER
character	CHARACTER FOR BIT DATA
character	CLOB The maximum length allowed for the CLOB data type is 5 megabytes.
date	DATE
number	DECIMAL
number	DOUBLE
number	FLOAT
integer	INTEGER
number	REAL
integer	SMALLINT
character	VARCHAR
character	VARCHAR FOR BIT DATA

Chapter 31. Federated database systems monitor elements

A federated system is a multidatabase server that provides remote data access. It provides client access to diverse data sources that can reside on different platforms, both IBM and other vendors, relational and non-relational. It integrates access to distributed data and presents a single database image of a heterogeneous environment to its users.

The following elements list information about the total access to a data source by applications running in a DB2 federated system and information about access to a data source by a given application running in a federated server instance. They include:

- datasource_name - Data Source Name monitor element
- disconnects - Disconnects monitor element
- insert_sql_stmts - Inserts monitor element
- update_sql_stmts - Updates monitor element
- delete_sql_stmts - Deletes monitor element
- create_nickname - Create Nicknames monitor element
- passthrus - Pass-Through monitor element
- stored_procs - Stored Procedures monitor element
- remote_locks - Remote Locks monitor element
- sp_rows_selected - Rows Returned by Stored Procedures monitor element
- select_time - Query Response Time monitor element
- insert_time - Insert Response Time monitor element
- update_time - Update Response Time monitor element
- delete_time - Delete Response Time monitor element
- create_nickname_time - Create Nickname Response Time monitor element
- passthru_time - Pass-Through Time monitor element
- stored_proc_time - Stored Procedure Time monitor element
- remote_lock_time - Remote Lock Time monitor element

Chapter 32. SYSPROC.NNSTAT stored procedure

Retrieve currently available statistics on one or more nicknames.

Authorization

None.

Syntax

```
CALL SYSPROC.NNSTAT(
     'server/null', 'schema/null','nickname/null', 'filepath/null',?,?)
```

Parameter descriptions

Server The server on which the federated server gathers the nickname statistics. This server is what the user registers to define a data source in the federated database. If you specify one nickname, you can specify NULL for this parameter.

Schema
If NULL is specified, the federated server retrieves all the nicknames under the given server. If the Server parameter is NULL, the federated server retrieves the statistics of the nickname under the given schema. If the Schema parameter and Nickname parameter are NULL and you specify a server, the federated server retrieves statistics on the given server.

Nickname
The name of the nickname. If you specify a nickname, you must also specify a schema.

Log_File_Path
The path name and file name for the log file. The federated server creates the log file on the server. The directories that you list in the path must exist. On Windows systems use two backslashes to specify the log path. For example: c:\\temp\\nnstat.log. If you specify NULL, the federated server does not create a log.

Output parameters

out_SQLCode
The SQL error as a result of the statistics.

out_Trace
The trace.

Examples

```
CALL SYSPROC.NNSTAT(
    'NULL', 'ADMIN','STAFF','/home/iiuser/reportlogs/log1.txt',?,?)

CALL SYSPROC.NNSTAT(
    'DB2SERV','ADMIN','NULL','c:\\reports\\log1.txt',?,?)

CALL SYSPROC.NNSTAT(
    'DB2SERV','NULL','NULL','NULL',?,?)
```

Related concepts:

- "Nickname statistics update facility - overview" on page 187

Related tasks:

- "Retrieving nickname statistics" on page 188

Related reference:

- "Retrieving nickname statistics from the command line - examples" on page 190

Chapter 33. High availability disaster recovery with federated data sources

In high availability disaster recovery (HADR), a primary database sends logs to a standby database. If the primary database becomes unavailable, you can switch to the standby database.

Requirements for relational wrappers

To implement HADR to work on a federated database that uses relational wrappers, the following conditions must be met:

- You must define data source partition names identically on both systems. For example, if a system with the primary database has a partition INF1 that refers to an Informix instance at host xyz, then on the system with the secondary database, the partition INF1 must refer to the same Informix instance at the same host.
- For Oracle data sources, the client version on the system with the primary database must be the same as the client version on the system with the secondary database. For other data sources, the client versions must be the same.
- The following environment variables in the federated configuration file must be the same on both the primary database and the secondary database systems:

  Oracle: NLS_LANG

  Informix: CLIENT_LOCALE, DB_LOCALE, DBNLS

  Sybase: SYBASE_CHARSET

  Teradata: TERADATA_CHARSET

Requirements for nonrelational wrappers

For table-structured flat file, XML, and Excel wrappers, external files must be accessible with the same path names on both systems.

For HMMER and BLAST wrappers, each system needs TCP/IP access to the appropriate daemon, if the daemon does not run on the same computer as DB2 Information Integrator.

For the Documentum wrapper, the Documentum client must be on the secondary system. The client configuration files must direct connections to the same docbroker.

For WebSphere Business Integration:

- If the WebSphere MQ manager is not on the same system as the wrapper, then you must install the WebSphere MQ client on the secondary system. The secondary system must be able to access the same WebSphere MQ manager as the primary system.
- If the WebSphere MQ manager and the wrapper are on the same system, but the adapter is on a different computer, then your secondary system must have an WebSphere MQ manager that is running. If your primary system fails, you must shutdown the adapter and reconfigure the adapter to point to the new WebSphere MQ manager. The new WebSphere MQ manager must have the same queue names defined as the primary system. You then must restart the adapter to connect to the secondary WebSphere MQ manager.

- If the WebSphere MQ manager, the wrapper, and the adapter are on the same system, then you must create a replica of this setup on the secondary computer.

Related concepts:

- "High availability disaster recovery overview" in the *Data Recovery and High Availability Guide and Reference*

Chapter 34. Query gateway server information for engine traps

To help IBM Service diagnose a DB2 UDB engine failure, DB2 UDB can write some of the query gateway server information to a file. To write this information to a file:

- Set the database manager configuration parameter FEDERATED to YES.
- Create at least one server object.
- If you want to add the query gateway memory pool information to the same file, set the DB2MEMDBG register variable to ON.

If a trap occurs, the logger places a message in the db2diag.log file. The message specifies the full path of the file that contains the query gateway server information.

DB2 Information Integrator documentation

This topic provides information about the documentation that is available for DB2 Information Integrator. The tables in this topic provide the official document title, form number, and location of each PDF book. To order a printed book, you must know either the official book title or the document form number. Titles, file names, and the locations of the DB2 Information Integrator release notes and installation requirements are also provided in this topic.

This topic contains the following sections:

- Accessing DB2 Information Integrator documentation
- Documentation for replication function on z/OS
- Documentation for event publishing function for DB2 Universal Database on z/OS
- Documentation for event publishing function for IMS and VSAM on z/OS
- Documentation for event publishing and replication function on Linux, UNIX, and Windows
- Documentation for federated function on z/OS
- Documentation for federated function on Linux, UNIX, and Windows
- Documentation for enterprise search on Linux, UNIX, and Windows
- Release notes and installation requirements

Accessing DB2 Information Integrator documentation

All DB2 Information Integrator books and release notes are available in PDF files from the DB2 Information Integrator Support Web site at www.ibm.com/software/data/integration/db2ii/support.html.

To access the latest DB2 Information Integrator product documentation, from the DB2 Information Integrator Support Web site, click on the Product Information link, as shown in Figure 10 on page 306.

Figure 10. Accessing the Product Information link from DB2 Information Integrator Support Web site

You can access the latest DB2 Information Integrator documentation, in all supported languages, from the Product Information link:

- DB2 Information Integrator product documentation in PDF files
- Fix pack product documentation, including release notes
- Instructions for downloading and installing the DB2 Information Center for Linux, UNIX, and Windows
- Links to the DB2 Information Center online

Scroll though the list to find the product documentation for the version of DB2 Information Integrator that you are using.

The DB2 Information Integrator Support Web site also provides support documentation, IBM Redbooks, white papers, product downloads, links to user groups, and news about DB2 Information Integrator.

You can also view and print the DB2 Information Integrator PDF books from the *DB2 PDF Documentation* CD.

To view or print the PDF documentation:

1. From the root directory of the *DB2 PDF Documentation* CD, open the index.htm file.
2. Click the language that you want to use.
3. Click the link for the document that you want to view.

Documentation about replication function on z/OS

Table 75. DB2 Information Integrator documentation about replication function on z/OS

Name	Form number	Location
ASNCLP Program Reference for Replication and Event Publishing	N/A	DB2 Information Integrator Support Web site
Introduction to Replication and Event Publishing	GC18-7567	DB2 Information Integrator Support Web site
Migrating to SQL Replication	N/A	DB2 Information Integrator Support Web site
Replication and Event Publishing Guide and Reference	SC18-7568	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Replication Installation and Customization Guide for z/OS	SC18-9127	DB2 Information Integrator Support Web site
SQL Replication Guide and Reference	SC27-1121	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Tuning for Replication and Event Publishing Performance	N/A	DB2 Information Integrator Support Web site
Tuning for SQL Replication Performance	N/A	DB2 Information Integrator Support Web site
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	N/A	• In the DB2 Information Center, **Product Overviews** > **Information Integration** > **DB2 Information Integrator overview** > **Problems, workarounds, and documentation updates** • DB2 Information Integrator Installation launchpad • DB2 Information Integrator Support Web site • The *DB2 Information Integrator* product CD

Documentation about event publishing function for DB2 Universal Database on z/OS

Table 76. DB2 Information Integrator documentation about event publishing function for DB2 Universal Database on z/OS

Name	Form number	Location
ASNCLP Program Reference for Replication and Event Publishing	N/A	DB2 Information Integrator Support Web site
Introduction to Replication and Event Publishing	GC18-7567	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Replication and Event Publishing Guide and Reference	SC18-7568	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Tuning for Replication and Event Publishing Performance	N/A	DB2 Information Integrator Support Web site
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	N/A	• In the DB2 Information Center, **Product Overviews** > **Information Integration** > **DB2 Information Integrator overview** > **Problems, workarounds, and documentation updates** • DB2 Information Integrator Installation launchpad • DB2 Information Integrator Support Web site • The *DB2 Information Integrator* product CD

Documentation about event publishing function for IMS and VSAM on z/OS

Table 77. DB2 Information Integrator documentation about event publishing function for IMS and VSAM on z/OS

Name	Form number	Location
Client Guide for Classic Federation and Event Publisher for z/OS	SC18-9160	DB2 Information Integrator Support Web site
Data Mapper Guide for Classic Federation and Event Publisher for z/OS	SC18-9163	DB2 Information Integrator Support Web site
Getting Started with Event Publisher for z/OS	GC18-9186	DB2 Information Integrator Support Web site
Installation Guide for Classic Federation and Event Publisher for z/OS	GC18-9301	DB2 Information Integrator Support Web site
Operations Guide for Event Publisher for z/OS	SC18-9157	DB2 Information Integrator Support Web site

Table 77. DB2 Information Integrator documentation about event publishing function for IMS and VSAM on z/OS (continued)

Name	Form number	Location
Planning Guide for Event Publisher for z/OS	SC18-9158	DB2 Information Integrator Support Web site
Reference for Classic Federation and Event Publisher for z/OS	SC18-9156	DB2 Information Integrator Support Web site
System Messages for Classic Federation and Event Publisher for z/OS	SC18-9162	DB2 Information Integrator Support Web site
Release Notes for IBM DB2 Information Integrator Event Publisher for IMS for z/OS	N/A	DB2 Information Integrator Support Web site
Release Notes for IBM DB2 Information Integrator Event Publisher for VSAM for z/OS	N/A	DB2 Information Integrator Support Web site

Documentation about event publishing and replication function on Linux, UNIX, and Windows

Table 78. DB2 Information Integrator documentation about event publishing and replication function on Linux, UNIX, and Windows

Name	Form number	Location
ASNCLP Program Reference for Replication and Event Publishing	N/A	DB2 Information Integrator Support Web site
Installation Guide for Linux, UNIX, and Windows	GC18-7036	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Introduction to Replication and Event Publishing	GC18-7567	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Migrating to SQL Replication	N/A	DB2 Information Integrator Support Web site
Replication and Event Publishing Guide and Reference	SC18-7568	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
SQL Replication Guide and Reference	SC27-1121	DB2 Information Integrator Support Web site
Tuning for Replication and Event Publishing Performance	N/A	DB2 Information Integrator Support Web site
Tuning for SQL Replication Performance	N/A	DB2 Information Integrator Support Web site

Table 78. DB2 Information Integrator documentation about event publishing and replication function on Linux, UNIX, and Windows (continued)

Name	Form number	Location
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	N/A	• In the DB2 Information Center, **Product Overviews** > **Information Integration** > **DB2 Information Integrator overview** > **Problems, workarounds, and documentation updates** • DB2 Information Integrator Installation launchpad • DB2 Information Integrator Support Web site • The *DB2 Information Integrator* product CD

Documentation about federated function on z/OS

Table 79. DB2 Information Integrator documentation about federated function on z/OS

Name	Form number	Location
Client Guide for Classic Federation and Event Publisher for z/OS	SC18-9160	DB2 Information Integrator Support Web site
Data Mapper Guide for Classic Federation and Event Publisher for z/OS	SC18-9163	DB2 Information Integrator Support Web site
Getting Started with Classic Federation for z/OS	GC18-9155	DB2 Information Integrator Support Web site
Installation Guide for Classic Federation and Event Publisher for z/OS	GC18-9301	DB2 Information Integrator Support Web site
Reference for Classic Federation and Event Publisher for z/OS	SC18-9156	DB2 Information Integrator Support Web site
System Messages for Classic Federation and Event Publisher for z/OS	SC18-9162	DB2 Information Integrator Support Web site
Transaction Services Guide for Classic Federation for z/OS	SC18-9161	DB2 Information Integrator Support Web site
Release Notes for IBM DB2 Information Integrator Classic Federation for z/OS	N/A	DB2 Information Integrator Support Web site

Documentation about federated function on Linux, UNIX, and Windows

Table 80. DB2 Information Integrator documentation about federated function on Linux, UNIX, and Windows

Name	Form number	Location
Application Developer's Guide	SC18-7359	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site

Table 80. DB2 Information Integrator documentation about federated function on Linux, UNIX, and Windows (continued)

Name	Form number	Location
C++ API Reference for Developing Wrappers	SC18-9172	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Data Source Configuration Guide	N/A	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Federated Systems Guide	SC18-7364	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Guide to Configuring the Content Connector for VeniceBridge	N/A	DB2 Information Integrator Support Web site
Installation Guide for Linux, UNIX, and Windows	GC18-7036	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Java API Reference for Developing Wrappers	SC18-9173	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Migration Guide	SC18-7360	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Wrapper Developer's Guide	SC18-9174	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	N/A	• In the DB2 Information Center, **Product Overviews** > **Information Integration** > **DB2 Information Integrator overview** > **Problems, workarounds, and documentation updates** • DB2 Information Integrator Installation launchpad • DB2 Information Integrator Support Web site • The *DB2 Information Integrator* product CD

Documentation about enterprise search function on Linux, UNIX, and Windows

Table 81. DB2 Information Integrator documentation about enterprise search function on Linux, UNIX, and Windows

Name	Form number	Location
Administering Enterprise Search	SC18-9283	DB2 Information Integrator Support Web site
Installation Guide for Enterprise Search	GC18-9282	DB2 Information Integrator Support Web site
Programming Guide and API Reference for Enterprise Search	SC18-9284	DB2 Information Integrator Support Web site
Release Notes for Enterprise Search	N/A	DB2 Information Integrator Support Web site

Release notes and installation requirements

Release notes provide information that is specific to the release and fix pack level for your product and include the latest corrections to the documentation for each release.

Installation requirements provide information that is specific to the release of your product.

Table 82. DB2 Information Integrator Release Notes and Installation Requirements

Name	File name	Location
Installation Requirements for IBM DB2 Information Integrator Event Publishing Edition, Replication Edition, Standard Edition, Advanced Edition, Advanced Edition Unlimited, Developer Edition, and Replication for z/OS	Prereqs	• The *DB2 Information Integrator* product CD • DB2 Information Integrator Installation Launchpad
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	ReleaseNotes	• In the DB2 Information Center, **Product Overviews** > **Information Integration** > **DB2 Information Integrator overview** > **Problems, workarounds, and documentation updates** • DB2 Information Integrator Installation launchpad • DB2 Information Integrator Support Web site • The *DB2 Information Integrator* product CD
Release Notes for IBM DB2 Information Integrator Event Publisher for IMS for z/OS	N/A	DB2 Information Integrator Support Web site

Table 82. DB2 Information Integrator Release Notes and Installation Requirements (continued)

Name	File name	Location
Release Notes for IBM DB2 Information Integrator Event Publisher for VSAM for z/OS	N/A	DB2 Information Integrator Support Web site
Release Notes for IBM DB2 Information Integrator Classic Federation for z/OS	N/A	DB2 Information Integrator Support Web site
Release Notes for Enterprise Search	N/A	DB2 Information Integrator Support Web site

To view the installation requirements and release notes that are on the product CD:

- On Windows operating systems, enter:

  ```
  x:\doc\%L
  ```

  x is the Windows CD drive letter and *%L* is the locale of the documentation that you want to use, for example, en_US.
- On UNIX operating systems, enter:

  ```
  /cdrom/doc/%L/
  ```

  cdrom refers to the UNIX mount point of the CD and *%L* is the locale of the documentation that you want to use, for example, en_US.

Accessibility

Accessibility features help users with physical disabilities, such as restricted mobility or limited vision, to use software products successfully. The following list specifies the major accessibility features in DB2® Version 8 products:

- All DB2 functionality is available using the keyboard for navigation instead of the mouse. For more information, see "Keyboard input and navigation."
- You can customize the size and color of the fonts on DB2 user interfaces. For more information, see "Accessible display."
- DB2 products support accessibility applications that use the Java™ Accessibility API. For more information, see "Compatibility with assistive technologies" on page 316.
- DB2 documentation is provided in an accessible format. For more information, see "Accessible documentation" on page 316.

Keyboard input and navigation

Keyboard input

You can operate the DB2 tools using only the keyboard. You can use keys or key combinations to perform operations that can also be done using a mouse. Standard operating system keystrokes are used for standard operating system operations.

For more information about using keys or key combinations to perform operations, see Keyboard shortcuts and accelerators: Common GUI help.

Keyboard navigation

You can navigate the DB2 tools user interface using keys or key combinations.

For more information about using keys or key combinations to navigate the DB2 Tools, see Keyboard shortcuts and accelerators: Common GUI help.

Keyboard focus

In UNIX® operating systems, the area of the active window where your keystrokes will have an effect is highlighted.

Accessible display

The DB2 tools have features that improve accessibility for users with low vision or other visual impairments. These accessibility enhancements include support for customizable font properties.

Font settings

You can select the color, size, and font for the text in menus and dialog windows, using the Tools Settings notebook.

For more information about specifying font settings, see Changing the fonts for menus and text: Common GUI help.

Non-dependence on color

You do not need to distinguish between colors in order to use any of the functions in this product.

Compatibility with assistive technologies

The DB2 tools interfaces support the Java Accessibility API, which enables you to use screen readers and other assistive technologies with DB2 products.

Accessible documentation

Documentation for DB2 is provided in XHTML 1.0 format, which is viewable in most Web browsers. XHTML allows you to view documentation according to the display preferences set in your browser. It also allows you to use screen readers and other assistive technologies.

Syntax diagrams are provided in dotted decimal format. This format is available only if you are accessing the online documentation using a screen-reader.

Related concepts:

- "Dotted decimal syntax diagrams" in the *Infrastructure Topics (DB2 Common Files)*

Related tasks:

- "Keyboard shortcuts and accelerators: Common GUI help"
- "Changing the fonts for menus and text: Common GUI help"

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country/region or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, 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 Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems, and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious, and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs, in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. 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 IBM's application programming interfaces.

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:

IBM
AIX
DataJoiner
DB2
DB2 Connect
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
Informix
iSeries
Lotus
Lotus Notes
MVS
OS/390
VM/ESA
VSE/ESA
WebSphere
z/OS

The following terms are trademarks or registered trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product or service names may be trademarks or service marks of others.

Index

E

F

G

O

P

Q

R

S

T

U

V

W

X

Contacting IBM

To contact IBM customer service in the United States or Canada, call 1-800-IBM-SERV (1-800-426-7378).

To learn about available service options, call one of the following numbers:

- In the United States: 1-888-426-4343
- In Canada: 1-800-465-9600

To locate an IBM office in your country or region, see the IBM Directory of Worldwide Contacts on the Web at www.ibm.com/planetwide.

Product information

Information about DB2 Information Integrator is available by telephone or on the Web.

If you live in the United States, you can call one of the following numbers:

- To order products or to obtain general information: 1-800-IBM-CALL (1-800-426-2255)
- To order publications: 1-800-879-2755

On the Web, go to www.ibm.com/software/data/integration/db2ii/support.html. This site contains the latest information about:

- The technical library
- Ordering books
- Client downloads
- Newsgroups
- Fix packs
- News
- Links to Web resources

Comments on the documentation

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

- Send your comments using the online readers' comment form at www.ibm.com/software/data/rcf.
- Send your comments by e-mail to comments@us.ibm.com. Include the name of the product, the version number of the product, and the name and part number of the book (if applicable). If you are commenting on specific text, please include the location of the text (for example, a title, a table number, or a page number).

Printed in USA

SC18-7364-01

Spine information:

IBM DB2 Information Integrator **Federated Systems Guide** Version 8.2