Configuration Guide for Federated Data Sources
Configuration Guide for Federated Data Sources
Before using this information and the product that it supports, read the information in "Notices" on page 493.
Contents

Chapter 1. Configuring the federated server .............................................. 1
Issuing commands in a federated system .............................................. 1
Verifying the setup of the federated server ........................................ 1
Enabling the federated server to access data sources .......................... 1
Setting required environment variables for data source clients ........... 2
Verifying that library files are linked to data source client software ....... 4
Managing the federated license key and license policy ......................... 6
Creating a federated database .............................................................. 6
Code sets, collating sequences, and national language support ............. 7
Configuring the federated server to access data sources ......................... 11
Case sensitivity and the correct use of quotation marks ....................... 13
Configuring multiple federated servers to access data sources ............. 16

Chapter 2. Configuring access to BioRS data sources ......................... 19
BioRS wrapper ...................................................................................... 19
Adding BioRS data sources to a federated server ................................. 20
Registering the custom functions for the BioRS wrapper .................... 21
Registering the BioRS wrapper ............................................................ 22
Registering the server definition for a BioRS data source .................... 25
Creating the user mappings for a BioRS data source .......................... 28
Registering nicknames for BioRS data sources .................................... 30
Custom functions and BioRS queries .................................................... 34
Equity operations in BioRS queries ..................................................... 37
Equijoin predicates for the BioRS wrapper .......................................... 39
The BioRS AllText element ................................................................. 40
BioRS data source - Example queries ................................................ 41
Optimizing BioRS wrapper performance ............................................. 46
Guidelines for optimizing BioRS wrapper performance ....................... 46
BioRS statistical information .............................................................. 47
Determining BioRS database cardinality statistics ............................... 48
Updating BioRS nickname cardinality statistics .................................. 48
Updating BioRS column cardinality statistics ..................................... 49
Updating BioRS _ID_ column cardinality ........................................... 50

Chapter 3. Configuring access to BLAST data sources .................. 53
BLAST wrapper .................................................................................... 53
Adding BLAST data sources to a federated server ............................... 56
Verifying the versions of the BLAST server files .................................. 57
Configuring the BLAST daemon ......................................................... 57
Starting the BLAST daemon .............................................................. 61
Registering the BLAST wrapper ......................................................... 64
Creating the user mappings for a BLAST data source (optional) .......... 65
Registering the server definition for a BLAST data source .................. 67
Registering nicknames for BLAST data sources .................................. 70
Querying BLAST data sources ........................................................... 77
Setting up TurboBlast to work with the BLAST wrapper .................... 77
Construct BLAST SQL queries .......................................................... 78
BLAST data source - example queries ............................................... 79
Guidelines for optimizing BLAST query performance ....................... 81

Chapter 4. Configuring access to DB2 data sources ..................... 83
Cataloging a DB2 node entry ............................................................. 83
Cataloging the remote DB2 database ............................................... 83
Registering the DB2 wrapper ............................................................. 84
DB2 wrapper library files .................................................................. 84
Registering server definitions for DB2 data sources .......................... 85
CREATE SERVER statement - Examples for the DB2 wrapper .......... 86
Creating user mappings for DB2 data sources ................................... 87
CREATE USER MAPPING statement - Examples for the DB2 wrapper .. 87
Testing the connection to the DB2 data source server ......................... 88
Registering nicknames for DB2 tables and views ............................... 88
CREATE NICKNAME statement - Examples for DB2 data sources .... 89

Chapter 5. Configuring access to Entrez data sources ................. 91
Entrez Wrapper ................................................................................... 91
Adding Entrez data sources to a federated server ............................... 93
Registering the custom functions for the Entrez wrapper ................. 93
Registering the Entrez wrapper ......................................................... 95
Registering the server definition for Entrez data sources .................... 97
Creating the user mappings for an Entrez data source (optional) ........ 100
Access Entrez using a proxy server ............................................... 101
Nicknames for the Entrez data sources ............................................ 102
Queries and custom functions for Entrez data sources ...................... 110
Query the Entrez databases using the custom functions .................... 110
Relational predicates for the Entrez wrapper .................................... 112
Records returned from Entrez queries ............................................. 112
Entrez data sources - Example queries ............................................. 113
Fixed columns for the Nucleotide nicknames .................................. 115
Fixed columns for the OMIM nicknames ......................................... 118
Fixed columns for the PubMed nicknames ....................................... 126
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registering the server definitions for an ODBC data source</td>
<td>215</td>
</tr>
<tr>
<td>CREATE SERVER statement - Examples for the ODBC wrapper</td>
<td>215</td>
</tr>
<tr>
<td>Creating a user mapping for an ODBC data source</td>
<td>216</td>
</tr>
<tr>
<td>CREATE USER MAPPING statement - Examples for the ODBC wrapper</td>
<td>217</td>
</tr>
<tr>
<td>Testing the connection to the ODBC data source server</td>
<td>218</td>
</tr>
<tr>
<td>Registering nicknames for ODBC data source tables and views</td>
<td>218</td>
</tr>
<tr>
<td>CREATE NICKNAME statement - Examples for the ODBC wrapper</td>
<td>219</td>
</tr>
<tr>
<td>Optimizing ODBC wrapper performance with the ODBC tuning utility (db2fedsvrcfg)</td>
<td>219</td>
</tr>
<tr>
<td>db2fedsvrcfg command syntax - ODBC tuning utility</td>
<td>221</td>
</tr>
<tr>
<td>Test table definitions for the ODBC tuning utility (db2fedsvrcfg)</td>
<td>222</td>
</tr>
<tr>
<td><strong>Chapter 12. Configuring access to OLE DB data sources</strong></td>
<td>225</td>
</tr>
<tr>
<td>Registering the OLE DB wrapper</td>
<td>225</td>
</tr>
<tr>
<td>OLE DB wrapper library files</td>
<td>226</td>
</tr>
<tr>
<td>Registering the server definitions for an OLE DB data source</td>
<td>226</td>
</tr>
<tr>
<td>CREATE SERVER statement - Examples for the OLE DB wrapper</td>
<td>227</td>
</tr>
<tr>
<td>Creating the user mappings for an OLE DB data source</td>
<td>228</td>
</tr>
<tr>
<td>CREATE USER MAPPING statement - Examples for the OLE DB wrapper</td>
<td>228</td>
</tr>
<tr>
<td><strong>Chapter 13. Configuring access to Oracle data sources</strong></td>
<td>231</td>
</tr>
<tr>
<td>Setting the Oracle environment variables</td>
<td>231</td>
</tr>
<tr>
<td>Oracle environment variables</td>
<td>232</td>
</tr>
<tr>
<td>Setting up and testing the Oracle client configuration file</td>
<td>235</td>
</tr>
<tr>
<td>Registering the Oracle wrapper</td>
<td>236</td>
</tr>
<tr>
<td>Oracle wrapper library files</td>
<td>236</td>
</tr>
<tr>
<td>Registering the server definitions for an Oracle data source</td>
<td>237</td>
</tr>
<tr>
<td>CREATE SERVER statement - Examples for the Oracle wrapper</td>
<td>238</td>
</tr>
<tr>
<td>Creating the user mappings for an Oracle data source</td>
<td>239</td>
</tr>
<tr>
<td>CREATE USER MAPPING statement - Examples for the Oracle wrapper</td>
<td>239</td>
</tr>
<tr>
<td>Testing the connection to the Oracle server</td>
<td>240</td>
</tr>
<tr>
<td>Troubleshooting connectivity problems with Oracle data sources</td>
<td>240</td>
</tr>
<tr>
<td>Registering nicknames for Oracle tables and views</td>
<td>241</td>
</tr>
<tr>
<td>CREATE NICKNAME statement - Examples for the Oracle wrapper</td>
<td>242</td>
</tr>
<tr>
<td><strong>Chapter 14. Configuring access to scripts as data sources</strong></td>
<td>243</td>
</tr>
<tr>
<td>Script wrapper overview</td>
<td>243</td>
</tr>
<tr>
<td>Adding scripts as data sources to a federated system</td>
<td>244</td>
</tr>
<tr>
<td>Registering the custom function for the script</td>
<td>244</td>
</tr>
<tr>
<td>Configuring the script daemon</td>
<td>245</td>
</tr>
<tr>
<td>Starting the script daemon</td>
<td>248</td>
</tr>
<tr>
<td>Registering the script wrapper</td>
<td>250</td>
</tr>
<tr>
<td>Registering the server definition for a script as a data source (DB2 command line)</td>
<td>251</td>
</tr>
<tr>
<td>Registering nicknames for scripts (DB2 command line)</td>
<td>252</td>
</tr>
<tr>
<td>CREATE NICKNAME statement - examples for the script wrapper</td>
<td>253</td>
</tr>
<tr>
<td>Script wrapper nickname options</td>
<td>253</td>
</tr>
<tr>
<td>SQL queries with the script wrapper</td>
<td>256</td>
</tr>
<tr>
<td>Optimizing script wrapper performance</td>
<td>257</td>
</tr>
<tr>
<td><strong>Chapter 15. Configuring access to Sybase data sources</strong></td>
<td>259</td>
</tr>
<tr>
<td>Sybase wrapper support for Adaptive Server Enterprise (ASE)</td>
<td>259</td>
</tr>
<tr>
<td>Setting the Sybase environment variables</td>
<td>260</td>
</tr>
<tr>
<td>Sybase environment variables</td>
<td>262</td>
</tr>
<tr>
<td>Setting up and testing the Sybase client configuration file (Windows)</td>
<td>263</td>
</tr>
<tr>
<td>Setting up and testing the Sybase client configuration file (UNIX)</td>
<td>264</td>
</tr>
<tr>
<td>Registering the Sybase wrapper</td>
<td>265</td>
</tr>
<tr>
<td>Sybase wrapper library files</td>
<td>265</td>
</tr>
<tr>
<td>Registering the server definitions for a Sybase data source</td>
<td>266</td>
</tr>
<tr>
<td>CREATE SERVER statement - Examples for the Sybase wrapper</td>
<td>267</td>
</tr>
<tr>
<td>Creating the user mappings for a Sybase data source</td>
<td>268</td>
</tr>
<tr>
<td>CREATE USER MAPPING statement - Examples for the Sybase wrapper</td>
<td>269</td>
</tr>
<tr>
<td>Registering nicknames for Sybase tables and views</td>
<td>269</td>
</tr>
<tr>
<td>CREATE NICKNAME statement - Examples for the Sybase wrapper</td>
<td>270</td>
</tr>
<tr>
<td>Registering nicknames for Sybase tables and views</td>
<td>271</td>
</tr>
<tr>
<td>CREATE NICKNAME statement - Examples for the Sybase wrapper</td>
<td>271</td>
</tr>
<tr>
<td>Problems loading the Sybase wrapper library</td>
<td>272</td>
</tr>
<tr>
<td>Missing SYBASE environment variable</td>
<td>272</td>
</tr>
<tr>
<td>Missing Sybase node name</td>
<td>273</td>
</tr>
<tr>
<td><strong>Chapter 16. Configuring access to table-structured file data sources</strong></td>
<td>275</td>
</tr>
<tr>
<td>Table-structured files - overview</td>
<td>275</td>
</tr>
<tr>
<td>Attributes of table-structured files</td>
<td>275</td>
</tr>
<tr>
<td>Table-structured files wrapper</td>
<td>276</td>
</tr>
<tr>
<td>Adding table-structured file data sources to a federated server</td>
<td>277</td>
</tr>
<tr>
<td>Registering the table-structured file wrapper</td>
<td>277</td>
</tr>
<tr>
<td>Registering the server definition for table-structured files</td>
<td>279</td>
</tr>
<tr>
<td>Registering nicknames for table-structured files</td>
<td>279</td>
</tr>
<tr>
<td>File access control model for the table-structured file wrapper</td>
<td>285</td>
</tr>
</tbody>
</table>
Chapter 17. Configuring access to Teradata data sources .......................... 287
Testing the connection to the Teradata server ....................................... 287
Verifying that the Teradata library is enabled for run-time linking (AIX) .... 288
Setting the Teradata environment variables ............................................. 288
Teradata environment variables ............................................................... 289
Verifying the character set on the Teradata server .................................. 291
Registering the Teradata wrapper ............................................................ 292
Teradata wrapper library files ................................................................. 293
Registering the server definitions for a Teradata data source ................. 294
CREATE SERVER statement - Examples for the Teradata wrapper ........... 295
Creating the user mapping for a Teradata data source ......................... 296
CREATE USER MAPPING statement - Examples for the Teradata wrapper ... 296
Testing the connection from the federated server to the Teradata server .... 297
Troubleshooting data source connection errors ...................................... 298
Registering nicknames for Teradata tables and views ............................. 298
Teradata nicknames on federated servers .............................................. 299
CREATE NICKNAME statement - Examples for the Teradata wrapper ....... 300
Enabling run-time linking for libciv2.so (AIX) ...................................... 300
Troubleshooting character sets for Teradata data sources ...................... 301
Troubleshooting UPDATE or DELETE operation errors on nicknames .... 301
Working with Teradata access logging .................................................... 302

Chapter 18. Configuring access to Web services data sources ............... 303
Web services and the Web services wrapper .......................................... 303
Registering the Web services wrapper .................................................... 309
Web services wrapper library files ......................................................... 309
Registering the server definition for Web services data sources ................ 310
CREATE SERVER statement - Examples for Web services wrapper .......... 310
Enabling security through the Web services wrapper .............................. 312
Registering nicknames for Web services data sources ............................ 312
Registering nicknames for Web services data sources (DB2 command line) . 313
Registering nicknames for Web services data sources (DB2 Control Center) . 314
CREATE NICKNAME statement - examples for the Web services wrapper . 315
SOAP header access .............................................................................. 315
Nicknames and XPath expressions ......................................................... 321
TEMPLATE option at the nickname and column levels ......................... 323
Creating federated views for Web services nicknames ............................ 324
Query restrictions for wrappers for business applications and Web services . 325
Web services data sources – example queries ....................................... 328

Chapter 19. Configuring access to business application data sources ...... 331
WebSphere Business Integration wrapper .............................................. 331
Configuring the WebSphere Business Integration Adapters .................. 333
Business object definitions ..................................................................... 333
Configuration properties for business object definitions ....................... 335
WebSphere MQ message queues for the WebSphere Business Integration wrapper and the adapter ................................................................. 335
Registering the WebSphere Business Integration wrapper (DB2 command line) . 336
Business integration wrapper library files .............................................. 337
Registering the server definition for business application data sources (DB2 command line) .......................................................... 337
Registering nicknames for business application data sources ............... 338
Registering nicknames for business application data sources (DB2 Control Center) . 339
Registering nicknames for business application data sources (DB2 command line) . 340
CREATE NICKNAME statement – Examples for the WebSphere Business Integration wrapper ............................... 341
Nicknames and XPath expressions ......................................................... 351
TEMPLATE option at the nickname and column levels ......................... 352
Query restrictions for wrappers for business applications and Web services ................................................................. 352
Business application data sources – Example queries ......................... 355

Chapter 20. Configuring access to XML data sources .......................... 359
XML wrapper ......................................................................................... 359
Adding XML to a federated system ....................................................... 362
Registering the XML wrapper ............................................................... 362
Registering the server definition for an XML data source ...................... 365
Access XML files using a proxy server .................................................. 368
Nicknames for XML data sources ......................................................... 370
Queries for XML data sources .............................................................. 378
Creating federated views for the XML wrapper nicknames ................... 379
CREATE VIEW statement - examples for the XML wrapper ................. 380
Query optimization tips for the XML cost model facility ......................... 381
XML data source - example queries ...................................................... 382

Chapter 21. Data source support for federated features ....................... 385
Chapter 22. Data source options reference ................................................. 389
  Blast options reference ................................................................. 389
  BioRS options reference .............................................................. 391
  DB2 database options reference .................................................... 395
  Entrez options reference ............................................................... 400
  Excel options reference ............................................................... 402
  HMMER options reference ............................................................. 403
  Informix options reference ........................................................... 406
  Microsoft SQL Server options reference ........................................... 411
  ODBC options reference ............................................................... 415
  Oracle options reference .............................................................. 420
  Script options reference ............................................................... 424
  Sybase options reference .............................................................. 429
  Teradata options reference ........................................................... 434
  Table-structured file options reference ............................................. 437
  WebSphere Business Integration options reference ............................ 439
  Web services options reference ....................................................... 444
  XML options reference ................................................................. 450

Chapter 23. Views in the global catalog table containing federated information ................................................. 457

Chapter 24. Function mapping options for federated systems ................................................. 461

Chapter 25. Valid server types in SQL statements ................................................. 463

Chapter 26. Data type mappings ................................................. 465
  Default forward data type mappings ................................................. 465
  DB2 Database for Linux, UNIX, and Windows data sources ................. 465
  DB2 for System i data sources ......................................................... 466
  DB2 for VM and VSE data sources ................................................... 466
  DB2 for z/OS data sources ............................................................. 467
  Informix data sources ................................................................. 468
  Microsoft SQL Server data sources ............................................... 469
  ODBC data sources ................................................................. 470
  Oracle NET8 data sources ............................................................ 471
  Sybase data sources ................................................................. 472
  Teradata data sources ................................................................. 473
  Default reverse data type mappings ................................................. 474
  DB2 Database for Linux, UNIX, and Windows data sources ................. 475
  DB2 for System i data sources ......................................................... 475
  DB2 for VM and VSE data sources ................................................... 476
  DB2 for z/OS data sources ............................................................. 477
  Informix data sources ................................................................. 477
  Microsoft SQL Server data sources ............................................... 478
  ODBC data sources ................................................................. 479
  Oracle NET8 data sources ............................................................ 479
  Sybase data sources ................................................................. 480
  Teradata data sources ................................................................. 480
  Unicode default data type mappings ................................................. 481
  Unicode default forward data type mappings - Microsoft SQL Server wrapper ............................................... 481
  Unicode default reverse data type mappings - Microsoft SQL Server wrapper ............................................... 482
  Unicode default forward data type mappings - NET8 wrapper ............. 482
  Unicode default reverse data type mappings - NET8 wrapper ............. 482
  Unicode default forward data type mappings - ODBC wrapper .......... 483
  Unicode default reverse data type mappings - ODBC wrapper .......... 483
  Unicode default forward data type mappings - Sybase wrapper ......... 483
  Unicode default reverse data type mappings - Sybase wrapper ......... 484
  Data types supported for nonrelational data sources ........................ 484

Accessing information about the product ................................................. 489
  Providing comments on the documentation ........................................ 489

Accessible documentation ................................................. 491

Notices ................................................. 493
  Trademarks ................................................. 495

Index ................................................. 497
Chapter 1. Configuring the federated server

Before you configure data sources, configure the federated server and verify that it is properly set up.

Procedure

To configure the federated server, complete these tasks:

**Issuing commands in a federated system**

If you are new to federation, review how to access the DB2® Control Center and how to start the DB2 command line. You use these interfaces to perform many tasks in a federated system.

The Control Center provides dialog boxes and wizards that guide you through the completion of a task, without requiring that you know the DB2 commands required to perform the task. If you are new to DB2 and federation, you might want to use the Control Center, which is intuitive and easy to use. To use the DB2 command line, you enter the DB2 commands that are required to accomplish a task. If you are already familiar with DB2, you might prefer using the command line. When both interfaces are available to accomplish a task, the documentation describes the specific steps to take and the commands to enter.

To access an interface, do the following:

- To start the DB2 command line, enter `db2` at the operating system prompt. Then you can enter commands, one after the other. When you finish using the DB2 command line, you enter `quit`.
- To use the DB2 Control Center, expand the Federated Objects Folder to display the list of objects that you can work with. Then select an object and use the panels and wizards to accomplish the task.

**Verifying the setup of the federated server**

When you install federation, the software attempts to configure the federated server for you. To avoid problems when you configure data sources, verify that the federated server is properly set up.

To verify the setup of the federated server, complete these tasks:

**Enabling the federated server to access data sources**

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

The installation program attempts to set the FEDERATED parameter, but it is best to confirm that it is properly set to YES. If the FEDERATED parameter is not properly set, all attempts to access data sources will fail with message `SQL20076`, reason code 1.

Procedure

To verify that the FEDERATED parameter is properly set and enable it if necessary:
1. From the DB2 command line, issue this command to display the parameters and their current settings:
   
   ```sql
   GET DATABASE MANAGER CONFIGURATION
   ```

2. Check the value of the `MAX_CONNECTIONS` parameter to determine if the concentrator is on:
   - If `MAX_CONNECTIONS` is equal to `MAX_COORDAGENTS`, the concentrator is off.
   - If `MAX_CONNECTIONS` is greater than `MAX_COORDAGENTS`, the concentrator is on.

   The `MAX_CONNECTIONS` parameter cannot be active when the `FEDERATED` parameter is set to YES. If the concentrator is on, change the value of `MAX_CONNECTIONS` to be equal to `MAX_COORDAGENTS`.

3. Check the `FEDERATED` parameter setting. If the `FEDERATED` parameter is set to NO, change the setting to YES. Issue this command to change the setting:
   
   ```sql
   UPDATE DATABASE MANAGER CONFIGURATION USING FEDERATED YES
   ```

## Setting required environment variables for data source clients

Ensure that the required environment variables are set in the db2dj.ini file. This task is required for Informix®, Microsoft® SQL Server, Oracle, Sybase, and Teradata data sources, as well as for applications that use the ODBC wrapper.

### Before you begin

The system administrator performs this task.

If you install the client software before you install the federated server, the installation program automatically sets the required environment variables for the data sources that you select. If you install the client software after you install the federated server, upgrade to a new version of the client software, or migrate to new hardware, you must manually set the required environment variables.

This following table lists the required environment variables, by data source.

<table>
<thead>
<tr>
<th>Data source</th>
<th>Required variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informix</td>
<td>INFORMIXDIR, INFORMIXSERVER</td>
<td>For INFORMIXDIR, the directory containing the Informix client software. For INFORMIXSERVER, the name of the default Informix server.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>DJX_ODBC_LIBRARY_PATH</td>
<td>Directory containing the ODBC library.</td>
</tr>
<tr>
<td>ODBC</td>
<td>Varies, depending on the ODBC application</td>
<td>Varies, depending on the ODBC application.</td>
</tr>
<tr>
<td>Oracle</td>
<td>ORACLE_HOME</td>
<td>Directory containing the Oracle client software.</td>
</tr>
<tr>
<td>Sybase</td>
<td>SYBASE, SYBASE_OCS</td>
<td>For SYBASE, the directory containing the Sybase client software. For SYBASE_OCS, the directory, version, and release of the client software.</td>
</tr>
</tbody>
</table>
Table 1. Required environment variables, by data source (continued)

<table>
<thead>
<tr>
<th>Data source</th>
<th>Required variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata</td>
<td>COPYLIB</td>
<td>Directory containing the Teradata client software.</td>
</tr>
</tbody>
</table>

Procedure

To set the environment variables manually:

1. Open the db2dj.ini file.

   The db2dj.ini file is added to the federated server during installation. By default, the file is in the location that the DB registry variable DB2_DJ_INI specifies. If the DB2_DJ_INI registry variable is not set, the file is in these locations:
   - On UNIX®, the file is in instancehome/sqlib/cfg/db2dj.ini, where instancehome is the home directory of the instance owner.
   - On Microsoft Windows®, the file is in %DB2_PATH%\cfg\db2dj.ini, where %DB2_PATH% is the directory where the DB2 database system is installed. For example, C:\Program Files\IBM\sqlib.

2. Use the syntax variable_name=variable_value to set the value of the environment variable. If the variable_value is a file or directory name, you must specify the fully qualified path. For example, to set the INFORMIXDIR variable to the directory informix in the home directory /home/user1, include this entry in the db2dj.ini file:
   
   INFORMIXDIR=/home/user1/informix

3. From the DB2 command line, issue these commands:

   db2stop
   db2start
   
   You stop and then restart the database instance to ensure that the environment variables take effect.

4. If you are configuring a multi-partition system, copy the db2dj.ini file to each partition.

Restrictions for the db2dj.ini file

The db2dj.ini file contains the required and optional environment variables for a data source.

When you modify the db2dj.ini file, keep these restrictions in mind.

To specify each variable, use the format variable_name=variable_value

where

- variable_name is the name of the environment variable.
- variable_value is the value.

To specify a file name or directory name as the variable_value, specify the fully qualified name. The name cannot contain file name meta characters such as ~ (tilde) and environment variables, such as $HOME. For example, to set the INFORMIXDIR environment variable to the informix directory and the home directory that is named /home/user1, specify this value in the db2dj.ini file:

   INFORMIXDIR=/home/user1/informix

Keep these limits in mind:
The maximum length for the name of an environment variable is 255 bytes
The maximum length for the value of an environment variable is 756 bytes
The maximum length for a line in the db2dj.ini file is 1021 bytes; data beyond this length is ignored.

Verifying that library files are linked to data source client software

For best results, always verify that the library files are correctly linked to the data source client software before you use the CREATE WRAPPER command to register the wrapper. This task applies only when the federated server uses UNIX and the data source client is Informix, Microsoft SQL Server, Oracle, Sybase, or Teradata.

When you install federation, you can choose to configure wrappers automatically. If you install the data source client software on the federated server before you install federation on the server and choose to configure the wrapper automatically during the installation program, the wrapper library files are linked to the corresponding data source client software. If you install the data source client software after you install federation, for example to update to a new version of the client, you must manually link the wrapper library files to the data source client software.

These are the common causes of link failure:
• A required environment variable is not available during installation.
• The data source client is not installed on the federated server.
• The data source client that is installed on the federated server is not at a supported level.

Procedure

To verify that library files are correctly linked:
1. Open the message file for the data source. The message file is in the directory where the federated server is installed, in the lib32 or lib64 subdirectory. This table lists the data sources, message files, and library files.

Table 2. Data sources, message files, and library files

<table>
<thead>
<tr>
<th>Data source</th>
<th>Message file</th>
<th>Library files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informix</td>
<td>djxlinkInformix.out</td>
<td>AIX - libdb2informix.a AIX - libdb2informix.a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HP-UX - libdb2informix.so HP-UX - libdb2informix.so</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linux - libdb2informix.so Linux - libdb2informix.so</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Solaris - libdb2informix.so Solaris - libdb2informix.so</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>djxlinkMssql.out</td>
<td>AIX - libdb2mssql3.a AIX - libdb2mssql3.a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HP-UX - libdb2mssql3.so HP-UX - libdb2mssql3.so</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linux - libdb2mssql3.so Linux - libdb2mssql3.so</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Solaris - libdb2mssql3.so Solaris - libdb2mssql3.so</td>
</tr>
<tr>
<td>Oracle</td>
<td>djxlinkOracle.out</td>
<td>AIX - libdb2net8.a AIX - libdb2net8.a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HP-UX - libdb2net8.so HP-UX - libdb2net8.so</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linux - libdb2net8.so Linux - libdb2net8.so</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Solaris - libdb2net8.so Solaris - libdb2net8.so</td>
</tr>
<tr>
<td>Sybase</td>
<td>djxlinkSybase.out</td>
<td>AIX - libdb2ctlib.a AIX - libdb2ctlib.a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HP-UX - libdb2ctlib.so HP-UX - libdb2ctlib.so</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linux - libdb2ctlib.so Linux - libdb2ctlib.so</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Solaris - libdb2ctlib.so Solaris - libdb2ctlib.so</td>
</tr>
</tbody>
</table>
Table 2. Data sources, message files, and library files (continued)

<table>
<thead>
<tr>
<th>Data source</th>
<th>Message file</th>
<th>Library files</th>
</tr>
</thead>
</table>
| Teradata    | djxlinkTeradata.out | AIX - libdb2teradata.a  
                |               | HP-UX - libdb2teradata.so  
                |               | Linux - libdb2teradata.so  
                |               | Solaris - libdb2teradata.so |

2. Evaluate the contents of the message file:
   - If linking succeeded, the library file displays in the directory where federation is installed, and the message file displays a success message.
   - If linking failed, the message file displays an error message. If neither the library file nor the message file display in the directory where federation is installed, you must manually link the library file to the data source client software.

**Manually linking library files to data source client software**

If the library files do not display in the directory path, you must run a script to link them to the data source client software.

**Before you begin**
- The data source client software must be installed and configured on the federated server.
- By default, the scripts issue all messages in English. To issue messages in another language, one federated database must be configured to use the code page for the non-English language.
- The required data source environment variables must be set.

Table 3. Data sources, scripts, and required environment variables

<table>
<thead>
<tr>
<th>Data source</th>
<th>Script</th>
<th>Required environment variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informix</td>
<td>djxlinkInformix</td>
<td>INFORMIXDIR</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>djxlinkMssql</td>
<td>DJX_ODBC_LIBRARY_PATH</td>
</tr>
<tr>
<td>Oracle</td>
<td>djxlinkOracle</td>
<td>ORACLE_HOME</td>
</tr>
<tr>
<td>Sybase</td>
<td>djxlinkSybase</td>
<td>SYBASE, SYBASE_OCS</td>
</tr>
<tr>
<td>Teradata</td>
<td>djxlinkTeradata</td>
<td>COPLIB</td>
</tr>
</tbody>
</table>

**Procedure**

To link the library files to the data source client software:

1. Open a UNIX command prompt, and enter this command:
   ```
   cd /opt/IBM/db2/V9.5/bin
   ```
2. From the DB2 command line, enter these commands:
   ```
   db2 disconnect all
   db2stop
   ```
3. Run the script for each data source that you want to access. See the table above for the names of the scripts.
4. If you are installing a multi-partition environment, repeat Steps 1 - 3 on each database instance.
5. For each database instance, go to the directory `install_dir/instance`, where `install_dir` is the DB2 installation directory, and enter this command:
./db2updt instance_name
This command updates the configuration of the instance and gives it access to
the data sources.

6. For each database instance, display the file permissions on the library files.
Verify that the database instance owner has the permission to read and run the
files.

Managing the federated license key and license policy
Each computer on which federation is installed must store a license file that
specifies the product license key. The license policy controls and monitors the
number of users who are allowed to connect to the federated server.

The license key is located in the license directory of the installation software. If the
license key is not registered, you must manually register it.
1. Look for one of these files in the license directory of the installation software:
   
   - wsfs.lic - The full license
   - wsfs_t.lic - The try-and-buy license
2. If license key file is not in the directory, enter this command at a command
   prompt to register a license key:
   
   db2licm -a file_name

   where file_name is the mounted CD drive and location of the license file. For
   example, to register the full license on Microsoft Windows, enter this command:
   
   db2licm -a D:\license\wsfs.lic
3. Set the license policy, which controls and monitors the users who are allowed
to connect to the federated server.

Creating a federated database
Before you configure the federated server to access data sources, you must create a
federated database.

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

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

Procedure
To create the federated database:
1. Determine the code set and collating sequence that you want to specify when
   you create the federated database.
2. From the DB2 command line, issue the CREATE DATABASE command.
   For example, to create a database named federated that uses the code set
   ISO8859-15 in the territory of Brazil, issue this command:

   CREATE DATABASE federated
   USING CODESET ISO8859-15
   TERRITORY BR
Code sets, collating sequences, and national language support

When you create a federated database, you specify the code set, territory, and collating sequence. This information controls the language in which data is stored and the sequence in which character data is sorted.

A code set is a set of unique bit patterns that map to the characters of a specific natural language. IBM® products use the term code page as a synonym for code set. A territory identifies a locale and specifies region-specific information for the specified code set. If you do not specify these options, the database uses the language and collating sequence of the DB2 client that you use to create the database.

Before you create the federated database, determine which code set and territory values to specify. After you create the database, you cannot change these values. To choose a code set for the federated database, evaluate the code set specified by the remote data source that the federated database will access. Choose a code set for the federated database that corresponds to the code set that the remote data sources use. If the federated database will access multiple data sources, evaluate the code sets specified by all of the remote data sources. If the data sources use different code sets or incompatible code sets, specify Unicode as the code set for the federated server.

For many data sources, the first time that a wrapper connects to a data source, the wrapper performs these tasks:
1. Determines the code page and territory of the federated database.
2. Maps the code set and territory to a data source client locale, if the data source supports one.
3. Sets an environment variable, calls a data source API to tell the data source what the client locale is, or prepares to perform code set conversion.

Code page conversion involves converting character data between the code page of the data source database and the code page of the federated database. Some data sources perform code page conversion. For some data sources that do not perform code page conversion, the wrapper performs the conversion. For example, if the federated database uses code page 819, territory US, the equivalent Oracle client locale is American_America.WE8ISO8859P1. The Oracle wrapper automatically sets the NLS_LANG environment variable to the Oracle client locale value. Then when data is sent from the Oracle database to the wrapper, the Oracle database converts the data from code set American_America.WE8ISO8859P1 to code page 819. When data is sent from the wrapper to the Oracle database, the Oracle server or client converts the data from code page 819 to the code set that the Oracle database uses.

The collating sequence is related to the language that the federated server supports and that the data source server supports. To specify the collating sequence for the federated database, include the COLLATE USING option on the CREATE DATABASE command. If the federated database and the data source use the same collating sequence, set the COLLATING_SEQUENCE server option to ‘Y’ when you issue the CREATE SERVER statement. The collating sequence that you specify for the federated database affects where queries that involve character sorting or character comparisons are performed. By default, the federated database uses a case-sensitive collating sequence. However, some data sources use a case-insensitive collating sequence as the default. A data source might also allow the collating sequence to be customized or might offer multiple options for setting
the default code page. If the collating sequence of the federated database and the
data source are different, a query might not return the results that you expect. For
example, if the query involves character-sorting, the correct results are returned,
but they are not in the order that you expect. If the query involves character
comparisons, incorrect results might returned.

Where character-based operations are performed has an effect on query
performance. When the collating sequences differ, the federated server performs
character sorting and character comparisons locally to ensure that consistently
sorted result sets are returned. For best results, set the federated database collating
sequence to the same sequence that the data source uses. Then, where possible, the
query optimizer pushes down character-based operations to the data source, so
that the data source, not the federated database, performs the operations.

**Unicode support for federated systems**
All relational and nonrelational wrappers and user-defined functions can run on a
database in the Unicode code page (UTF-8).

The database in the Unicode code page provides federated server environments
that are platform independent. The database can manipulate data that is stored in
various code pages on different data sources.

In Figure 1 on page 9 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.
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

To specify the client code page, issue a CREATE SERVER statement with the CODEPAGE option set to the value of the client code page.

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');
```
Example: If the data source is Microsoft SQL Server and the federated server is running on UNIX and the 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:

```sql
CREATE SERVER FEDSERVERU TYPE MSSQLSERVER VERSION 2000 WRAPPER MSSQLODBC3 OPTIONS(NODE 'SAMPLE', DBNAME 'TESTDB', CODEPAGE '1208');
```

Supported Unicode code pages for the MSSQL and ODBC wrapper CODEPAGE option

Valid code page values are those that DB2 Database for Linux®, UNIX, and Windows supports plus those shown in the following table.

<table>
<thead>
<tr>
<th>CODEPAGE option value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1200</td>
<td>Codepage1200 - UCS-2 (big-endian)</td>
</tr>
<tr>
<td>1202</td>
<td>Codepage1202 - UCS-2 (little-endian)</td>
</tr>
<tr>
<td>1208</td>
<td>Codepage1208 - UTF-8</td>
</tr>
<tr>
<td>1232</td>
<td>Codepage1232 - UTF-32 (big-endian)</td>
</tr>
<tr>
<td>1234</td>
<td>Codepage1234 - UTF-32 (little-endian)</td>
</tr>
</tbody>
</table>

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.

About this task

Valid values are those that DB2 Database for Linux, UNIX, and Windows supports. The default value is the code page of the federated database.

Procedure

To specify the code page of a table-structured file, issue the CREATE NICKNAME statement with the CODEPAGE option set to the code page of the data in the table-structured file.

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:

```sql
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');
```
Specifying the file code page for Unicode support of table-structured file data sources - example

The following example shows you how to specify the code page of a table-structured file.

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),
    Manufactuer 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');
```

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 truncated data returned or insertion or update failures.

When you select data from the remote data source, that data is truncated if the character string conversion results in a larger number of bytes than the size of the nickname column. If the truncated data ends in a dangling character, blanks fill the remaining bytes. Also, you can insert or update data that is larger than the nickname column size if the converted data size 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 or update of data can fail. Insertions or updates fail if the character string conversion results in a larger number of bytes than the size of the remote data source column.

Adjust nickname column size or remote table column size to prevent data truncation or a truncation error.

Configuring the federated server to access data sources

After you install and set up the federated server and create a federated database, plan and then configure access to the data sources.

The process for configuring access to data sources is the same, regardless of the data source. What differs in the process are the particular settings that you apply as you complete each configuration task for each data source. The process described here is generic; for complete information on configuring a particular data source, see the detailed configuration information for the data source.

Review how to name and specify objects

During the configuration process, you use DB2 SQL statements to register objects. Before you perform any configuration tasks, be sure that you understand the federated naming rules and understand how the use of quotation marks in SQL statements affects the case-sensitivity of the objects that you specify.

Register the wrapper

After you use the federation installation program to install support for a data source, you must register the corresponding wrapper. A wrapper is a set of library
files that the federated server uses to communicate with the data source and to retrieve data from it. For each type of data source that you want to access, you register one wrapper. For example, to access one table in DB2 for Linux, UNIX, and Windows, one table in DB2 for iSeries™, and one table in Teradata, you register two wrappers: the DRDA® wrapper for the DB2 data sources and the Teradata wrapper for Teradata data sources.

As part of your planning, decide whether or not to use the default wrapper name or assign a different name to the wrapper, and review the wrapper options that are available for each data source that you are configuring. Each data source has one or more required wrapper options that you must set.

**Register the server definitions**

Before you can access specific data source objects, you register one or more server definitions. For a relational data source, a server definition represents a remote database, a database partition, or a node. For a nonrelational data source, a server definition often maps to other types of external data objects. Each data source has required and optional parameters that you must specify when you register the server definition.

As part of your planning, review the server options that are available for the specific data source that you are configuring. Each data source has one or more required server options that you must set.

**Register user mappings**

If a remote data source requires user authentication and if a user’s remote user ID and remote password are different from the ones that the user uses to connect to the federated database, you define a user mapping. A user mapping is an association between a federation server authorization ID and a data source user ID and password. By default, user mappings are stored in the catalog on the federated server.

As part of your planning, decide if you want to store user mapping information in an external repository, such as on an LDAP server or in a file. To use an external repository, you must create a plug-in that provides the federated server with the interface to the repository.

**Update data source statistics**

For each relational data source that you plan to access, use a command that is equivalent to the DB2 RUNSTATS command to update the statistics at the remote data source. Then when you create nicknames, the most up-to-date statistical information is added to the system catalog in the federated database. Later when you run a query on the data source, the query optimizer uses this information to determine the most efficient way to perform the query.

After you create nicknames, statistics at the data source might change. When statistics for a relational data source change, use the SYSPROC.NNSTAT stored procedure to update the statistical information in the system catalog. When statistics for a nonrelational data source change, use the tool that is provided by the nonrelational data source, or manually update the statistics in the SYSTAT catalog views.
Register nicknames

You create a nickname for each relational data source object that you want to access. For some nonrelational data sources, you define a fixed list of input and output columns when you register the nickname. Each column that you specify is mapped to a particular field, column, or element in the data source object.

As part of your planning, review the nickname and column options that are available for the data source that you are configuring. Some data sources have required nickname and column options that you must set.

Perform additional configuration tasks

Depending on how you want to work with the data source, you might want to perform these additional configuration tasks.

Create index specifications

You can define an index specification for objects that do not have an index. For example, you create an index specification when a table acquires a new index or if the data source object, such as a view, does not have an index.

Define alternative data type mappings

In the federated system, there are default mappings between the data source data types and the federated database data types. For relational data sources, you can define alternative data type mappings. For example, you can change a type mapping for all data source objects located on a specific server or change a type mapping for a specific data source object, data source type, or data source object and type.

Define alternative function mappings

In the federated system, there are default function mappings between the built-in data source functions and the built-in federated database functions. For relational data sources, you can define alternative function mappings. For example, you can define an alternative function mapping when you want to use a new built-in function or a user-defined function that is available at the data source but for which the federated database lacks a mapped function.

Case sensitivity and the correct use of quotation marks

When you specify option values and objects in DB2 SQL statements, you need to know when quotation marks are required, which type to use, and how they affect cases sensitivity.

How you name objects when you first create them affects the case of the characters in the object name and determines how you specify object names and option values in commands. For example, if, when you create a nickname, you do not enclose the name in double quotation marks, the system catalog stores the nickname in uppercase characters, regardless of the case of the characters that you use to name the object. If you use double quotation marks when you create the nickname, the catalog stores the characters of the object name in exactly the case that you specify. Then, when you use the object name as an option value, you must specify exactly that case. For example, the FOREIGN_KEY column option that the Script, Web services, and XML wrappers support requires that you specify the nickname for the foreign key column as the option value. When you enter the option value, you must use the same case that the federated server catalog uses to store the nickname.
The following table describes the correct use of case and quotation marks when you specify option values and objects in DB2 SQL statements.

Table 5. Correct use of case and quotation marks

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Case and quotation mark use</th>
<th>Examples</th>
</tr>
</thead>
</table>
| Option value                     | Use the case that the option value requires, and enclose the option value in single quotation marks.               | This statement creates a data source table named remote_schema.remote_table (all lowercase):
|                                  |                                                                  | CREATE TABLE newton.my_nick (c1 int) OPTIONS (remote_server 'MY_SERVER'
|                                  |                                                                  | remote_schema 'remote_schema',
|                                  |                                                                  | remote_tabname 'remote_table');                                              |
|                                  |                                                                  | This statement creates a data source table named REMOTE_SCHEMA.REMOTE_TABLE (all uppercase):
|                                  |                                                                  | CREATE TABLE newton.my_nick (c1 int) OPTIONS (remote_server'MY_SERVER'
|                                  |                                                                  | remote_schema 'REMOTE_SCHEMA',
|                                  |                                                                  | remote_tabname 'REMOTE_TABLE');                                             |
| Object that contains only lower- | Use all lowercase characters, and enclose in the identifier double quotation marks.                             | This statement creates a nickname on a data source table named infx_user.remote_table (all lowercase):
| case characters                  |                                                                  | CREATE NICKNAME my_nick FOR infx_server.
|                                  |                                                                  | "infx_user"."remote_table";                                                   |
|                                  |                                                                  | Note: Some data sources such as Informix and Teradata use lowercase names by default.                                                                                                               |
| Object that contains only upper- | There are two choices:                                          | Each of these statements creates the nickname MY_NICK (all uppercase):
| case characters, numbers, and    | • Use all uppercase characters, and enclose the identifier in double quotation marks.                           | CREATE NICKNAME my_nick FOR infx_server.
| underscore characters (_)        | • Use any case and do not enclose the identifier in double quotation marks.                                      | "infx_user"."remote_table";
|                                  |                                                                  | CREATE NICKNAME "MY_NICK" FOR infx_server.
|                                  |                                                                  | "infx_user"."remote_table";                                                   |

For data source authorization IDs and passwords, you can also use the server options FOLD_ID and FOLD_PW to convert the ID and password to the correct case.

**From a UNIX operating system command prompt**

If you enclose a case-sensitive value in quotation marks at the UNIX command prompt on the federated server, you must ensure that the quotation marks are parsed correctly:
SQL statements that contain double quotation marks, but that do not contain single quotation marks

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

For example, if you want to issue this SQL statement:
CREATE NICKNAME my_nickname FOR my_server."owner".my_table"

You enter the following text at the UNIX command prompt:
DB2 'CREATE NICKNAME my_nickname FOR my_server."owner".my_table"

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

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

For example, if you want to issue this SQL statement:
CREATE USER MAPPING FOR USER SERVER my_server
   OPTIONS(REMOTE_AUTHID 'my_id', REMOTE_PASSWORD 'my_password')

You enter the following text at the UNIX command prompt:
DB2 "CREATE USER MAPPING FOR USER SERVER my_server
   OPTIONS(REMOTE_AUTHID 'my_id', REMOTE_PASSWORD 'my_password')"

SQL statements that contain both double and single quotation marks

If the SQL statement contains both single and double quotation marks:
- Enclose the entire statement in double quotation marks
- Precede each double quotation mark in the statement with a backward slash.

For example, to issue this SQL statement:
CREATE USER MAPPING FOR "local_id" SERVER my_server
   OPTIONS(REMOTE_AUTHID 'my_id', REMOTE_PASSWORD 'my_password')

You enter the following text at the UNIX command prompt:
DB2 "CREATE USER MAPPING FOR "local_id" SERVER my_server
   OPTIONS(REMOTE_AUTHID 'my_id', REMOTE_PASSWORD 'my_password')"

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

From a Windows operating system command prompt

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

At the Windows command prompt on the federated server, you type:
DB2 CREATE NICKNAME nick1
   FOR NORBASE."my_schema"."weekly_salary"
From the DB2 command line or from an application

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

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

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

Configuring multiple federated servers to access data sources

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

Before you begin

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

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

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

Procedure

To configure multiple federated servers to access data sources:

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

You must change the user mappings to specify the proper passwords for the federated server. When the DDL for the CREATE USER MAPPING statements is captured in the **Action Output** window, the passwords are masked by asterisks. You must replace the asterisks with the proper passwords.

6. After you change the DDL statements that were generated in the **Action Output** window, run the DDL statements on the next federated server.
Chapter 2. Configuring access to BioRS data sources

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

Procedure

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

BioRS wrapper

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

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

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

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

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

BioRS provides a Web-based interface that enables users to run queries on the data in BioRS databanks. The BioRS wrapper uses the same application programming interfaces (APIs) as the BioRS Web-based interface to run queries.
From the client, users or applications submit a query using SQL statements. Then, the query is sent to your federated system where the BioRS wrapper is installed. Depending on how the query is constructed, both the federated server and your BioRS server might be used to process the query. The BioRS server can be on a different computer from the federated system. Authentication information must be provided by the federated system to the BioRS server for each query. This information can be either a user ID and password combination, or an unauthenticated indication (usually a guest account).

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

**Adding BioRS data sources to a federated server**

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

**Before you begin**
- Federation must be installed on a server that will act as the federated server.
- A database must exist on the federated server.

**About this task**

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

**Procedure**

To add BioRS data sources to a federated server:

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

Registering the custom functions for the BioRS wrapper

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

About this task

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

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

Procedure

To register the BioRS custom functions:

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

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

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

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

Custom functions for the BioRS wrapper

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

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

The custom functions for the BioRS wrapper are:

- BIORs.CONTAINS
- BIORs.CONTAINS_LE
• BIOR.S.CONTAINS_GE
• BIOR.S.SEARCH_TERM

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

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

Table 6. Custom functions for the BioRS wrapper

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

Registering the BioRS wrapper

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

If you use a proxy server to access BioRS files, you must specify the proxy information as options when you register the wrapper. By default, wrapper options are used when you query any BioRS file. However, if you also specify proxy options when you register a server definition, the server options take precedence.

Procedure

To register the BioRS wrapper:

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
</tbody>
</table>
Method | Procedure
--- | ---
From the command line | Issue the CREATE WRAPPER statement.
CREATE WRAPPER wrapper_name
LIBRARY library_name;

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

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

**BioRS wrapper library files**
The BioRS wrapper library files are added to the federated server when you install the federated server.

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

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

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

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Wrapper library file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/&lt;install_path&gt;/lib32/</td>
<td>libdb2lsbiors.a</td>
</tr>
<tr>
<td></td>
<td>/usr/opt/&lt;install_path&gt;/lib64/</td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib32</td>
<td>libdb2lsbiors.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib64</td>
<td></td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib32</td>
<td>libdb2lsbiors.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib64</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>%DB2PATH%\bin</td>
<td>db2lsbiors.dll</td>
</tr>
</tbody>
</table>

<install_path> is the directory path where the federated server is installed on UNIX or Linux.

%DB2PATH% is the environment variable that is used to specify the directory path where the federated server is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.
CREATE_WRAPPER statement - Examples for the BioRS wrapper

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

Registering a wrapper

If you are not using a proxy server to access BioRS documents, you issue this statement to register the wrapper:

CREATE.WRAPPER biors_wrapper LIBRARY 'libdb2lsbiors.a';

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

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

Registering a wrapper for an HTTP proxy server

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

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

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

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

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

Registering a wrapper for a SOCKS proxy server

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

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

LIBRARY 'libdb2lsbiors.so'
   The name of the wrapper library file for federated servers that use the Linux and Solaris operating systems.

PROXY_TYPE 'SOCKS'
   Specifies the proxy type that is used to access the Internet when the federated server is behind a firewall. The valid values are ‘NONE’, ‘HTTP’, or ‘SOCKS’.
PROXY_SERVER_NAME 'proxy_socks'
Specifies the proxy server name or IP address. This option is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

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

Registering the server definition for a BioRS data source
You must register each BioRS server that you want to access in the federated database.

If you use a proxy server to access BioRS files, you can specify the proxy information as options when you register the server definition. If you specify the proxy information in the server definition, the server options take precedence over the options that you specified when you registered the wrapper.

Procedure
To register a server definition for a BioRS data source:
Choose the method that you want to use to register the server definition:

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>From the command line</td>
<td>Issue the CREATE SERVER statement.</td>
</tr>
<tr>
<td></td>
<td>CREATE_SERVER server_definition_name</td>
</tr>
<tr>
<td></td>
<td>VERSION version_number</td>
</tr>
<tr>
<td></td>
<td>WRAPPER wrapper_name</td>
</tr>
<tr>
<td></td>
<td>OPTIONS (NODE 'node_name');</td>
</tr>
<tr>
<td></td>
<td>If you use a proxy server to access BioRS files, you must specify several options when you register the BioRS wrapper or the server definition. To specify the proxy server information when you register the BioRS wrapper, you issue this statement:</td>
</tr>
<tr>
<td></td>
<td>CREATE_SERVER server_definition_name</td>
</tr>
<tr>
<td></td>
<td>VERSION version_number</td>
</tr>
<tr>
<td></td>
<td>WRAPPER wrapper_name</td>
</tr>
<tr>
<td></td>
<td>OPTIONS (PROXY_TYPE 'type', PROXY_SERVER_NAME 'server_name', PROXY_SERVER_PORT 'port_number');</td>
</tr>
</tbody>
</table>

When you register the server definition, you specify server options in the CREATE SERVER statement. There are required server options and optional server options. The NODE server option is required.

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

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

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

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

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

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

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

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

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

Registering a server definition using optional parameters and server options

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

```
CREATE SERVER biors_server TYPE BioRS VERSION '5.2'
  WRAPPER biors_wrapper
  OPTIONS (NODE 'biors_server2.com', PORT '5555', TIMEOUT 30,
    CASESENSITIVE 'N');
```

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

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

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

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

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

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

**Registering a server definition that includes a proxy server**

```plaintext
CREATE SERVER biors_proxy_serv VERSION 5.2 WRAPPER biors_proxy
  OPTIONS (NODE 'biors.mysite.com',
           PORT '5555',
           PROXY_TYPE 'HTTP',
           PROXY_SERVER_NAME 'proxy.mysite.com',
           PROXY_SERVER_PORT '81')

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

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

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

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

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

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

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

PROXY_PASSWORD 'bargle'
  Specifies the password on the proxy server that is associated with the user name 'argle'. This server option is required when the value for the PROXY_TYPE server option is 'SOCKS'.
```
Creating the user mappings for a BioRS data source

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

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

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

Before you begin

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

- If your BioRS server is configured for guest access for all user accounts, you do not need to create user mappings. The federated server uses a guest account to access the BioRS server.
- If you have an external repository, such as LDAP, to store the user mappings, you do not need to create user mappings. You must specify the DB2_UM_PLUGIN option on the BioRS wrapper. You can specify this option when you register or alter the wrapper. The schema in the external repository must include guest access.
- If your BioRS server is configured to authenticate user accounts with IDs and passwords, you must create user mappings in the federated database for any user accounts that will use the BioRS wrapper to access BioRS data sources.
- If your BioRS server is configured to use a mixture of guest and authenticated user accounts, you must create user mappings for the authenticated user accounts that will use the BioRS wrapper to access BioRS data sources.

About this task

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

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

Procedure

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

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the <strong>Federated Database Objects</strong> folder and click <strong>Create Federated Objects</strong>.</td>
</tr>
</tbody>
</table>
Issue the `CREATE USER MAPPING` statement. For example:

```sql
CREATE USER MAPPING FOR local_userID
SERVER server_definition_name
OPTIONS (REMOTE_AUTHID 'remote_userID',
REMOTE_PASSWORD 'remote_password',
PROXY_AUTHID 'proxy_server_userID',
PROXY_PASSWORD 'proxy_server_password');
```

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

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

**CREATE USER MAPPING statement - Examples for the BioRS wrapper**

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

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

**Creating a user mapping for a guest user account**

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

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

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

*charlie* Specifies the local authorization ID that you are mapping to a remote user ID, which is defined at the BioRS server.

*SERVER biors_server*

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

*GUEST 'Y'*

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

**Creating a user mapping for an authenticated user account**

The following example shows how to map a federated server user ID to a BioRS server user ID and password:
CREATE USER MAPPING FOR charlie SERVER biors_server
    OPTIONS (REMOTE_AUTHID 'charlene',
             REMOTE_PASSWORD 'all4one');

`charlie` Specifies the local authorization ID that you are mapping to a remote user ID and password, which are defined at the BioRS server.

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

REMOTE_AUTHID 'charlene'
    Specifies the user ID at the BioRS server to which you are mapping `charlie`. This remote ID must be in a format that is expected by the BioRS server.

REMOTE_PASSWORD 'all4one'
    Specifies the password that is associated with `charlene`.

Creating a user mapping by using a special register

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

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

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

Creating a user mapping for a proxy server

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

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

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

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

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

Registering nicknames for BioRS data sources

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

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

Restrictions

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

About this task

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

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

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

Procedure

To register a nickname for a BioRS databank:

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard.</td>
</tr>
<tr>
<td></td>
<td>Right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Using the command line</td>
<td>Issue the CREATE NICKNAME statement.</td>
</tr>
<tr>
<td></td>
<td>For example: CREATE NICKNAME nickname (</td>
</tr>
<tr>
<td></td>
<td>column_name data_type OPTIONS (nickname_column_options),</td>
</tr>
<tr>
<td></td>
<td>column_name data_type OPTIONS (nickname_column_options),</td>
</tr>
<tr>
<td></td>
<td>column_name data_type OPTIONS (nickname_column_options)</td>
</tr>
<tr>
<td></td>
<td>) FOR SERVER server_definition_name OPTIONS (nickname_options);</td>
</tr>
</tbody>
</table>

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

Repeat this step for each BioRS databank that you want to create a nickname for.
CREATE NICKNAME statement - examples for the BioRS wrapper

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

Creating simple, but limited, nicknames for databanks

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

For example:

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

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

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

1. You are limited to one family of nicknames for each federated database schema. For example, you have two databanks that have a parent-child relationship. The databanks are SWISSPROT and SPFEAT. These databanks form a family. If you use the default syntax for the CREATE NICKNAME statement, you will have the (SWISSPROT nickname for the SWISSPROT databank and the SPFEAT nickname for the SPFEAT databank. To have more than one nickname for SWISSPROT in the schema, you must use the REMOTE_OBJECT nickname option when you register the nickname.

2. You are limited to databanks whose names can be used as nickname names. The databank name must conform to the syntax that is supported by the federated server. For example, if the databank name includes a period or space, you must use the REMOTE_OBJECT nickname option when you register the nickname.

Creating multiple nicknames for the same databank

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

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

```
CREATE NICKNAME NewSP
   (ID VARCHAR(32) OPTIONS (ELEMENT_NAME '_ID_'),
    ALLTEXT VARCHAR(128),
    ENTRYDATE VARCHAR(64))
FOR SERVER biors_server
   OPTIONS (REMOTE_OBJECT 'SwissProt');
```
Creating nicknames for databanks that do not conform to federated syntax

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

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

SwissFT

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

**ID VARCHAR(32) OPTIONS (ELEMENT_NAME '_ID_')**
The name and data type for a table column. The ELEMENT_NAME column option is specified for the ID column.

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

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

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

**ALLTEXT VARCHAR(128)**
The name and data type for a table column.

**ENTRYDATE VARCHAR(64)**
The name and data type for a table column.

**FtLength VARCHAR(16)**
The name and data type for a table column.

**SERVER biors1**
The name that you assigned to the BioRS server in the CREATE SERVER statement.

**OPTIONS (REMOTE_OBJECT 'SwissProt.Features')**
Specifies the name of the BioRS databank that is associated with the nickname. This name determines the schema and the BioRS databank for the nickname. This name also specifies the relationship of the nickname to other nicknames.
The case sensitivity of this name depends on the case sensitivity of the BioRS server and on the value of the CASE_SENSITIVE server option.

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

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

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

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

**Creating nicknames for a databank linked to another BioRS databank**

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

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

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

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

**Custom functions and BioRS queries**

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

All of the custom functions for the BioRS wrapper must be registered with the schema name B10RS. You must include the B10RS schema whenever you use the functions.

The custom functions for the BIORS wrapper are:

- BIORS.CONTAINS
- BIORS.CONTAINS.LE
- BIORS.CONTAINS.GE
- BIORS.SEARCH_TERM

**The BioRS CONTAINS functions**

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

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

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

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

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

The case of the query term argument does not matter.

**Valid data types**

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

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

<table>
<thead>
<tr>
<th>Data type of search term column</th>
<th>Valid query term</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>VARCHAR() or CHAR()</td>
<td>BioRS text term, including wildcards.</td>
</tr>
</tbody>
</table>
Table 8. BioRS data types and valid custom function query terms (continued)

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

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

Using wildcard characters

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

The ? wildcard matches a single character. For example, the predicate BIOR.S.CONTAINS (description, ‘bacteri?’)=1 matches the term bacteria but not the term bacterial.

The * wildcard character matches zero or more characters. For example, the predicate BIOR.S.CONTAINS (description, ‘bacteri*’) =1 matches the terms bacteri, bacteria, and bacterial.

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

Specifying BioRS CONTAINS functions in queries

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

The BIOR.S.CONTAINS_GE and BIOR.S.CONTAINS_LE custom functions can be specified only for columns whose underlying BioRS data type is Number or Date. The BIOR.S.CONTAINS_GE function selects rows where the column contains a value that is greater than or equal to the value that is represented by the query term argument. The BIOR.S.CONTAINS_LE function selects rows where the column contains a value that is less than or equal to the value that is represented by the query term argument.
The BIORs.CONTAINS, BIORs.CONTAINS_GE, and BIORs.CONTAINS_LE functions return an integer result. When any of the three CONTAINS functions are used in a predicate, the return value must be compared to the value 1 using the = or <> operators. For example:

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

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

**The BIORs SEARCH_TERM function**

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

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

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

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

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

The case of the query term argument does not matter.

**Specifying BIORs SEARCH_TERM function in queries**

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

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

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

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

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

**Equality operations in BIORs queries**

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

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

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

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

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

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

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

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

**Case-insensitive matching**

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

- '100k_rat'
- '100K_RAT'

**Wildcard matching**

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

**Databank prefixing**

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

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

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

The local table, w46851_t1, contains the following values:

<table>
<thead>
<tr>
<th>ID</th>
<th>C1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-------------------</td>
<td>----</td>
</tr>
<tr>
<td>swissprot:100K_RAT</td>
<td>0</td>
</tr>
<tr>
<td>swissprot:RAT</td>
<td>1</td>
</tr>
<tr>
<td>swissprot:100K_R</td>
<td>2</td>
</tr>
<tr>
<td>swissprot:100K_R*</td>
<td>3</td>
</tr>
<tr>
<td>swissprot:100k_rat</td>
<td>4</td>
</tr>
<tr>
<td>100K_RAT</td>
<td>100</td>
</tr>
<tr>
<td>RAT</td>
<td>101</td>
</tr>
<tr>
<td>100K_R</td>
<td>102</td>
</tr>
<tr>
<td>100K_R*</td>
<td>103</td>
</tr>
<tr>
<td>100k_rat</td>
<td>104</td>
</tr>
</tbody>
</table>

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

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

The following results are returned:
However, the expected behavior is that only row 0 would be returned.

**Equijoin predicates for the BioRS wrapper**

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

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

**Example for nickname definitions and an equijoin query**

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

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

**Nickname definition for the SwissProt databank**

The SwissProt databank is the parent databank.

```
CREATE NICKNAME tc600sprot ( 
    ID VARCHAR (32) OPTIONS (ELEMENT_NAME '_ID_'),
    AllText VARCHAR (128),
    EntryDate VARCHAR (128),
    Update VARCHAR (128),
    Description VARCHAR (128),
    Crossreference VARCHAR (32),
    Authors VARCHAR (256),
    Journal VARCHAR (256),
    JournalIssue VARCHAR (64) OPTIONS (IS_INDEXED 'N'),
    PublicationYear VARCHAR (1024),
    Gene VARCHAR (20) OPTIONS (IS_INDEXED 'Y'),
    Remarks VARCHAR (1200),
    RemarkType CHAR (20),
    CatalyticActivity VARCHAR (20),
    CoFactor VARCHAR (64),
    Disease VARCHAR (128),
    Function VARCHAR (128),
    Pathway VARCHAR (128),
    Similarity VARCHAR (128),
    Complex VARCHAR (64),
    FtKey VARCHAR (32),
    FtDescription VARCHAR (128),
```

Chapter 2. Configuring access to BioRS data sources 39
Nickname definition for the SwissProt.features databank

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

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

The query that references both nicknames in an equijoin

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

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

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

The BioRS AllText element

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

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

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

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

**BioRS data source - Example queries**

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

These examples show you how to:

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

The sample queries use the nicknames swiss and swissft.

**CREATE NICKNAME statement for the swiss nickname**

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

```sql
CREATE NICKNAME swiss
(
  ID CHAR (30) OPTIONS (ELEMENT_NAME '_ID_'),
  EntryDate VARCHAR (15),
  Update CLOB (15),
  Description CLOB (15),
  Crossreference CLOB (15),
  Authors CLOB (15),
  Journal VARCHAR (15),
  JournalIssue VARCHAR (15),
  PublicationYear CLOB (15),
  PublicationTitle CLOB (15),
  Gene CLOB (15),
  Remarks CLOB (15),
  RemarkType VARCHAR (15),
  CatalyticActivity VARCHAR (15),
  Cofactor VARCHAR (15),
  Disease VARCHAR (15),
  Function CLOB (15),
  Pathway VARCHAR (15),
  Similarity CLOB (15),
  Complex VARCHAR (15),
  FtKey VARCHAR (15),
  FtDescription CLOB (15),
  FtLength VARCHAR (15),
  MolWeight CHAR (15),
  Protein_Length VARCHAR (15),
  Sequence CLOB (15),
  AccNumber VARCHAR (15),
  Taxonomy CLOB (15),
  Organelle VARCHAR (15),
  Organism VARCHAR (15),
  Keywords VARCHAR (15),
  Localization VARCHAR (15),
  FtKey_count VARCHAR (15),
);```
CREATE NICKNAME statement for the swissft nickname

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

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

Query structures impact federated server performance

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

<table>
<thead>
<tr>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
</table>
| `SELECT s.id FROM Swiss s WHERE BIORs.CONTAINS(s.id, '100K_RAT') = 1 FETCH FIRST 3 ROWS ONLY;` | ID --------------
|                                                                     | 100K_RAT
|                                                                     | 1 record(s) selected.                      |
| `SELECT s.id FROM Swiss s WHERE s.id LIKE '%100K_RAT%' FETCH FIRST 3 ROWS ONLY;` | ID --------------
|                                                                     | 100K_RAT
|                                                                     | 1 record(s) selected.                      |

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

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

Queries that use wildcards in the BIORs.CONTAINS custom function

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

**Table 10. Sample queries that use wildcards in the BIOR.S.CONTAINS custom function**

<table>
<thead>
<tr>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
</table>
| SELECT s.crossreference FROM Swiss s WHERE BIOR.S.CONTAINS(s.crossreference, 'MEDLINE') = 1 FETCH FIRST 3 ROWS ONLY; | CROSSREFERENCE
|                                                                       | NCBI_TaxID=1011
|                                                                       | NCBI_TaxID=5875
|                                                                       | NCBI_TaxID=4081
|                                                                       | 3 record(s) selected.                                                   |
| SELECT s.crossreference FROM Swiss s WHERE BIOR.S.CONTAINS(s.crossreference, '?ED?IN?') = 1 FETCH FIRST 3 ROWS ONLY; | CROSSREFERENCE
|                                                                       | NCBI_TaxID=1011
|                                                                       | NCBI_TaxID=5875
|                                                                       | NCBI_TaxID=4081
|                                                                       | 3 record(s) selected.                                                   |
| SELECT s.crossreference FROM Swiss s WHERE BIOR.S.CONTAINS(s.crossreference, '*D*N*') = 1 FETCH FIRST 3 ROWS ONLY; | CROSSREFERENCE
|                                                                       | NCBI_TaxID=1011
|                                                                       | NCBI_TaxID=5875
|                                                                       | NCBI_TaxID=4081
|                                                                       | 3 record(s) selected.                                                   |

**Queries that access BioRS Author data type columns**

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

**Table 11. Sample queries that access BioRS Author data type columns**

<table>
<thead>
<tr>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
</table>
| SELECT s.authors FROM Swiss s WHERE BIOR.S.CONTAINS(s.authors, 'Mueller') = 1 FETCH FIRST 3 ROWS ONLY; | AUTHORS
|                                                                       | Mueller D. Rehb
|                                                                       | Mayer K.F.X. Sc
|                                                                       | Zemmour J. Litt
|                                                                       | 3 record(s) selected.                                                   |
| SELECT s.authors FROM Swiss s WHERE BIOR.S.CONTAINS(s.authors, 'Mueller,D') = 1 FETCH FIRST 3 ROWS ONLY; | AUTHORS
|                                                                       | 0 record(s) selected.                                                   |
| SELECT s.authors FROM Swiss s WHERE BIOR.S.CONTAINS(s.authors, 'Mueller ,D') = 1 FETCH FIRST 3 ROWS ONLY; | AUTHORS
|                                                                       | 0 record(s) selected.                                                   |
Table 11. Sample queries that access BioRS Author data type columns (continued)

<table>
<thead>
<tr>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT s.authors FROM Swiss s WHERE BIOR.S.CONTAINS(s.authors, 'Mueller, D') = 1 FETCH FIRST 3 ROWS ONLY;</td>
<td>AUTHORS</td>
</tr>
<tr>
<td></td>
<td>Mueller D. Rehb</td>
</tr>
<tr>
<td></td>
<td>Zou P.J. Borovo</td>
</tr>
<tr>
<td></td>
<td>Davies J.D. Mue</td>
</tr>
<tr>
<td></td>
<td>3 record(s) selected.</td>
</tr>
</tbody>
</table>

Queries that access BioRS Date data type columns

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

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

Table 12. Sample queries that access BioRS Date data type columns

<table>
<thead>
<tr>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT e.entrydate FROM embl e WHERE BIOR.S.CONTAINS(e.entrydate, date('11/01/1997')) = 1 FETCH FIRST 3 ROWS ONLY;</td>
<td>ENTRYDATE</td>
</tr>
<tr>
<td></td>
<td>01-NOV-1997</td>
</tr>
<tr>
<td></td>
<td>01-NOV-1997</td>
</tr>
<tr>
<td></td>
<td>01-NOV-1997</td>
</tr>
<tr>
<td></td>
<td>3 record(s) selected.</td>
</tr>
</tbody>
</table>

| SELECT g.update FROM gen g WHERE BIOR.S.CONTAINS(g.update, date('11/01/1997')) = 1 FETCH FIRST 3 ROWS ONLY; | UPDATE              |
|                                                                                         | 01-NOV-1997 11-     |
|                                                                                         | 01-NOV-1997 12-     |
|                                                                                         | 01-NOV-1997 06-     |
|                                                                                         | 3 record(s) selected.|

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

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

Table 13. Sample queries that use the BIORS.CONTAINS_LE and BIORS.CONTAINS_GE custom functions

<table>
<thead>
<tr>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT s.molweight FROM Swiss s WHERE BIOR.S.CONTAINS_LE(s.molweight, 100368) = 1 FETCH FIRST 3 ROWS ONLY;</td>
<td>MOLWEIGHT</td>
</tr>
<tr>
<td></td>
<td>100368</td>
</tr>
<tr>
<td></td>
<td>10576</td>
</tr>
<tr>
<td></td>
<td>8523</td>
</tr>
<tr>
<td></td>
<td>3 record(s) selected.</td>
</tr>
</tbody>
</table>

44 Configuration Guide for Federated Data Sources
Table 13. Sample queries that use the BIOR.S.CONTAINS_LE and BIOR.S.CONTAINS_GE custom functions (continued)

<table>
<thead>
<tr>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT s.molweight FROM Swiss s WHERE BIOR.S.CONTAINS_GE(s.molweight, 100368) = 1 FETCH FIRST 3 ROWS ONLY;</td>
<td>MOLWEIGHT-------</td>
</tr>
<tr>
<td></td>
<td>100368</td>
</tr>
<tr>
<td></td>
<td>103625</td>
</tr>
<tr>
<td></td>
<td>132801</td>
</tr>
<tr>
<td>3 record(s) selected.</td>
<td></td>
</tr>
<tr>
<td>SELECT s.journalissue FROM Swiss s WHERE BIOR.S.CONTAINS_GE(s.journalissue, 172) = 1 FETCH FIRST 3 ROWS ONLY;</td>
<td>JOURNALISSUE----</td>
</tr>
<tr>
<td></td>
<td>172 21</td>
</tr>
<tr>
<td></td>
<td>242</td>
</tr>
<tr>
<td></td>
<td>196</td>
</tr>
<tr>
<td>3 record(s) selected.</td>
<td></td>
</tr>
</tbody>
</table>

Queries that use the BIOR.S.SEARCH_TERM custom function

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

Table 14. Sample queries that use the BIOR.S.SEARCH_TERM custom function

<table>
<thead>
<tr>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT s.publicationyear FROM Swiss s WHERE BIOR.S.SEARCH_TERM (s.id, '[PublicationYear EQ number:1997;]')=1 FETCH FIRST 10 ROWS ONLY;</td>
<td>PUBLICATIONYEAR</td>
</tr>
<tr>
<td></td>
<td>1997</td>
</tr>
<tr>
<td></td>
<td>1997 2000</td>
</tr>
<tr>
<td></td>
<td>1994 1997</td>
</tr>
<tr>
<td></td>
<td>1997 1998</td>
</tr>
<tr>
<td></td>
<td>1994 1995 1997</td>
</tr>
<tr>
<td></td>
<td>1997 1999</td>
</tr>
<tr>
<td></td>
<td>1997</td>
</tr>
<tr>
<td>10 record(s) selected.</td>
<td></td>
</tr>
<tr>
<td>SELECT s.molweight FROM Swiss s WHERE BIOR.S.SEARCH_TERM (s.id, '[MolWeight EQ number:100368;]') = 1 FETCH FIRST 10 ROWS ONLY;</td>
<td>MOLWEIGHT-------</td>
</tr>
<tr>
<td></td>
<td>100368</td>
</tr>
<tr>
<td></td>
<td>100368</td>
</tr>
<tr>
<td>2 record(s) selected.</td>
<td></td>
</tr>
</tbody>
</table>
Table 14. Sample queries that use the BIOR.S.SEARCH TERM custom function (continued)

<table>
<thead>
<tr>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
</table>
| SELECT s.molweight FROM Swiss s WHERE BIOR.S.SEARCH_TERM (s.id, '?[MolWeight GREATER number:100368;]') = 1 FETCH FIRST 10 ROWS ONLY; | MOLWEIGHT  
|                                                                      | 100368                  |
|                                                                      | 103625                  |
|                                                                      | 132801                  |
|                                                                      | 194328                  |
|                                                                      | 130277                  |
|                                                                      | 287022                  |
|                                                                      | 289130                  |
|                                                                      | 135502                  |
|                                                                      | 112715                  |
|                                                                      | 112599                  |
|                                                                      | 10 record(s) selected.  |

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

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

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

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

<table>
<thead>
<tr>
<th>ID</th>
<th>ID</th>
<th>PARENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>100K_RAT</td>
<td>100K_RAT.1</td>
<td>swissprot:100K_RAT</td>
</tr>
<tr>
<td>100K_RAT</td>
<td>100K_RAT.2</td>
<td>swissprot:100K_RAT</td>
</tr>
<tr>
<td>100K_RAT</td>
<td>100K_RAT.3</td>
<td>swissprot:100K_RAT</td>
</tr>
<tr>
<td>100K_RAT</td>
<td>100K_RAT.4</td>
<td>swissprot:100K_RAT</td>
</tr>
<tr>
<td>100K_RAT</td>
<td>100K_RAT.5</td>
<td>swissprot:100K_RAT</td>
</tr>
<tr>
<td>100K_RAT</td>
<td>100K_RAT.6</td>
<td>swissprot:100K_RAT</td>
</tr>
<tr>
<td>100K_RAT</td>
<td>100K_RAT.7</td>
<td>swissprot:100K_RAT</td>
</tr>
<tr>
<td>100K_RAT</td>
<td>100K_RAT.8</td>
<td>swissprot:100K_RAT</td>
</tr>
<tr>
<td>100K_RAT</td>
<td>100K_RAT.9</td>
<td>swissprot:100K_RAT</td>
</tr>
<tr>
<td>104K_THEPA</td>
<td>104K_THEPA.1</td>
<td>swissprot:104K_THEPA</td>
</tr>
</tbody>
</table>

10 record(s) selected.

Optimizing BioRS wrapper performance

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

Guidelines for optimizing BioRS wrapper performance

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

Minimize the amount of data that is transferred between search engines.

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

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

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

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

**Maintain BioRS statistical information in the federated environment.**

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

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

**BioRS statistical information**

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

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

For BioRS data sources, critical statistical information includes:

- The cardinality of a nickname. For BioRS data sources, nickname cardinality is equivalent to the number of entries in the corresponding BioRS databank.
- The cardinality of the column that corresponds to the BioRS _ID_ element. The cardinality of this column must match the cardinality of the nickname in which the column is referenced.
- The cardinality of all columns that the BioRS wrapper might need to use.
Tip: to optimize the performance of the BioRS wrapper, you should maintain current statistics about the BioRS data sources. If the statistical data or structural characteristics change for a remote object on which a nickname is defined, you must update the corresponding cardinality statistics in your federated system. The cardinality statistics are stored in the federated database system catalog in the SYSSTAT.TABLES view and in the SYSSTAT.COLUMNS view.

To maintain BioRS cardinality statistics in your federated system:
1. **Determine the cardinality statistics** for the nickname, if necessary.
2. **Update the cardinality statistics for the nickname** in the catalog views.
3. **Update the cardinality statistics for the columns** in the catalog views.

**Determining BioRS databank cardinality statistics**

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

**Procedure**

To determine cardinality statistics for a specific databank in BioRS:

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

**Updating BioRS nickname cardinality statistics**

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

**Before you begin**

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

**About this task**

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

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

**Procedure**

To update the BioRS nickname cardinality statistics:

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

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

For example, if the nickname is JONES.SWISS, you use the following UPDATE statement to update the statistics:
**Updating BioRS column cardinality statistics**

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

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

**Restrictions**

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

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

**About this task**

**Procedure**

To update BioRS column cardinality statistics:

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

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

**SYSSTAT.COLUMNS**

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

**COLCARD=(SELECT COUNT(DISTINCT column_name)**

The name of the column in the nickname that you are updating the statistics for.

**TABSCHEMA= nickname_schema**

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

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

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

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

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


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

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

**Updating BioRS _ID_ column cardinality**

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

**Before you begin**

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

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

**About this task**

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

**Procedure**
To update BioRS _ID_ column cardinality statistics:

Issue the UPDATE statement to modify the catalog view.

For example, :

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

**SYSSTAT.COLUMNS**

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

**SET COLCARD=cardinality_number**

The BioRS databank cardinality number that corresponds to the nickname of the column that you are updating the statistics for.

**TABSHEMA= nickname_schema**

The name of the schema for the nickname that you want to update.

**TABNAME=nickname_name**

The name of the nickname that you want to update.

**COLNAME=column_name**

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

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

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

**OPTION='ELEMENT_NAME'**

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

**SETTING='_ID_'**

Specifies that the column for the ELEMENT_NAME option is '_ID_'.
Chapter 3. Configuring access to BLAST data sources

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

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

BLAST wrapper

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

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

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

Table 15. BLAST search types supported by the BLAST wrapper

<table>
<thead>
<tr>
<th>BLAST search type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLASTn</td>
<td>A type of BLAST search in which a nucleotide sequence is compared with the contents of a nucleotide sequence database to find sequences with regions homologous to regions of the original sequence.</td>
</tr>
<tr>
<td>BLASTp</td>
<td>A type of BLAST search in which an amino acid sequence is compared with the contents of an amino acid sequence database to find sequences with regions homologous to regions of the original sequence.</td>
</tr>
<tr>
<td>BLASTx</td>
<td>A type of BLAST search in which a nucleotide sequence is compared with the contents of an amino acid sequence database to find sequences with regions homologous to regions of the original sequence. The query sequence is translated in all six reading frames, and each of the resulting sequences is used to search the sequence database.</td>
</tr>
<tr>
<td>tBLASTn</td>
<td>A type of BLAST search in which an amino acid sequence is compared with the contents of a nucleotide sequence database to find sequences with regions homologous to regions of the original sequence. The sequences in the sequence database are translated in all six reading frames, and the resulting sequences are searched for regions homologous to regions of the query sequence.</td>
</tr>
</tbody>
</table>
Table 15. **BLAST search types supported by the BLAST wrapper (continued)**

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

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

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

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

The results are returned to BLAST and then to the daemon. The daemon returns the retrieved data to the BLAST wrapper. The wrapper transforms the data into a relational table format, and returns this table to you or your application. The returned data contains two parts:
- A series of standard, fixed columns familiar to BLAST users, and
- User-configured definition line information.

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

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

![Figure 4. Sample FASTA file, nucleo1](image)

The standard formatdb application from NCBI transforms the FASTA file to a BLAST-able data set. The data is then ready to be queried.
The following query is transformed by the BLAST wrapper and runs against the BLAST-able data set.

```sql
SELECT Unique_ID, Experiment_Number, Organism_Number, HSP_Info, Score
FROM nucleo
WHERE BlastSeq = 'ACATTCTTATAGATTTGCTACTCCAGGATAGATCATCTCTGGTGCTACCACTAGTGGGAGGCTGAGGATGAGAGATTTGGACCAATACAGAAACTCACCTACAGGAGTGAGCGGGGTGGTAGAAGGAGCCCGAGATATGAAAACGTGTTTCACAGCACTACGAGGATGAGCCCGAGATATGAAA';
```

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

<table>
<thead>
<tr>
<th>Unique ID</th>
<th>Experiment number</th>
<th>Organism number</th>
<th>HSP_INFO</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMON5744</td>
<td>4986</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMON5426</td>
<td>8924</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMON5498</td>
<td>6512</td>
<td>13</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

### Adding BLAST data sources to a federated server

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

**Before you begin**
- Federation must be installed on a server that will act as the federated server.
- A database must exist on the federated server.

**About this task**

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

**Procedure**
To add BLAST data sources to a federated server:

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

### Verifying the versions of the BLAST server files

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

**Before you begin**

Verify that you have:

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

**Procedure**

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

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

### Configuring the BLAST daemon

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

**Before you begin**

The BLAST daemon must have:

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

**Restrictions**

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

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

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

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

**Procedure**

To configure the BLAST daemon:

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>On federated servers that run UNIX</td>
<td>The name of the BLAST daemon executable file is:</td>
</tr>
<tr>
<td></td>
<td>• libdb2osse.a on AIX operating systems</td>
</tr>
<tr>
<td></td>
<td>• libdb2osse.so on Linux or Solaris operating systems</td>
</tr>
<tr>
<td></td>
<td>This file is installed in the $DB2PATH/bin directory.</td>
</tr>
<tr>
<td>On federated servers that run Windows</td>
<td>The name of the OSS library file is db2osse.dll. This file is installed in the %DB2PATH%\bin directory.</td>
</tr>
</tbody>
</table>
3. Ensure that the BLAST daemon configuration file is on the correct server.

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

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

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

4. Edit the daemon configuration file.
   a. Optional: You can rename the configuration file.
   b. Ensure that the first line in the configuration file is an equal sign. If the equal sign is missing, the daemon will not start. An error message will indicate that the DAEMON_PORT was not specified.
   c. Ensure that the last line in the configuration file ends with a new line. The sample configuration file that is provided with the federated server ends with a new line. If the last line does not end with a new line, you will receive an error message when you attempt to run your first BLAST query using the data source that is listed on the last line.
   d. Specify the following options in the configuration file.
      For options that require paths, you can specify relative paths. Relative paths are relative to the directory from which the daemon process was started.

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

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

   **DAEMON_LOGFILE_DIR**
   This is the directory in which the daemon creates its log file. This file contains useful status and error information generated by the BLAST daemon.
Q_SEQ_DIR_PATH
This is the directory in which a temporary query sequence data file is created by the daemon. This temporary file is cleaned up once the BLAST job completes.

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

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

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

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

data_source_name=data_source_path

On federated servers that run UNIX
For example, to specify the GenBank BLAST-able data source you would add the following line to the daemon configuration file:
genbank=/dsk/1/nucl_data/genbank

On federated servers that run Windows
For example, to specify the GenBank BLAST-able data source you would add the following line to the daemon configuration file:
c:\vnr_data\genbank_nonest1.fasta

The path indicated in a database specification entry must contain the three index files.
• For nucleotide data sources, the index files have these extensions:
  – .nhr
  – .nin
  – .nsq
• For amino acid data sources, the index files have these extensions:
  – .phr
  – .pin
  – .psq
The database specification entry must indicate the name of the file that contains the original FASTA-formatted data. The three index files must have the same root name as the file containing the original FASTA-formatted data.

**BLAST daemon configuration file - Examples**

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

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

**BLAST daemon configuration file (UNIX)**

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

```bash
= DAEMON_PORT=4007
MAX_PENDING_REQUESTS=10
DAEMON_LOGFILE_DIR=/home/blastuser/logs
Q_SEQ_DIR_PATH=/tmp
BLAST_OUT_DIR_PATH=/tmp
BLASTALL_PATH=/home/blastuser/blast/blastall
SSL_ENABLED=Yes
SSL_KEYSTORE_FILE=/home/artk/mycerts.kdb
SSL_KEYSTORE_PASSWORD=arglebargle
SSL_VERIFY_CLIENT_CERTIFICATE=Yes
SSL_SERVER_CERTIFICATE_LABEL=barney
genbank=/dsk/1/nucl_data/genbank
swissprot=/dsk/1/prot_data/swissprot
```

**BLAST daemon configuration file (Windows)**

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

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

**Starting the BLAST daemon**

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

**Before you begin**

Before you start the BLAST daemon:
- You must configure the BLAST daemon
- You must have write access to all of the paths that are listed under the following entries in the configuration file
About this task

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

Procedure

To start the BLAST daemon:
1. Open the directory where the daemon executable files are located.
2. Issue the db2blast_daemon command to run the executable files.

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

**db2blast_daemon command - options and examples**

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

**Options - db2blast_daemon command**

The options for the db2blast_daemon command are:

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

- **-a action**
  Performs the specified activity. Valid actions are `status`, `install`, `start`, `stop`, and `remove`.
  You can specify this option only on Windows servers.

- **-c config_file**
  Instructs the daemon to use the specified configuration file. If you do not specify the configuration file, the daemon searches for the `BLAST_DAEMON.config` file in the directory where the daemon executable files are installed.
  You can specify this option on UNIX and Windows servers. On Windows servers, you can specify this option only with the `install` or `start` actions.

- **-d debug_level**
  Sets the daemon debug level to the specified value. The valid values are 1, 2, or 3. You can use this option with the `install` and `start` actions.
  You can specify this option on UNIX and Windows servers.

- **-u user_id**
  Sets the daemon to run under the specified user ID. You can use this option with the `install` action.
  You can specify this option only on Windows servers.

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

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

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

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

**Examples - db2blast_daemon command**

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

**Start the daemon**

To start the daemon, issue the following command:

```
db2blast_daemon -a start
```

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

**Daemon configuration file**

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

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

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

**Debugging**

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

```
db2blast_daemon -a install -d 2
 db2blast_daemon -a start
```

**Status of the daemon (Windows)**

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

```
db2blast_daemon -a status
```

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

**Stop the daemon**

To stop the daemon, issue the following command:

```
db2blast_daemon -a stop
```
This command assumes that the daemon configuration file is in the same directory as the executable file.

**Remove the daemon**

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

```
db2blast_daemon -a remove
```

**Registering the BLAST wrapper**

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

**Procedure**

To register the BLAST wrapper:

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

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

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

**BLAST wrapper library files**

The BLAST wrapper library files are added to the federated server when you install federation.

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

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

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

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Wrapper library file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/&lt;install_path&gt;/lib32/</td>
<td>libdb2lsblast.a</td>
</tr>
<tr>
<td></td>
<td>/usr/opt/&lt;install_path&gt;/lib64/</td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib32</td>
<td>libdb2lsblast.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib64</td>
<td></td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib32</td>
<td>libdb2lsblast.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib64</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib32</td>
<td>db2lsblast.dll</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib64</td>
<td></td>
</tr>
</tbody>
</table>
<install_path> is the directory path where the federated server is installed on UNIX or Linux.

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

**CREATE WRAPPER statement - Examples for the BLAST wrapper**

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

**Example of registering a wrapper**

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

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

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

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

**Example of registering a wrapper for a SOCKS proxy server**

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

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

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

- **PROXY_TYPE 'SOCKS'**
  - Specifies the proxy type that is used to access the Internet when the federated server is behind a firewall.

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

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

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

When you attempt to access a BLAST server, the federated server establishes a connection to the BLAST server. If your federated system uses a proxy server to access BLAST data sources, you must create user mappings.
A user mapping is an association between each federated server user ID and password, and the corresponding data source user ID and password.

About this task

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

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

Procedure

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

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Using the command line</td>
<td>Issue the CREATE USER MAPPING statement. For example:</td>
</tr>
<tr>
<td></td>
<td>CREATE USER MAPPING FOR local_userID</td>
</tr>
<tr>
<td></td>
<td>SERVER server_definition_name</td>
</tr>
<tr>
<td></td>
<td>OPTIONS (REMOTE_AUTHID 'remote_userID',</td>
</tr>
<tr>
<td></td>
<td>REMOTE_PASSWORD 'remote_password')</td>
</tr>
<tr>
<td></td>
<td>PROXY AUTHID 'proxy_server_userID',</td>
</tr>
<tr>
<td></td>
<td>PROXY PASSWORD 'proxy_server_password';</td>
</tr>
</tbody>
</table>

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

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

**CREATE USER MAPPING statement - examples for the BLAST wrapper**

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

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

**Proxy server example**

The following example shows how to map a federated server user ID to a BLAST proxy server user ID and password:
To register a server definition and specify a SOCKS proxy server with authentication information, use the following statement:

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

- **groucho** Specifies the local authorization ID that you are mapping to a remote user ID and password, which are defined at the BLAST proxy server.
- **SERVER blast_proxy** Specifies the server definition name that you registered in the CREATE SERVER statement for the BLAST server.
- **REMOTE_AUTHID 'grouchom'** Specifies the user ID at the BioRS server to which you are mapping groucho. This remote ID must be in a format that is expected by the BLAST server.
- **REMOTE_PASSWORD 'all4one'** Specifies the password that is associated with 'grouchom'.
- **PROXY_AUTHID 'marxg'** Specifies the user ID on the proxy server. This user mapping option is required when the proxy server requires authentication.
- **PROXY_PASSWORD 'them2us'** Specifies the password on the proxy server that is associated with the user name 'marxg'. This user mapping option is required when the proxy server requires authentication.

### Registering the server definition for a BLAST data source

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

**About this task**

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

**Procedure**

To register a server definition for a BLAST data source:

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Use the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects. Follow the steps in the wizard.</td>
</tr>
</tbody>
</table>
From the command line

Issue the CREATE SERVER statement. For example:

```sql
CREATE SERVER server_definition_name
  TYPE blast_search_type
  VERSION version_number
  WRAPPER wrapper_name
  OPTIONS (NODE 'node_name');
```

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

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

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

**CREATE SERVER statement - Examples for the BLAST wrapper**

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

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

**Required parameters and options**

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

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

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

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

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

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

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

  Although the name of the node is specified as an option in the CREATE SERVER statement, it is required for BLAST data sources.
Registering server definitions for multiple search types

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

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

Optional server options

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

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

**DAEMON_PORT '3232'**

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

**USE_CLOB_SEQUENCE 'N'**

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

Server definitions when a proxy server is used

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

- You want to retrieve data by using a URI.
- The URI retrieves data from behind a firewall, through a SOCKS proxy.

Registering a server definition for a SOCKS proxy server

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

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

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

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

PROXY_TYPE 'SOCKS'
    Specifies the proxy type that is used to access the Internet when behind a firewall.
PROXY_SERVER_NAME 'proxy_socks'
   Specifies the proxy server name or IP address. This option is required if the value for the PROXY_TYPE server option is 'SOCKS'.

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

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

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

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

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

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

Registering nicknames for BLAST data sources

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

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

About this task

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

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

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

Procedure

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects. Follow the steps in the wizard.</td>
</tr>
<tr>
<td>From command line</td>
<td>Issue the CREATE NICKNAME statement. For example:</td>
</tr>
<tr>
<td></td>
<td>CREATE NICKNAME nickname</td>
</tr>
<tr>
<td></td>
<td>(column_name data_type</td>
</tr>
<tr>
<td></td>
<td>OPTIONS (nickname_column_options), column_name data_type</td>
</tr>
<tr>
<td></td>
<td>OPTIONS (nickname_column_options), column_name data_type</td>
</tr>
<tr>
<td></td>
<td>OPTIONS (nickname_column_options)</td>
</tr>
<tr>
<td></td>
<td>FOR SERVER server_definition_name OPTIONS (nickname_options);</td>
</tr>
</tbody>
</table>

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

**Definition line parsing**

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

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

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

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

Valid data types for the definition line columns are CLOB, DOUBLE, FLOAT, INTEGER, and VARCHAR.
Fixed input columns for BLAST nicknames
A set of fixed input columns are automatically created when you register a nickname for a BLAST data source.

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

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

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

*Table 18. Fixed input columns for BLAST nicknames*

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlastSeq</td>
<td>VARCHAR (32000) or CLOB</td>
<td>=</td>
<td>Passes the query sequence to the BLAST wrapper.</td>
</tr>
<tr>
<td>DB_Genetic_Code</td>
<td>INTEGER</td>
<td>=</td>
<td>DB genetic code for tblastn and tblastx queries uses default = 1.</td>
</tr>
<tr>
<td>E_Value</td>
<td>DOUBLE</td>
<td>&lt;</td>
<td>Both an input and an output parameter. As an input parameter, this column indicates to the BLAST wrapper the upper limit of the expected values that should be returned from blastall.</td>
</tr>
<tr>
<td>ExtendedGapCost</td>
<td>INTEGER</td>
<td>=</td>
<td>Specifies the value that blastall deducts from the score of an alignment if a gap that was already introduced in either the query sequence or the hit sequence must be extended by one nucleotide or amino acid to allow the length of the alignment to grow.</td>
</tr>
<tr>
<td>FilterSequence</td>
<td>CHAR(1)</td>
<td>=</td>
<td>Indicates to blastall whether to perform filtering to remove biologically uninteresting segments from the query sequence. If the search type is BLASTn, the filter used is DUST. Otherwise, filtering is performed by SEG.</td>
</tr>
<tr>
<td>Final_XDropoff</td>
<td>INTEGER</td>
<td>=</td>
<td>The X dropoff value for the final gapped alignment, measured in bits. The value 0.0 invokes the default behavior. For blastn and megablast queries the default is 50 bits. For tblastx queries the default is 0 bits. For all other query types the default is 25 bits [INTEGER data types].</td>
</tr>
<tr>
<td>Name</td>
<td>Data type</td>
<td>Operators</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------</td>
<td>-----------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GapCost</td>
<td>INTEGER</td>
<td>=</td>
<td>Specifies the value that blastall deducts from the score of an alignment if a gap must be introduced in either the query sequence or the hit sequence to allow the length of the alignment to grow.</td>
</tr>
<tr>
<td>GapAlign</td>
<td>CHAR(1)</td>
<td>=</td>
<td>Indicates to the wrapper whether gapped alignments are permitted in the BLAST output.</td>
</tr>
<tr>
<td>Mask_Lower_Case</td>
<td>CHAR(1)</td>
<td>=</td>
<td>Use lowercase filtering with a FASTA sequence.</td>
</tr>
<tr>
<td>Matrix</td>
<td>VARCHAR(50)</td>
<td>=</td>
<td>Determines which substitution matrix is used by blastall to determine the degree of similarity between the pairings of amino acids. Only those BLAST search types that compare amino acids to amino acids use this predicate.</td>
</tr>
<tr>
<td>NMatchReward</td>
<td>INTEGER</td>
<td>=</td>
<td>Specifies the value that blastall adds to the score of an alignment for each of the pairs of nucleotides in the homologous region that do match. Only those BLAST search types that compare nucleotides to nucleotides use this predicate.</td>
</tr>
<tr>
<td>NMisMatchPenalty</td>
<td>INTEGER</td>
<td>=</td>
<td>Specifies the value that blastall deducts from the score of an alignment if one of the pairs of nucleotides in the homologous region does not match. Only those BLAST search types that compare nucleotides to nucleotides use this predicate.</td>
</tr>
<tr>
<td>NumberOfAlignments</td>
<td>INTEGER</td>
<td>=</td>
<td>Specifies how many HSP alignments to include in the BLAST output.</td>
</tr>
<tr>
<td>Query_Genetic_Code</td>
<td>INTEGER</td>
<td>=</td>
<td>Query genetic code uses default = 1.</td>
</tr>
<tr>
<td>QueryStrands</td>
<td>INTEGER</td>
<td>=</td>
<td>Specifies which strands should be compared when performing a BLASTn search. A value of 1 indicates that the top strand should be used, 2 indicates the bottom strand, and 3 indicates that both strands should be compared.</td>
</tr>
<tr>
<td>ThresholdEx</td>
<td>INTEGER</td>
<td>=</td>
<td>Indicates the score threshold below which BLAST does not attempt to extend a hit any further.</td>
</tr>
<tr>
<td>WordSize</td>
<td>INTEGER</td>
<td>=</td>
<td>Indicates to blastall the length of the initial hits that blastall initially searches in the database.</td>
</tr>
</tbody>
</table>
**Table 18. Fixed input columns for BLAST nicknames (continued)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDropoff_Gapped</td>
<td>INTEGER</td>
<td>=</td>
<td>The X dropoff value for gapped alignment, measured in bits. The value 0.0 invokes the default behavior. For blastn queries the default is 30 bits. For megablast queries the default is 20 bits. For all other query types the default is 15 bits [INTEGER data types].</td>
</tr>
<tr>
<td>XDropoff_Ungapped</td>
<td>DOUBLE</td>
<td>=</td>
<td>The X dropoff value for ungapped extension, measured in bits. The value 0.0 invokes the default behavior. For blastn queries the default is 20 bits. For megablast queries the default is 10 bits. For all other query types the default is 7 bits [REAL data types].</td>
</tr>
</tbody>
</table>

**BLAST search types and switches for fixed input columns**

You need to know the BLAST search types and switches that each of the BLAST fixed input columns can support.

The supported BLAST search types and switches for each fixed input column are listed in the following table.

**Table 19. BLAST search types and switches supported by the input fixed columns**

<table>
<thead>
<tr>
<th>Fixed input column</th>
<th>BLAST search types</th>
<th>BLAST switch</th>
<th>Required</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlastSeq</td>
<td>n, p, x, tn, tx</td>
<td>-i</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>DB_Genetic_Code</td>
<td>tn, tx</td>
<td>-D</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>E_Value</td>
<td>n, p, x, tn, tx</td>
<td>-e</td>
<td>No</td>
<td>10</td>
</tr>
<tr>
<td>ExtendedGapCost</td>
<td>n, p, x, tn, tx</td>
<td>-E</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>FilterSequence</td>
<td>n, p, x, tn, tx</td>
<td>-F</td>
<td>No</td>
<td>T</td>
</tr>
<tr>
<td>Final_XDropoff</td>
<td>n, p, x, tn, tx</td>
<td>-Z</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>GapAlign</td>
<td>n, p, x, tn, tx</td>
<td>-g</td>
<td>No</td>
<td>T</td>
</tr>
<tr>
<td>GapCost</td>
<td>n, p, x, tn, tx</td>
<td>-G</td>
<td>No</td>
<td>11</td>
</tr>
<tr>
<td>Mask_Lower_Case</td>
<td>n, p, x, tn, tx</td>
<td>-U</td>
<td>No</td>
<td>F</td>
</tr>
<tr>
<td>Matrix</td>
<td>p, x, tn, tx</td>
<td>-n</td>
<td>No</td>
<td>BLOSUM62</td>
</tr>
<tr>
<td>NMisMatchPenalty</td>
<td>n</td>
<td>-q</td>
<td>No</td>
<td>-3</td>
</tr>
<tr>
<td>NMatchReward</td>
<td>n</td>
<td>-r</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>NumberOfAlignments</td>
<td>n, p, x, tn, tx</td>
<td>-b</td>
<td>No</td>
<td>250</td>
</tr>
<tr>
<td>Query_Genetic_Code</td>
<td>n, p, x, tn, tx</td>
<td>-Q</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>QueryStrands</td>
<td>n</td>
<td>-S</td>
<td>No</td>
<td>3</td>
</tr>
<tr>
<td>ThresholdEx</td>
<td>n, p, x, tn, tx</td>
<td>-f</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>WordSize (for Blastn, a value less than 7 is invalid)</td>
<td>n, p, x, tn, tx</td>
<td>-W</td>
<td>No</td>
<td>11 -BLASTn 3 -BLASTp</td>
</tr>
<tr>
<td>XDropoff_Ungapped</td>
<td>n, p, x, tn, tx</td>
<td>-y</td>
<td>No</td>
<td>0.0</td>
</tr>
</tbody>
</table>
### Fixed input column

<table>
<thead>
<tr>
<th>Fixed input column</th>
<th>BLAST search types</th>
<th>BLAST switch</th>
<th>Required</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDropoff_Gapped</td>
<td>n, p, x, tn, tx</td>
<td>-X</td>
<td>No</td>
<td>0</td>
</tr>
</tbody>
</table>

### Fixed output columns for BLAST nicknames

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

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

#### Table 20. Fixed output columns for BLAST nicknames

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_value</td>
<td>DOUBLE</td>
<td>Both an input and an output parameter. As an output parameter, this column provides the computed score for an HSP as reported in the BLAST results.</td>
</tr>
<tr>
<td>HIT_NUM</td>
<td>INTEGER</td>
<td>The hit number as reported in the BLAST results, starting with 1.</td>
</tr>
<tr>
<td>HSP_ALIGNMENT_LENGTH</td>
<td>INTEGER</td>
<td>The length of the HSP alignment.</td>
</tr>
<tr>
<td>HSP_IDENTITY</td>
<td>INTEGER</td>
<td>The percent identity of the alignment defined as the number of identities divided by the alignment length.</td>
</tr>
<tr>
<td>HSP_Info</td>
<td>VARCHAR(100)</td>
<td>The information string for the given HSP, as reported by BLAST. This string contains information about the number of nucleotides or amino acids that matched between the query sequence and the hit sequence.</td>
</tr>
<tr>
<td>HSP_GAPS</td>
<td>INTEGER</td>
<td>The percent gaps in the alignment defined as the number of gaps divided by the alignment length.</td>
</tr>
<tr>
<td>HSP_H_Start</td>
<td>INTEGER</td>
<td>The numeric position of the first homologous nucleotide or amino acid on the hit sequence.</td>
</tr>
<tr>
<td>HSP_H_End</td>
<td>INTEGER</td>
<td>The numeric position of the last homologous nucleotide or amino acid on the hit sequence.</td>
</tr>
<tr>
<td>HSP_H_Seq</td>
<td>VARCHAR(32000)</td>
<td>The segment of the hit sequence beginning at HSP_H_Start and ending at HSP_H_End. You can override the default data type for this column and specify CLOB, with a maximum length of 5 megabytes.</td>
</tr>
<tr>
<td>HSP_HIT_FRAME</td>
<td>INTEGER</td>
<td>The reading frame of the alignment in the hit sequence. Only available for blastx, tblastn, and tblastx type servers.</td>
</tr>
</tbody>
</table>
Table 20. Fixed output columns for BLAST nicknames (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSP_Midline</td>
<td>VARCHAR(32000)</td>
<td>The string output by BLAST that indicates the degree of homology between the amino acids or nucleotides at each position in the homologous regions of the query and hit sequences. You can override the default data type for this column and specify CLOB, with a maximum length of 5 megabytes.</td>
</tr>
<tr>
<td>HSP_NUM</td>
<td>INTEGER</td>
<td>The HSP number as reported in the BLAST results, starting with 1.</td>
</tr>
<tr>
<td>HSP_Q_Start</td>
<td>INTEGER</td>
<td>The numeric position of the first homologous nucleotide or amino acid on the query sequence.</td>
</tr>
<tr>
<td>HSP_Q_End</td>
<td>INTEGER</td>
<td>The numeric position of the last homologous nucleotide or amino acid on the query sequence.</td>
</tr>
<tr>
<td>HSP_Q_Seq</td>
<td>VARCHAR(32000)</td>
<td>The segment of the query sequence beginning at HSP_Q_Start and ending at HSP_Q_End. You can override the default data type for this column and specify CLOB, with a maximum length of 5 megabytes.</td>
</tr>
<tr>
<td>HSP_POSITIVE</td>
<td>INTEGER</td>
<td>The percent positives of the alignment defined as the number of positives divided by the alignment length.</td>
</tr>
<tr>
<td>HSP_QUERY_FRAME</td>
<td>INTEGER</td>
<td>The reading frame of the alignment in the query sequence. Only available for blastx, tblastn, and tblastx type servers.</td>
</tr>
<tr>
<td>HSP_Score</td>
<td>DOUBLE</td>
<td>Use lowercase filtering with a FASTA sequence.</td>
</tr>
<tr>
<td>Length</td>
<td>INTEGER</td>
<td>The length of the hit sequence as reported in the BLAST results.</td>
</tr>
<tr>
<td>Score</td>
<td>DOUBLE</td>
<td>The computed score for an HSP as reported in the BLAST results.</td>
</tr>
</tbody>
</table>

CREATE NICKNAME statement - example for the BLAST wrapper

Use the CREATE NICKNAME statement to register a nickname for a BLAST data source that you want to access.

The CREATE NICKNAME statement for a genbank nickname

The following example defines the nickname genbank. This example assumes that the definition field in a BLAST result set contains the following information:

>276342  15:8924  PMON5426

276342 The accession field of the BLAST result.
The definition field in a BLAST result set that contains an organism number (15) followed by an experiment number (8924) and then a unique identifier (PMON5426).

With this information, you can create the following nickname:

```sql
CREATE NICKNAME genbank
    (
        accession_num INTEGER OPTIONS(INDEX '1', DELIMITER ' '),
        organism_num INTEGER OPTIONS(INDEX '2', DELIMITER ':'),
        experiment_num INTEGER OPTIONS(INDEX '3', DELIMITER ': '),
        unique_id VARCHAR(10) OPTIONS(INDEX '4')
    )
FOR SERVER server_blastn
    OPTIONS(DATASOURCE 'genbank', TIMEOUT '300');
```

In this example, the columns use the INDEX options to refer to portions of the definition field:

- The INDEX 1 refers to the accession field 276342
- The INDEX 2 refers to the organism number 15
- The INDEX 3 refers to the experiment number 8924
- The INDEX 4 refers to the unique identifier PMON5426

After you submit the CREATE NICKNAME statement, you can use the `genbank` nickname to query your federated system. You can also join the `genbank` nickname with other nicknames and tables in your federated system.

---

**Querying BLAST data sources**

After you create the BLAST nicknames, you can query BLAST data sources. The choices that you make when you create the queries impact the performance of the BLAST wrapper when it accesses those data sources.

**Setting up TurboBlast to work with the BLAST wrapper**

TurboBlast is a proprietary version of the BLAST program. To use TurboBlast with the BLAST wrapper, you must complete several additional configuration steps.

**Restrictions**

TurboBlast does not support certain blastall command options. For example, the gapped alignment option `-g F` is not supported. If you specify the `F` option for GapAlign column in your BLAST nickname, TurboBlast generates an error. For a complete list of unsupported options, refer to the *TurboBlast 2.0 User Guide*.

**About this task**

The BLAST wrapper and TurboBlast support the AIX, Linux, Solaris and Windows operating systems. However, the BLAST daemon is not available on Windows operating systems. The daemon will work with TurboBlast on Windows operating systems when the BLAST daemon is available on those operating systems.

**Procedure**

To set up TurboBlast to work with the BLAST wrapper:
1. Install and configure the BLAST wrapper.
2. Run a query on a BLAST-able database to test your setup.
3. Install and configure TurboBlast according to the TurboBlast 2.0 Installation and Reference Guide. You can install and set up the TurboBlast system in various ways. To allow the BLAST wrapper to work with TurboBlast, you need to install and set up the TurboBlast Client on the computer on which you have your BLAST daemon. The BLAST daemon can invoke the tblastall program. Be sure to test the TurboBlast system after you have installed and configured TurboBlast. Follow the instructions in the TurboBlast 2.0 Installation and Reference Guide.

4. Edit the BLAST_DAEMON.config file:
   a. Add the BLASTALL_PATH parameter and specify the complete path of the executable file for the tblastall program.
      For example:
      
      ```
      BLASTALL_PATH=/home/blasttst/turboblast/TBlast-2.1/tblastall
      ```
   b. Add the BLAST-able database specification entry and specify the BLAST-able database name that you used to upload your BLAST-able database to TurboBlast. The database names display when you enter the listdatabase -l command under TurboBlast. This TurboBlast database name should be used instead of the path to the BLAST-able data source. For example:
      
      ```
      genbank=genbank_database_name_in_TurboBlast
      ```

5. Restart the BLAST daemon.
   The BLAST daemon invokes the tblastall program instead of the blastall program to search the BLAST-able databases.
   The log files that are related to the tblastall program are written to the directory that is specified in the DAEMON_LOGFILE_DIR parameter in the BLAST_DAEMON.config file. Also check the STDERR.log and STDOUT.log that are produced by the BLAST daemon in the same directory.

### Construct BLAST SQL queries

How you construct your BLAST queries impacts the quality of the results that are returned. Learn how to use predicates and custom functions in your BLAST queries.

#### Predicates on input and output columns

Predicates on input columns are used to pass standard BLAST switches to the blastall executable program.

Predicates on the output columns are processed by the federated server.

#### The BlastSeq input column predicate

To be valid, every query passed to the BLAST wrapper must contain at least a predicate with the BlastSeq input column. All other predicates are optional.

Wrappers that require predicates, such as the BlastSeq fixed input column, cannot process queries that result in a left outer join on the required predicate. For example, the following query returns an SQL0901N error:

```sql
SELECT n1.Score FROM blastNN1 n1
   LEFT OUTER JOIN myseqs n2 ON N1.BlastSeq = n2.seq
```
Function templates for CLOB columns

If the BlastSeq column for a BLAST nickname is defined as a CLOB data type, you must create sequence match function templates before you can query BLAST data sources. The SQL statements that you issue on the federated database to create the function templates are:

CREATE FUNCTION lblast.sequence_match (CLOB, CLOB) 
  RETURNS INTEGER AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION;

CREATE FUNCTION lblast.sequence_match (CLOB, VARCHAR(1)) 
  RETURNS INTEGER AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION;

CREATE FUNCTION lblast.sequence_match (VARCHAR(1), CLOB) 
  RETURNS INTEGER AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION;

CREATE FUNCTION lblast.sequence_match (VARCHAR(1), VARCHAR(1)) 
  RETURNS INTEGER AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION;

Using the SUBSTR scalar function in queries

You can use the SUBSTR scalar function in the query predicate. The SUBSTR function returns part of a string.

For example:

SELECT BlastSeq FROM t1, t2 
  WHERE BlastSeq = SUBSTR (t2.sequence, 15, 300);

You can use the SUBSTR function in a SEQUENCE_MATCH function for CLOB data types.

For example:

SELECT BlastSeq FROM t1, t2 
  WHERE LsBlast.SEQUENCE_MATCH (BlastSeq, SUBSTR(t2.sequence, 15, 300))=1;

Important: Using the SUBSTR() function does not produce the same results as the -L option with the blastall program on the command line. Use the SUBSEQUENCE custom function instead of the SUBSTR() function when you want to produce the same results as -L option.

Function templates for the SUBSEQUENCE custom function

To produce the same results as the -L option on the command line with the blastall program, use the SUBSEQUENCE custom function. You must create subsequence function templates before you can query BLAST data sources. The SQL statements that you issue on the federated database to create the function templates are:

CREATE FUNCTION lsblast.subsequence (CLOB, INTEGER, INTEGER) 
  RETURNS VARCHAR(1) AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION;

CREATE FUNCTION lsblast.subsequence (VARCHAR(1), INTEGER, INTEGER) 
  RETURNS VARCHAR(1) AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION;

For more details about the -L option, refer to the blastall documentation.

BLAST data source – example queries

These examples show you how to structure your queries to optimize system performance. Learn how to use the custom functions and wildcards in your queries, access specific BioRS data type columns, and to form an equijoin between parent and child nicknames.
Use the input predicates in the WHERE clause of your BLAST SQL queries.

In these queries, the name that is used for each nickname indicates the type of BLAST search and the data source. Some of the queries make use of other hypothetical data sources so that the examples show the behavior of the BLAST wrapper when the BLAST data is joined with data from other sources.

**Input predicates example**

This query searches the GenBank database for homologues, using a partial sequence and shows three input predicates: BlastSeq, GapCost, and NMisMatchPenalty:

```
SELECT * FROM genbank b
WHERE
BlastSeq = 'GTCCAGCC...' AND
GapCost = -10 AND
NMisMatchPenalty = -4;
```

**BLASTn search that returns all available columns**

When the following SELECT statement is run, the wrapper performs a BLASTn search of GenBank using the indicated sequence. The wrapper returns all of the available columns, including both the input parameter columns and the BLAST result columns.

```
SELECT * FROM blastn_genbank
WHERE
BlastSeq = 'caacccctccagccgagttgtcaatggcgaggaagctgttccccac';
```

**BLASTn search of GenBank that passes parameters to the daemon**

When the following SELECT statement is run, the wrapper performs a BLASTn search of GenBank using the indicated sequence. In addition, the wrapper passes the GapCost and the NMisMatchPenalty parameters to the daemon. These parameters are then passed to the blastall command line. The wrapper returns all of the available columns, including both the input parameter columns and the BLAST result columns.

```
SELECT * FROM blastn_genbank
WHERE
BlastSeq = 'caacccctccagccgagttgtcaatggcgaggaagctgttccccac'
AND GapCost = 8
AND NMisMatchPenalty = -4;
```

**BLASTp search of SWISS-PROT that is based on the number of sequences returned**

When the following SELECT statement is run, the wrapper performs zero or more BLASTp searches of SWISS-PROT, depending on the number of sequences that are returned from a hypothetical protein sequence database. This statement will be separated into two queries by the federated database. One query will perform a BLASTp search that is run for each row that is returned from the hypothetical protein database. The wrapper returns all of the available columns, including both the input parameter columns and the BLAST result columns.

```
SELECT blp.* FROM blastp_swissprot blp, protein_db prdb
WHERE prdb.keyword = 'malic enzyme'
AND blp.BlastSeq = prdb.sequence;
```
**BLASTx search of SWISS-PROT that is based on an input sequence**

When the following SELECT statement is run, the wrapper performs a BLASTx search of SWISS-PROT using the indicated sequence. In this example, blastall translates the input sequence in all six reading frames and perform the homology search using each of the six newly created protein sequences. The HSPs in the results will contain amino acid-amino acid alignments, rather than nucleotide-nucleotide alignments. The supplied parameter is passed to the daemon and then to blastall command line. The wrapper returns only those columns that are specifically requested in the query.

```sql
SELECT Score, E_Value, HSP_Info, HSP_Q_Seq, HSP_H_Seq, HSP_Midline
FROM blastx_swissprot
WHERE BlastSeq = 'gagttgtcaatggcgagg'
AND GapCost = 8;
```

**tBLASTx search of GenBank that is based on the number of sequences returned**

When the following SELECT statement is run, the wrapper performs zero or more tBLASTx searches of GenBank, depending on the number of sequences returned from a hypothetical gene expression database. The statement is broken into two separate queries by the federated database. One query will perform a tBLASTx search and is run for each row that is returned from the hypothetical gene expression database. In this case, blastall will translate the input sequence and all of the sequences in GenBank in all six reading frames. Additionally, blastall will perform the homology search by using each of the six newly created query protein sequences and all of the newly created database protein sequences. The HSPs in the results will contain amino acid-amino acid alignments, rather than nucleotide-nucleotide alignments. The supplied parameters are passed to the daemon and then to blastall command line. The wrapper returns only those columns that are specifically requested in the query.

```sql
SELECT tblx.Score, tblx.E_Value, tblx.HSP_Info, tblx.HSP_Q_Seq, tblx.HSP_H_Seq, tblx.HSP_Midline
FROM tblastx_genbank tblx, gen_exp_database gdb
WHERE tblx.BlastSeq = gdb.sequence
AND gdb.organism = 'interesting organism'
AND GapCost = 8
AND FilterSequence = 'F';
```

**Guidelines for optimizing BLAST query performance**

The location of the BLAST daemon can impact query performance.

To improve query performance, you should use a separate BLAST server for the BLAST daemon. The federated server and the BLAST server should be on separate hardware. The BLAST daemon should reside on the BLAST server.
Chapter 4. Configuring access to DB2 data sources

To configure a federated server to access DB2 family data sources, you must provide the federated server with information about the data sources and objects that you want to access.

Before you begin
- Verify the set up of the federated server.

Procedure

To configure the federated server to access DB2 data sources, complete these tasks:

Cataloging a DB2 node entry

You must catalog a node entry that specifies the protocol that the federated server uses to connect to the DB2 data source.

Procedure

To catalog a node entry:

Issue the CATALOG TCPIP NODE command.

For example:
CATALOG TCPIP NODE db2_node REMOTE system42 SERVER db2tcp42

where:
- db2_node is the name that you assign to the node
- system42 is the host name of the system where the data source resides
- db2tcp42 is the service name or primary port number of the server database manager instance

Cataloging the remote DB2 database

To identify which database the federated server connects to, you must catalog the remote DB2 database in the federated server system database directory.

Procedure

To catalog the remote database:

1. Use the Configuration Assistant to catalog the database, or issue the CATALOG DATABASE command.

For example:
CATALOG DATABASE DB2DB390 AS CLIENTS390 AT NODE DB2NODE AUTHENTICATION SERVER

where:
- DB2DB390 is the name of the remote database to catalog
CLIENTS390 is the alias for the remote database being cataloged. If you do not specify an alias, the database manager uses the name of the remote database as the alias.

DB2NODE is the name of the node that you previously cataloged.

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

2. If the name of the remote database has more than eight characters, issue the CATALOG DCS DATABASE command to create a DCS directory entry.

For example:

```
CATALOG DCS DATABASE SALES400 AS SALES_DB2DB400
```

where:

- **SALES400** is the alias of the remote database to catalog. The alias must match the name of an entry in the federated server system database directory that is associated with the remote node. The alias is the same name that you specify in the CATALOG DATABASE command.
- **SALES_DB2DB400** is the name of target host database that you want to catalog.

### Registering the DB2 wrapper

You must register a wrapper to access DB2 family data sources. The federated server uses the wrapper to communicate with and retrieve data from the data sources. A wrapper is implemented as a set of library files.

**Procedure**

To register a wrapper:

Issue the CREATE WRAPPER statement and specify the name for the wrapper. The default wrapper name for the DB2 family data sources is DRDA.

For example:

```
CREATE WRAPPER DRDA
```

When you use the default name to register the wrapper, you do not need to specify the library name because the federated server automatically uses the default library name that is associated with the wrapper. If the wrapper name conflicts with an existing wrapper name in the federated database, you can replace the default wrapper name with a name that you choose. However, if you do so, you must include the LIBRARY parameter in the CREATE WRAPPER statement.

For example, to register a wrapper with the name db2_wrapper on a federated server that uses the AIX operating system, issue this statement:

```
CREATE WRAPPER db2_wrapper LIBRARY 'libdb2drda.a'
```

The default library name is specific to the operating system of the federated server. For more information, see the list of DB2 wrapper library files.

### DB2 wrapper library files

When you install the federated server, wrapper library files are added to the default directory path.
If you do not use the default wrapper name when you register a wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

This table lists the default directory paths and default wrapper library file names. In the table install_path in the directory path where the federation server is installed on UNIX or Linux, and %DB2PATH% is the environment variable that specifies the directory path where the federation server is installed on Microsoft Windows. The default Windows directory path is C:\Program Files\IBM\sqlib.

Table 21. DB2 directory paths and library file names by operating system

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Library file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/install_path/lib32/</td>
<td>libdb2drda.a</td>
</tr>
<tr>
<td></td>
<td>/usr/opt/install_path/lib64/</td>
<td></td>
</tr>
<tr>
<td>HP-UX</td>
<td>/opt/IBM/db2/install_path/lib32</td>
<td>libdb2drda.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/install_path/lib64</td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/IBM/db2/install_path/lib32</td>
<td>libdb2drda.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/install_path/lib64</td>
<td></td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/IBM/db2/install_path/lib32</td>
<td>libdb2drda.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/install_path/lib64</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>%DB2PATH%\bin</td>
<td>db2drda.dll</td>
</tr>
</tbody>
</table>

For each operating system, three library files installed: one default library file and two additional files that are used only with specific wrapper options.

Registering server definitions for DB2 data sources

The federated server requires authorization and password information to connect to each DB2 server. Because this authorization and password information is not stored in the global catalog, you must include it in the each server definition.

Procedure

To register a server definition, use one of these methods:

- Use the Federated Objects wizard in the DB2 Control Center. To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.
- Issue the CREATE SERVER statement.

When you register the server, you must include certain required server options. This example includes only the server options that are required to register a DB2 server:

```
CREATE SERVER server_definition_name TYPE server_type
VERSION version_number WRAPPER DRDA
  AUTHORIZATION "userid" PASSWORD "password"
  OPTIONS (DBNAME 'database_name')
```

DBNAME is a required server option. The value of DBNAME is the alias for the DB2 database that you want to access. You define the alias when you catalog the database.

Note: For the VERSION, if you used DB2 for z/OS® Version 8 to create the database in compatibility mode, you must specify Version 7.
When you register the server, you can specify additional server options in the CREATE SERVER statement. These options include general server options and server options that are specific to the data source. For more information, see the DB2 options reference information.

After you register the server, use the ALTER SERVER statement to add additional server options or drop existing server options.

CREATE SERVER statement - Examples for the DB2 wrapper
Use the CREATE SERVER statement to register DB2 server definitions. This topic includes a complete example with the required options, and an example that shows the use of additional server options.

**Complete example**

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

```
CREATE SERVER DB2SERVER TYPE DB2/ZOS VERSION 7 WRAPPER DRDA
  AUTHORIZATION "spalten" PASSWORD "db2guru"
  OPTIONS (DBNAME 'CLNTS390')
```

**DB2SERVER**
A name that you assign to the DB2 database server. Duplicate server definition names are not allowed.

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

**VERSION 7**
The version of the DB2 database server that you want to access.

*Note:* If you used DB2 for z/OS Version 8 to create the database in compatibility mode, you must specify Version 7.

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

**AUTHORIZATION "spalten"**
The authorization ID at the data source. This ID must have BINDADD authority at the data source. This value is case sensitive.

**PASSWORD "db2guru"**
The password that is associated with the authorization ID at the data source. This value is case sensitive.

**DBNAME 'CLNTS390'**
The alias for the DB2 database that you want to access. You defined this alias when you cataloged the database using the CATALOG DATABASE command. This value is case sensitive.

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

**Server option example**

When you register the server definition, you can specify additional server options in the CREATE SERVER statement. These options include general server options and DB2 data source-specific server options. For more information, see the options reference information.
The CPU_RATIO option indicates how much faster or slower the data source CPU runs than the federated CPU. If you set the CPU_RATIO option to '0.001', this indicates that the CPU at the remote data source has 1000 times more available capacity than the federated server CPU.

For example:
CREATE SERVER DB2SERVER TYPE DB2/ZOS VERSION 7 WRAPPER DRDA
  AUTHORIZATION "spalten" PASSWORD "db2guru"
  OPTIONS (DBNAME 'CLNTS390', CPU_RATIO '0.001')

Creating user mappings for DB2 data sources

A user mapping defines an association between a user ID and password at the federated server and the corresponding user ID and password at the data source server.

Whether or not user mappings are required for DB2 data sources depends on the configuration of the federated environment. If the environment uses federated trusted contexts and proxy authentication, none or only a few user mappings may be required. For best results, plan and set up federated trusted contexts before you create user mappings.

Procedure

To map the local user ID to the DB2 server user ID and password:

Issue a CREATE USER MAPPING statement.

For example:
CREATE USER MAPPING FOR local_userID SERVER server_definition_name
  OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password')

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

CREATE USER MAPPING statement - Examples for the DB2 wrapper

Use the CREATE USER MAPPING statement to map a federated server authorization ID to a DB2 server user ID and password.

Complete example

The following example shows how to map a federated server authorization ID to a remote DB2 user ID and password:
CREATE USER MAPPING FOR ALONZO SERVER DB2SERVER
  OPTIONS (REMOTE_AUTHID 'al', REMOTE_PASSWORD 'day2night')

ALONZO
Maps the local authorization ID to the remote user ID and password.

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

REMOTE_AUTHID 'al'
  Specifies the connect user ID at the DB2 family data source server to which
you are mapping ALONZO. The value is case-sensitive, unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

REMOTE_PASSWORD 'day2night'

Specifies the password that is associated with 'al'. The value is case-sensitive, unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

Special register example

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

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

You can use the DB2 special register USER to map the authorization ID of the person issuing the CREATE USER MAPPING statement to the data source user ID specified in the REMOTE_AUTHID user option.

Testing the connection to the DB2 data source server

Test the connection to the DB2 data source server to determine if the federated server is properly configured to access the DB2 data source server.

To test the connection to the DB2 server, open a pass-through session and issue an SQL SELECT statement on the DB2 system tables.

<table>
<thead>
<tr>
<th>DB2 data source</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 for z/OS</td>
<td>SET PASSTHRU server_definition_name</td>
</tr>
<tr>
<td></td>
<td>SELECT count(*) FROM sysibm.systables</td>
</tr>
<tr>
<td></td>
<td>SET PASSTHRU RESET</td>
</tr>
<tr>
<td>DB2 for Linux, UNIX and Windows</td>
<td>SET PASSTHRU server_definition_name</td>
</tr>
<tr>
<td></td>
<td>SELECT count(*) FROM syscat.systables</td>
</tr>
<tr>
<td></td>
<td>SET PASSTHRU RESET</td>
</tr>
<tr>
<td>DB2 for System i</td>
<td>SET PASSTHRU server_definition_name</td>
</tr>
<tr>
<td></td>
<td>SELECT count(*) FROM qsys2.systables</td>
</tr>
<tr>
<td></td>
<td>SET PASSTHRU RESET</td>
</tr>
</tbody>
</table>

If the SQL SELECT statement returns a count, access to the data source is properly configured.

Registering nicknames for DB2 tables and views

For each DB2 server definition, register a nickname for each table and view that you want to access. Then use the nicknames, not the names of the data source objects, when you query the DB2 database.

Before you begin

Before you register a nickname, use the DB2 RUNSTATS command to update the statistics at the DB2 data source. The federated server uses the data source statistics to optimize query processing.

Restrictions

You cannot create a nickname on a DB2 database alias.
Procedure

To register a nickname:

Issue the CREATE NICKNAME statement.

For example:

```
CREATE NICKNAME nickname FOR server_definition_name."remote_schema"."remote.table"
```

When you create the nickname, the federated server queries the data source catalog using the nickname. This query tests the connection to the data source. If the connection does not work, an error message displays.

**CREATE NICKNAME statement - Examples for DB2 data sources**

Use the CREATE NICKNAME statement and the required nickname options to register a nickname for an DB2 table or view that you want to access.

This example shows the use of the required options for the CREATE NICKNAME statement. You can also include additional nickname and column options.

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

**DB2SALES**

A unique nickname that identifies the DB2 table or view. The nickname can include both a schema and the nickname. If you omit the schema, the authorization ID of the user who registers the nickname is used.

**DB2SERVER.VINNIE.EUROPE**

A three-part identifier for the remote object:

- *DB2SERVER* is the name that you assigned to the DB2 database server in the CREATE SERVER statement.
- *VINNIE* is the user ID of the owner of the table or view. This value is case sensitive.
- *EUROPE* is the name of the remote table or view that you want to access.
Chapter 5. Configuring access to Entrez data sources

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

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

Entrez Wrapper

Entrez is a query and retrieval system developed by the National Center for Biotechnology Information (NCBI). You can use the Entrez wrapper to access several linked databases hosted by the NCBI.

You can access all of the Entrez databases through a uniform set of Web-based tools. The Entrez wrapper uses these tools to federate the Entrez databases.

Although the Entrez interface supports many databases, the Entrez wrapper supports access only to the following databases:
- Nucleotide. This is a sequence database also called GenBank.
- OMIM. The Online Mendelian Inheritance in Man database from John Hopkins University.
- PubMed. A database containing biomedical literature.
Many elements of the Entrez wrapper are common to all of the databases. These elements include:

- Connectivity with NCBI through the Web and the Entrez ESearch and EFetch utilities
- Mapping of hierarchical XML data into relational tables
- Joins between related tables through the XML wrapper technology
Adding Entrez data sources to a federated server

To configure a federated server to access Entrez data sources, you must register custom functions, a wrapper, a server definition, user mappings, and nicknames for the Entrez databases.

Before you begin

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

About this task

If your network uses a firewall, the Entrez wrapper can access the Nucleotide, OMIM, and PubMed databases if the network includes a proxy server. The Entrez wrapper supports the HTTP and SOCKS proxies.

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

Procedure

To add Entrez data sources to a federated server:

1. Register custom functions for the Entrez wrapper.
2. Register the Entrez wrapper.
3. Register the server definitions.
4. Register the user mappings (optional).
5. Register the nicknames.

Registering the custom functions for the Entrez wrapper

You must register the Entrez custom functions before you register the Entrez wrapper. The Entrez custom functions are used with the Entrez wrapper to send predicates to the Entrez query engine.

About this task

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

All of the custom functions for the Entrez wrapper must be registered with the schema name ENTREZ.

Tip: If you use the Federated Objects wizard in the Control Center to register the Entrez wrapper, server definitions, and nicknames, the Entrez custom functions are automatically created for you. Otherwise, you must register the custom functions from the command line.

Procedure

To register the Entrez custom functions, run the create_function_mappings.ddl file on each federated database instance where the Entrez wrapper is installed.
The create_function_mappings.ddl file is in the sqllib/samples/lifesci/entrez directory on the federated server. This file contains the CREATE FUNCTION statements for each of the custom functions for the Entrez wrapper.

The AS TEMPLATE, DETERMINISTIC, and NO EXTERNAL ACTION keywords are included in the CREATE FUNCTION statement in the create_function_mappings.ddl file.

**Custom functions for the Entrez wrapper**

You must include specific keywords when you register the Entrez custom functions

The federated environment uses two query engines. For the Entrez wrapper, these query engines are the federated database query engine and the Entrez query engine. You can specify that predicates should be sent to the Entrez engine by using the Entrez custom functions in the WHERE clause of your SELECT statement.

The custom functions for the Entrez wrapper are:

- ENTREZ.CONTAINS
- ENTREZ.SEARCH_TERM

A separate ENTREZ custom function must be registered for each valid data type for the column_name argument. The data types that you can specify for the ENTREZ custom functions are listed in the following table:

**Table 22. Valid data types for the ENTREZ custom functions**

<table>
<thead>
<tr>
<th>ENTREZ.CONTAINS custom function</th>
<th>ENTREZ.SEARCH_TERM custom function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTREZ.CONTAINS (VARCHAR(), VARCHAR())</td>
<td>ENTREZ.SEARCH_TERM (CHAR(), VARCHAR())</td>
</tr>
<tr>
<td>ENTREZ.CONTAINS (INTEGER, VARCHAR())</td>
<td></td>
</tr>
<tr>
<td>ENTREZ.CONTAINS (SMALLINT, VARCHAR())</td>
<td></td>
</tr>
<tr>
<td>ENTREZ.CONTAINS (REAL, VARCHAR())</td>
<td></td>
</tr>
<tr>
<td>ENTREZ.CONTAINS (DOUBLE, VARCHAR())</td>
<td></td>
</tr>
<tr>
<td>ENTREZ.CONTAINS (DATE, VARCHAR())</td>
<td></td>
</tr>
<tr>
<td>ENTREZ.CONTAINS (TIME, VARCHAR())</td>
<td></td>
</tr>
<tr>
<td>ENTREZ.CONTAINS (CHAR(), VARCHAR())</td>
<td></td>
</tr>
<tr>
<td>ENTREZ.CONTAINS (TIMESTAMP(), VARCHAR())</td>
<td></td>
</tr>
</tbody>
</table>

**ENTREZ.CONTAINS custom function**

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

```sql
CREATE FUNCTION ENTREZ.CONTAINS
(column_name_data_type, search_value_data_type)
RETURNS INTEGER AS TEMPLATE
DETERMINISTIC NO EXTERNAL ACTION;
```
ENTREZ.SEARCH_TERM custom function

The following example shows the syntax for the ENTREZ.SEARCH_TERM custom function:

CREATE FUNCTION ENTREZ.SEARCH_TERM
  (parent.nickname_primary_key data_type, search_value data_type)
  RETURNS INTEGER AS TEMPLATE
  DETERMINISTIC NO EXTERNAL ACTION;

Registering the Entrez wrapper

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

If you use a proxy server to access Entrez files, you must specify the proxy information as options when you register the wrapper. By default, wrapper options are used when you query any Entrez file. However, if you also specify proxy options when you register a server definition, the server options take precedence.

Procedure

To register the Entrez wrapper:

Procedure

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>From the command line</td>
<td>Issue the CREATE WRAPPER statement. If you use a proxy server to access Entrez documents, you issue this statement:</td>
</tr>
<tr>
<td></td>
<td>CREATE WRAPPER wrapper_name LIBRARY library_name OPTIONS(EMAIL 'email_address');</td>
</tr>
<tr>
<td></td>
<td>OPTIONS (EMAIL 'email_address', PROXY_TYPE 'type', PROXY_SERVER_NAME 'server_name', PROXY_SERVER_PORT 'port_number');</td>
</tr>
</tbody>
</table>

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

You must specify an e-mail address when you register the wrapper. The e-mail address that you specify is included with all queries and allows NCBI to contact you if there are problems, such as too many queries overloading the NCBI servers.

Entrez wrapper library files

The Entrez wrapper library files are added to the federated server when you install IBM WebSphere Federation Server.
When you install IBM WebSphere Federation Server, three library files are added to the default directory path for the Entrez wrapper. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2lsentrez.a, libdb2lsentrezF.a, and libdb2lsentrezU.a.

The default wrapper library file is libdb2lsentrez.a. The other two wrapper library files are used with specific wrapper options.

When you register the Entrez wrapper from the command line, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

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

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Default directory path</th>
<th>Default wrapper library file</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/&lt;install_path&gt;/lib32/</td>
<td>libdb2lsentrez.a</td>
</tr>
<tr>
<td></td>
<td>/usr/opt/&lt;install_path&gt;/lib64/</td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib32</td>
<td>libdb2lsentrez.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib64</td>
<td></td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib32</td>
<td>libdb2lsentrez.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib64</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>%DB2PATH%\bin</td>
<td>db2lsentrez.dll</td>
</tr>
</tbody>
</table>

<install_path> is the directory path where the federated server is installed on UNIX or Linux.

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

**CREATE WRAPPER statement - examples for the Entrez wrapper**

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

**Registering a wrapper**

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

```sql
CREATE WRAPPER entrez_wrapper LIBRARY 'libdb2lsentrez.a';
```

`entrez_wrapper`

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

**LIBRARY 'libdb2lsentrez.a'**

The name of the wrapper library file for federated servers that use AIX operating systems.
Registering a wrapper for an HTTP proxy server

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

```
CREATE WRAPPER entrez_wrapper LIBRARY 'libdb2lsentrez.a'
  OPTIONS (PROXY_TYPE 'HTTP', PROXY_SERVER_NAME 'proxy_http',
           PROXY_SERVER_PORT '8080');
```

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

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

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

Registering a wrapper for a SOCKS proxy server

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

```
CREATE WRAPPER entrez_wrapper LIBRARY 'libdb2lsentrez.so'
  OPTIONS (PROXY_TYPE 'SOCKS', PROXY_SERVER_NAME 'proxy_socks',
           PROXY_SERVER_PORT '1081');
```

**LIBRARY 'libdb2lsentrez.so'**
The name of the wrapper library file for federated servers that use Linux and Solaris operating systems.

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

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

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

Registering the server definition for Entrez data sources

You must register a server definition for each Entrez database that you want to access. A server definition is required when you register the nicknames for the Entrez databases.

**About this task**

Specify the database that you want to access in the TYPE parameter. The value for the server type controls the structure of the nicknames that are created. The valid values for the TYPE parameter are Nucleotide, OMIM, and PubMed.
You can register a server definition by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps that are necessary to register the server definition.

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

Procedure

To register a server definition for a Entrez data source:

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>From the command line</td>
<td>Issue the CREATE SERVER statement. For example:</td>
</tr>
<tr>
<td></td>
<td>CREATE SERVER server_definition_name</td>
</tr>
<tr>
<td></td>
<td>TYPE database_name</td>
</tr>
<tr>
<td></td>
<td>WRAPPER wrapper_name</td>
</tr>
<tr>
<td></td>
<td>If you use a proxy server to access Entrez files, you must specify several options when you register either the Entrez wrapper or the server definition. To specify the proxy server information when you register the Entrez wrapper, the statement that you issue is: CREATE SERVER server_definition_name</td>
</tr>
<tr>
<td></td>
<td>TYPE database_name</td>
</tr>
<tr>
<td></td>
<td>WRAPPER wrapper_name</td>
</tr>
<tr>
<td></td>
<td>OPTIONS (SOCKET_TIMEOUT 'minutes',</td>
</tr>
<tr>
<td></td>
<td>PROXY_TYPE 'proxy_server_type',</td>
</tr>
<tr>
<td></td>
<td>PROXY_SERVER_NAME 'proxy_server_name',</td>
</tr>
<tr>
<td></td>
<td>PROXY_SERVER_PORT 'port_number');</td>
</tr>
</tbody>
</table>

CREATE SERVER statement - Examples for the Entrez wrapper

Use the CREATE SERVER statement to register server definitions for the Entrez wrapper. This topic includes an example with the required parameters, and an example with the server options required for proxy servers.

Required parameters

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

CREATE_SERVER nucleotide_server1
    TYPE Nucleotide
    WRAPPER entrez_wrapper;

nucleotide_server1
    A name that you assign to the Entrez server. Duplicate server definition names are not allowed.

TYPE Nucleotide
    Specifies the type of the data source. The valid values for server type are Nucleotide, OMIM, and PubMed. These values are case-insensitive.
WRAPPER entrez_wrapper
  Specifies the name of the wrapper that you registered in the CREATE
  WRAPPER statement.

Sample server definition for OMIM data sources

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

CREATE SERVER omim_server1
  TYPE OMIM
  WRAPPER entrez_wrapper;

  omim_server1
    A name that you assign to the Entrez server. Duplicate server definition
    names are not allowed.

TYPE OMIM
  Specifies the type of the data source. The valid values for server type are
  Nucleotide, OMIM, and PubMed. These values are case-insensitive.

WRAPPER entrez_wrapper
  Specifies the name of the wrapper that you registered in the CREATE
  WRAPPER statement.

Proxy server example

You can access Entrez data sources through a proxy server. The following example
registers a server definition for a proxy server that uses the SOCKS protocol with
authentication information.

CREATE SERVER pubmed_server_s5
  TYPE PUBMED
  VERSION 1.0
  WRAPPER entrez_wrapper
  OPTIONS (PROXY_TYPE 'SOCKS', PROXY_SERVER_NAME 'proxy_5',
    PROXY_SERVER_PORT '1081', PROXY_AUTHID 'Khalid',
    PROXY_PASSWORD 'good2go');

  pubmed_server_s5
    A name that you assign to the Entrez server. Duplicate server definition
    names are not allowed.

TYPE PUBMED
  Specifies the type of the data source. The valid values for the server type
  are Nucleotide, OMIM, and PubMed. These values are case-insensitive.

VERSION 1.0
  Specifies the version of the NCBI XML schema that you are using. This
  option is not required. If you do not specify the version of the schema, the
  default value is 1.0.

WRAPPER entrez_wrapper
  Specifies the name of the wrapper that you registered in the CREATE
  WRAPPER statement.

PROXY_TYPE 'SOCKS'
  Specifies the type of proxy server. This value must be enclosed in single
  quotation marks.

PROXY_SERVER_NAME 'proxy_5'
  Specifies the name of the proxy server. This value must be enclosed in
  single quotation marks.
**PROXY_SERVER_PORT '1081'**
Specifies the number of the port for the proxy server. This value must be enclosed in single quotation marks.

**PROXY_AUTHID 'Khalid'**
Specifies the user ID on the proxy server. This value must be enclosed in single quotation marks. This value is case sensitive.

**PROXY_PASSWORD 'good2go'**
Specifies the password on the proxy server that is associated with 'Khalid'. This value must be enclosed in single quotation marks. This value is case sensitive.

---

**Creating the user mappings for a Entrez data source (optional)**

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

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

**About this task**

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

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

**Procedure**

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

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Using the command line</td>
<td>Issue the CREATE USER MAPPING statement. For example: CREATE USER MAPPING FOR local_userID SERVER server_definition_name OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password') PROXY_AUTHID 'proxy_server_userID', PROXY_PASSWORD 'proxy_server_password';</td>
</tr>
</tbody>
</table>

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

**CREATE USER MAPPING statement - Examples for the Entrez wrapper**

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

You can create a user mapping by specifying a proxy server. The following example shows how to map a federated authorization ID to a Entrez proxy server user ID and password:

```
CREATE USER MAPPING FOR lucille SERVER blast_proxy
    OPTIONS (PROXY_AUTHID 'lball'
        PROXY_PASSWORD 'lball');
```

`lucille` Specifies the local authorization ID that you are mapping to a user ID that is defined at the Entrez proxy server.

`SERVER blast_proxy` Specifies the server definition name that you registered in the CREATE SERVER statement for the Entrez server.

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

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

**Access Entrez using a proxy server**

If your network uses a proxy server, you must specify information about the proxy server when you register the wrapper or server definition for Entrez data sources.

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

You can specify the proxy options when you register a wrapper or a server definition:

- **If you specify these options when you register the wrapper**, the nicknames that are associated with that wrapper will use the options that are set for the wrapper.
- **If you specify these options when you register a server definition**, the nicknames that are associated with that server definition will use the options that are set for the server definition.
- **If you specify different values for these options when you register the wrapper and the server definition**, the values that are set for the server definition take precedence over the values that are set for the wrapper.
Proxy server options

The following table lists the options that must you specify for each type of proxy server:

Table 25. Options that are required with proxy servers

<table>
<thead>
<tr>
<th>Type of proxy server</th>
<th>Required wrapper or server options</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP or SOCKS without authentication</td>
<td>PROXY_TYPE</td>
</tr>
<tr>
<td></td>
<td>PROXY_SERVER_NAME</td>
</tr>
<tr>
<td></td>
<td>PROXY_SERVER_PORT</td>
</tr>
<tr>
<td>HTTP or SOCKS with authentication</td>
<td>PROXY_TYPE</td>
</tr>
<tr>
<td></td>
<td>PROXY_SERVER_NAME</td>
</tr>
<tr>
<td></td>
<td>PROXY_SERVER_PORT</td>
</tr>
<tr>
<td></td>
<td>PROXY_AUTHID</td>
</tr>
<tr>
<td></td>
<td>PROXY_PASSWORD</td>
</tr>
</tbody>
</table>

In addition to the required server options for proxy servers, you can also specify the SOCKET_TIMEOUT server option when you register a server definition. The SOCKET_TIMEOUT server option specifies the maximum amount of time, in minutes, that the federated server waits for the results from the proxy server. If you do not specify the SOCKET_TIMEOUT server option, there is no time limit. The federated server will wait indefinitely for the results from the proxy server.

Nicknames for the Entrez data sources

To access the data in the Entrez databases, you must register a fixed list of nicknames for each database that you want to access.

Each Entrez database has a fixed set of tables that are organized in a hierarchy. Each table has a fixed set of fields. One table, which is the parent of all of the other tables in the database, is called the root or parent table. All of the other tables in the database have a parent-child relationship that leads back to the root table.

The nicknames for the Entrez databases use a similar hierarchical structure and each nickname contains a fixed list of columns.

You can use the predefined set of names for the nicknames or you can specify alternative names for the nicknames.

Nicknames for the Nucleotide database

The fields in the Nucleotide database are mapped to a set of nicknames that are stored on the federated server.

When the Entrez wrapper sends your query to the Nucleotide database, the data that is returned from the Nucleotide database is in the XML format. The Entrez wrapper maps the hierarchical structure of the XML data to the Nucleotide nicknames.

You can use the default names for the nicknames

Each Nucleotide nickname represents a group of fields in the Nucleotide database. Each nickname contains a fixed list of columns and the nicknames are organized in a hierarchical relationship.
The top of the hierarchy of nicknames is the GBSeq nickname. The GBSeq nickname is the parent of all of the other nicknames and is called the root nickname. All of the other nicknames have parent-child relationships that lead back to the root nickname.

The following table shows the relationship between the GBSeq nickname and the other nicknames in the Nucleotide hierarchy of nicknames.

Table 26. The Nucleotide hierarchy of nicknames

| Nickname     | Parent
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GBSeq</td>
<td>None. GBSeq is the root nickname.</td>
</tr>
<tr>
<td>GBFeatures</td>
<td>GBSeq</td>
</tr>
<tr>
<td>GBIIntervals</td>
<td>GBFeatures</td>
</tr>
<tr>
<td>GBQualifiers</td>
<td>GBFeatures</td>
</tr>
<tr>
<td>GBReference</td>
<td>GBSeq</td>
</tr>
</tbody>
</table>

If you want to query a child nickname, you must create joins between each child nickname and the parent nickname.

**Nicknames for the OMIM database**

The fields in the OMIM database are mapped to a set of nicknames that are stored on the federated server.

When the Entrez wrapper sends your query to the OMIM database, the data that is returned from the OMIM database is in the XML format. The Entrez wrapper maps the hierarchical structure of the XML data to the OMIM nicknames.

Each OMIM nickname represents a group of fields in the OMIM database. Each nickname contains a fixed list of columns and the nicknames are organized in a hierarchical relationship.

The top of the hierarchy of nicknames is the OMIM nickname. The OMIM nickname is the parent of all of the other nicknames and is called the root nickname. All of the other nicknames have parent-child relationships that lead back to the root nickname.
The following table shows the relationship between the OMIM nickname and the other nicknames in the OMIM hierarchy of nicknames.

<table>
<thead>
<tr>
<th>Nickname</th>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMIM</td>
<td>None. OMIM is the root nickname.</td>
</tr>
<tr>
<td>OMIM_Alias</td>
<td>OMIM</td>
</tr>
<tr>
<td>OMIM_Allelic_Variant</td>
<td>OMIM</td>
</tr>
<tr>
<td>OMIM_Allelic_Variant_Description</td>
<td>OMIM_Allelic_Variant</td>
</tr>
<tr>
<td>OMIM_ALPHA</td>
<td>OMIM</td>
</tr>
<tr>
<td>OMIM_Edit_History</td>
<td>OMIM</td>
</tr>
<tr>
<td>OMIM_Claim</td>
<td>OMIM</td>
</tr>
<tr>
<td>OMIM_Claim_Validation</td>
<td>OMIM</td>
</tr>
<tr>
<td>OMIM_Claim_Validation_UID</td>
<td>OMIM_Claim_Validation</td>
</tr>
<tr>
<td>OMIM_Claim_Validation_UID</td>
<td>OMIM_Claim_Validation</td>
</tr>
<tr>
<td>OMIM_Link</td>
<td>OMIM</td>
</tr>
<tr>
<td>OMIM_Link_UID</td>
<td>OMIM</td>
</tr>
</tbody>
</table>

Figure 6. Hierarchy of OMIM nicknames
Table 27. The OMIM hierarchy of nicknames (continued)

<table>
<thead>
<tr>
<th>Nickname</th>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMIM_Allelic_Variant_Description</td>
<td>OMIM_Allelic_Variant</td>
</tr>
<tr>
<td>OMIM_Allelic_Variant_Description_UID</td>
<td>OMIM_Allelic_Variant_Description</td>
</tr>
<tr>
<td>OMIM_Clinical_Synopsis</td>
<td>OMIM</td>
</tr>
<tr>
<td>OMIM_Contributors</td>
<td>OMIM</td>
</tr>
<tr>
<td>OMIM_Edit_History</td>
<td>OMIM</td>
</tr>
<tr>
<td>OMIM_Links</td>
<td>OMIM</td>
</tr>
<tr>
<td>OMIM_Links_UID</td>
<td>OMIM_Links</td>
</tr>
<tr>
<td>OMIM_References</td>
<td>OMIM</td>
</tr>
<tr>
<td>OMIM_See_Also</td>
<td>OMIM</td>
</tr>
<tr>
<td>OMIM_Text</td>
<td>OMIM</td>
</tr>
<tr>
<td>OMIM_Text_UID</td>
<td>OMIM_Text</td>
</tr>
</tbody>
</table>

If you want to query a child nickname, you must create joins between each child nickname and the corresponding parent nickname, up to the root nickname.

For example, if you want to query the OMIM_Allelic_Variant_Description nickname, you must join the OMIM_Allelic_Variant_Description nickname (child) to the OMIM_Allelic_Variant nickname (parent) and join the OMIM_Allelic_Variant nickname (child) to the OMIM nickname (parent).

**Nicknames for the PubMed database**

The fields in the PubMed database are mapped to a set of nicknames that are stored on the federated server.

When the Entrez wrapper sends your query to the PubMed database, the data that is returned from the PubMed database is in the XML format. The Entrez wrapper maps the hierarchical structure of the XML data to the PubMed nicknames.

Each PubMed nickname represents a group of fields in the PubMed database. Each nickname contains a fixed list of columns and the nicknames are organized in a hierarchical relationship.

The top of the hierarchy of nicknames is the PMARTICLES nickname. The PMARTICLES nickname is the parent of all of the other nicknames and is called the root nickname. All of the other nicknames have parent-child relationships that lead back to the root nickname.

The following table shows the relationship between the PMARTICLES nickname and the other nicknames in the PubMed hierarchy of nicknames.

```
CREATE NICKNAME PMARTICLES FOR SERVER pubmed_server;
CREATE NICKNAME PMACCESSION FOR SERVER pubmed_server;
CREATE NICKNAME PMCHEMICAL FOR SERVER pubmed_server;
CREATE NICKNAME PMMESH FOR SERVER pubmed_server;
CREATE NICKNAME PMCOMMENTS FOR SERVER pubmed_server;
CREATE NICKNAME PMARTICLEID FOR SERVER pubmed_server;
```

Table 28. The PubMed hierarchy of nicknames

<table>
<thead>
<tr>
<th>Nickname</th>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMARTICLES</td>
<td>None. PMARTICLES is the root nickname.</td>
</tr>
</tbody>
</table>
Table 28. The PubMed hierarchy of nicknames (continued)

<table>
<thead>
<tr>
<th>Nickname</th>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMACCESSION</td>
<td>PMARTICLES</td>
</tr>
<tr>
<td>PMCHEMICAL</td>
<td>PMARTICLES</td>
</tr>
<tr>
<td>PMMESH</td>
<td>PMARTICLES</td>
</tr>
<tr>
<td>PMCOMMENTS</td>
<td>PMARTICLES</td>
</tr>
<tr>
<td>PMARTICLEID</td>
<td>PMARTICLES</td>
</tr>
</tbody>
</table>

If you want to query a child nickname, you must create joins between each child nickname and the parent nickname.

**Registering nicknames for Entrez data sources**

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

**Before you begin**

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

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

**Procedure**

To register a nickname for an Entrez database:
Choose the method that you want to use to register the nickname:

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
</table>
| Using the Control Center | Start the Federated Objects wizard from the **Federated Database Objects** folder, or open the **Nicknames** folder.  
1. Click **Discover** and use the Discover window to assist you in generating the nicknames for the Entrez database.  
2. In the **Schema** field, specify the schema in which you want to register the nicknames.  
3. You can use the predefined set of names for the nicknames or you can specify other names for the nicknames:  
   - If you use the predefined set of names for the nicknames, a text string suffix is automatically added to the end of nicknames. You can replace the default suffix in the **Nickname suffix** field.  
   - To specify other names for the nicknames, click the **Define nicknames** radio button and specify a nickname for each of the remote objects. You will also need to specify the required nickname options. For a root nickname, only the REMOTE_OBJECT nickname option is required. For all of the other nicknames, you must specify both the REMOTE_OBJECT and the PARENT nickname options.  
4. Click **OK** to return to the Define Nicknames page in the Federated Objects wizard. The nicknames are listed in the **Nicknames** table.  
5. Click **Finish** to return to the Create Nicknames window in the Federated Objects wizard. The nicknames are listed in the **Nicknames** table. If you did not already create the Entrez custom functions, the custom functions are created for you and appear on the Additional Objects page of the Federated Objects wizard. After you complete the wizard, the function templates are listed in the **User Defined Functions** folder that is under the **Application Objects** folder in the Control Center.  
6. Click **Finish** to complete the wizard and create the nicknames. |
Repeat these steps for each Entrez database that you want to create nicknames for.

**CREATE NICKNAME statement - examples for the Entrez wrapper:**

Use the CREATE NICKNAME statement to register the nicknames for each Entrez database that you want to access. There are a set of predefined nicknames that you create for the Entrez databases. You can use the default names for the nicknames or specify alternative names.

The predefined set of CREATE NICKNAME statements that are necessary to register the nicknames for each of the Entrez databases are specified in the following lists. You can edit these statements to specify the name of the server definition that you used in the CREATE SERVER statement. Issue these CREATE NICKNAME statements from the command line to register the nicknames.

If you use the Control Center to register the nicknames, the statements are automatically generated and issued for you.

**Nucleotide nicknames**

The following list contains the CREATE NICKNAME statements that are necessary to register the nicknames for the Nucleotide tables:

```
CREATE NICKNAME GBSeq FOR SERVER nuc1;
CREATE NICKNAME GBFeatures FOR SERVER nuc1;
CREATE NICKNAME GBIntervals FOR SERVER nuc1;
CREATE NICKNAME GBQualifiers FOR SERVER nuc1;
CREATE NICKNAME GBReference FOR SERVER nuc1;
```

**OMIM nicknames**

The following list contains the CREATE NICKNAME statements that are necessary to register the nicknames for the OMIM tables:

```
CREATE NICKNAME omim FOR SERVER omim_server1;
CREATE NICKNAME omim_alias FOR SERVER omim_server1;
CREATE NICKNAME omim_text FOR SERVER omim_server1;
CREATE NICKNAME omim_text_uid FOR SERVER omim_server1;
CREATE NICKNAME omim_see_also FOR SERVER omim_server1;
CREATE NICKNAME omim_allelic_variant FOR SERVER omim_server1;
CREATE NICKNAME omim_allelic_variant_desc FOR SERVER omim_server1;
```
CREATE NICKNAME omim_allelic_variant_desc_uid FOR SERVER omim_server1;
CREATE NICKNAME omim_clinical_synopsis FOR SERVER omim_server1;
CREATE NICKNAME omim_edit_history FOR SERVER omim_server1;
CREATE NICKNAME omim_references FOR SERVER omim_server1;
CREATE NICKNAME omim_contributors FOR SERVER omim_server1;
CREATE NICKNAME omim_links FOR SERVER omim_server1;
CREATE NICKNAME omim_links_uid FOR SERVER omim_server1;

**PubMed nicknames**

The following list contains the CREATE NICKNAME statements that are necessary to register the nicknames for the PubMed tables:

CREATE NICKNAME PMARTICLES FOR SERVER pubmed_server;
CREATE NICKNAME PMACCESSION FOR SERVER pubmed_server;
CREATE NICKNAME PMCHEMICAL FOR SERVER pubmed_server;
CREATE NICKNAME PMMESH FOR SERVER pubmed_server;
CREATE NICKNAME PMCOMMENTS FOR SERVER pubmed_server;
CREATE NICKNAME PMARTICLEID FOR SERVER pubmed_server;

**Alternative names for the Entrez nicknames**

If the tables on your federated server use one or more of the predefined set of names for the Entrez nicknames, you must specify alternative names for the Entrez nicknames.

When you specify names for the Entrez nicknames other than the predefined set of names, you must include nickname options in the CREATE NICKNAME statements. These nickname options identify the predefined nickname that is associated with the nickname that are registering.

The REMOTE_OBJECT nickname option specifies the predefined name for the Entrez nickname. You must always include the REMOTE_OBJECT nickname option whenever you specify other names for the Entrez nicknames.

For the root nickname, you must specify only the REMOTE_OBJECT nickname option. For all other nicknames, you must specify both the REMOTE_OBJECT and PARENT nickname options.

**Specifying an alternative name for the root nickname**

For example, if a table on your federated server has the name OMIM, you need to register the root nickname OMIM with another name such as omim_nn:

```sql
CREATE NICKNAME omim_nn FOR SERVER omim_server1
  OPTIONS (REMOTE_OBJECT 'omim');
```

**Specifying an alternative name for a child nickname**

If a table on your federated server has the name OMIM_ALLELIC_VARIANT, you need to register the OMIM_ALLELIC_VARIANT nickname with another name such as omim_av_nn. You must identify the predefined name for the nickname in the REMOTE_OBJECT nickname option. You must also identify the parent nickname in the PARENT nickname option, which is the root nickname. For example:

```sql
CREATE NICKNAME omim_av_nn FOR SERVER omim_server1
  OPTIONS (REMOTE_OBJECT 'omim_allelic_variant', PARENT 'OMIM_NN');
```

You must always include the PARENT nickname option when the nickname that you are registering is a child nickname. The federated server changes the parent
nickname to uppercase when you register the nickname, unless you enclose the name of the nickname in double quotation marks.

For example:

- You specify the name _omim_nn_ when you register the parent nickname. You must specify _OMIM_NN_ for the value of the PARENT nickname option when you register the child nickname.
- You specify the name _"omim_nn"_ when you register the parent nickname. You must specify _"omim_nn"_ for the value of the PARENT nickname option when you register the child nickname.

**Specifying an alternative name for a child nickname that has a predefined parent nickname**

If you used the predefined name when you registered a parent nickname, you specify the predefined name in the PARENT nickname option:

```
CREATE NICKNAME _omim_av_nn_ FOR SERVER _omim_server1_
    OPTIONS (REMOTE_OBJECT _omim_allelic_variant_, PARENT _OMIM_);
```

---

**Queries and custom functions for Entrez data sources**

The Entrez wrapper includes custom functions that you can use in your queries.

**Query the Entrez databases using the custom functions**

The Entrez custom functions are used with the Entrez wrapper to send predicates to the Entrez query engine.

The custom functions that are provided with the Entrez wrapper must be registered on the federated server before you can use the functions in your queries.

**ENTREZ.CONTAINS function**

The ENTREZ.CONTAINS function requires two arguments. The first argument is the name of a fixed nickname column that can be searched. The second argument is the value that you want to search for.

The syntax for the ENTREZ.CONTAINS function is:

```
ENTREZ.CONTAINS (nickname_column, query_term)
```

In some situations, the predicates processed by the federated server and the predicates processed by the Entrez database server might be mixed in such a way that they cannot be processed. These cases generate the error message SQL0142N ("SQL statement not supported").

For example, in the following query fragment you cannot separate the parts of the predicate that are processed by the wrapper (the ENTREZ.CONTAINS invocations) and the parts that must be processed by the federated server (the relational predicate on BaseCountA).

```
WHERE
  OMIM.CONTAINS (Organism, 'helicobacter pylori') = 1
  OR (BaseCountA > 10 AND OMIM.CONTAINS (Keywords, 'cellular vacuolation ') = 1)
```

The nicknames that you register for the Entrez databases have a set of fixed columns. The only column names that you can specify in the ENTREZ.CONTAINS functions are the column names that have a search tag qualifier. If you specify a
column name that does not have a search tag qualifier, the error message SQL30090N ("Operation invalid for application execution environment") is returned.

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

If you want to search multiple columns, you must include a separate ENTREZ.CONTAINS function in your query for each column that you want to search on. Use Boolean operators (OR and AND) and parentheses in your query to specify the relationship between each ENTREZ.CONTAINS function. Search term qualifiers, such as [pd], are not allowed.

The ENTREZ.CONTAINS function returns an integer result. When you use the ENTREZ.CONTAINS function, the return value must be compared to the value 1 by using the = or <> operators. For example:

```
SELECT * FROM NewGBSeq
  WHERE ENTREZ.CONTAINS (Organism, 'drosophila') = 1;
```

**Pseudo columns**

Some search fields do not have corresponding columns in the Entrez schema. For example, in the Nucleotide database, the term [ALL] searches all searchable fields, while [WORD] searches all of the free text associated with a record. The AllText and FreeText columns for the GBSeq nickname are examples of pseudo columns and are provided for these search terms. If a pseudo column is referenced in a select list, a NULL value is returned.

The following example shows a query with a WHERE clause on a PubMed nickname. In this example, the individual predicates are authors, title, and alltext. The alltext pseudo column is used with the ENTREZ.CONTAINS function to search all searchable fields for the terms drosophila or fruit fly.

```
WHERE
  ENTREZ.CONTAINS (authors, 'kaufmann OR ito AND NOT rakesh')
AND
  (ENTREZ.CONTAINS (title, 'drosophila')
  OR
    ENTREZ.CONTAINS (alltext, 'drosophila OR "fruit fly"'))
```

**ENTREZ.SEARCH_TERM custom function**

The ENTREZ.SEARCH_TERM function requires two arguments. The first argument is the name of a fixed nickname column that is the primary key column for the parent nickname. The second argument is the value that you want to search for.

The syntax for the ENTREZ.SEARCH_TERM function is:

```
ENTREZ.SEARCH_TERM (nickname_column,query_term)
```

You can run queries that might not otherwise be possible by issuing the ENTREZ.SEARCH_TERM custom function. When you use the ENTREZ.SEARCH_TERM function, it must be the only custom function in a query.
The nickname column argument must be the primary key column for the parent nickname. The query term argument is an ENTREZ-format search term that includes search field qualifiers. This text is passed unmodified, except for URI escapes as required by the URI syntax, to ENTREZ.

The ENTREZ.SEARCH_TERM function returns an integer result. When you use the ENTREZ.SEARCH_TERM custom function, the return value must be compared to the value 1 by using the equal (=) or not equal (<> or =) operators. For example:

```
SELECT mim_id, SUBSTR(title,1,40) FROM omm
   WHERE entrez.search_term(mim_id, 'CYSTIC FIBROSIS') = 1;
```

**Relational predicates for the Entrez wrapper**

The Entrez wrapper supports relational predicates in queries. Some of the predicates are processed by the Entrez search engine and some are processed by the federated server.

The Entrez search engine processes equal and IN predicates on the ID column for each schema. These predicates allow the Entrez wrapper to bypass the search phase of a query process and execute the fetch phase of a query process. Examples of valid predicates are:

```
WHERE pmid = '1234567'
WHERE medlineid IN ('1234567', '9191919')
WHERE mim_id IN ('602421', '608583')
```

You can specify only the columns that have ID as the search tag for the nickname column in the WHERE clause of your query.

**Invalid WHERE clauses for the Entrez wrapper**

The Entrez wrapper rejects any query that will result in an unqualified scan of the NCBI database. A valid WHERE clause must contain either a custom function, or an equal or IN predicate on the primary ID for the schema. Queries that do not meet these criteria are rejected with error code SQL0142N or SQL30090N.

**Records returned from Entrez queries**

The MAX_ROWS server option can be used to limit the number of rows that are returned from a query on the Entrez data sources.

The MAX_ROWS server option value is always used as an upper (maximum) limit to the number of records for the root nickname that a query can return. If a query attempts to retrieve more records than what is specified in the MAX_ROWS server option, the result set is truncated. You are not notified that the result set is truncated.

For example, if you set the MAX_ROWS server option to 25, a maximum of 25 records for the root nickname and all of the records for the child-related nicknames are returned.

You can set the MAX_ROWS server option when you register a server definition with the CREATE SERVER statement, or you can use the ALTER SERVER statement to set or change the option value.
The MAX_ROWS server option is not required. If you do not set this option, a default value is used. The specific default value that is used depends on your operating system.

- For federated servers that run UNIX, the default is 5000 records.
- For federated servers that run Windows, the default is 2000 records.

You can specify only positive numbers and 0. When you set the MAX ROWS server option to 0, you enable queries to retrieve an unlimited number of records for the root nickname from the NCBI Web site. However, setting the MAX_ROWS server option to 0 or to a very high number might impact your query performance.

### Entrez data sources - Example queries

These examples show you how to query the Nucleotide, OMIM, and PubMed databases.

#### Queries for Nucleotide nicknames

The following query uses multiple fetch keys on a Nucleotide nickname:

```sql
SELECT PrimaryAccession, LocusName, SeqLength FROM GBSEQ
    WHERE PrimaryAccession IN ('NM_000890', 'NC_003106');
```

The following query searches for all of the searchable fields on a Nucleotide nickname:

```sql
SELECT PrimaryAccession, substr(Definition,1,300), GI FROM GBSEQ
    WHERE ENTREZ.CONTAINS(AllText, 'abcde')=1;
```

The following query searches for all of the free text on a Nucleotide nickname:

```sql
SELECT * FROM GBSEQ WHERE ENTREZ.CONTAINS(FreeText, 'abcde')=1;
```

The following query searches for a definition on a Nucleotide nickname:

```sql
SELECT PrimaryAccession, substr(Definition,1,300), version, GI FROM GBSEQ
    WHERE ENTREZ.CONTAINS(Definition, 'Sulfolobus tokodaii AND complete genome') = 1;
```

The following query searches for a keyword on a Nucleotide nickname:

```sql
SELECT PrimaryAccession, substr(KeywordList,1,200), Segment FROM GBSEQ
    WHERE ENTREZ.CONTAINS(KeywordList, 'nkcc1 gene') = 1;
```

#### Queries for OMIM nicknames

The following query returns the MIM identification number and the titles (up to 40 characters) of the OMIM records from the OMIM nickname, for a single MIM number:

```sql
SELECT mim_id, SUBSTR(title,1,40) FROM OMIM
    WHERE mim_id = '603855';
```

The following query returns the MIM identification numbers and the titles (up to 40 characters) of the OMIM records from the OMIM nickname, for multiple MIM identification numbers:

```sql
SELECT mim_id, SUBSTR(title,1,40) FROM OMIM
    WHERE mim_id IN ('603855', '605382', '600737', '605637', '219700');
```

The following query uses a CONTAINS function to look up all of the OMIM entries that contain CYSTIC FIBROSIS or CF in the title of the entry:
SELECT mim_id, SUBSTR(title,1,40) FROM OMIM
    WHERE OMIM.CONTAINS(title, 'CYSTIC FIBROSIS; CF') = 1;

The following query uses two CONTAINS functions to look up all of the OMIM entries that contain DIABETES in the title and that have a clinical synopsis:
SELECT mim_id, SUBSTR(title,1,75), has_synopsis FROM OMIM
    WHERE OMIM.CONTAINS(title, 'DIABETES') = 1
    AND OMIM.CONTAINS(has_synopsis, 'Clinical Synopsis') = 1;

The following query uses a CONTAINS function to look up all of the OMIM entries that were created on 5 May 1999:
SELECT mim_id, SUBSTR(title,1,40), creation_date FROM OMIM
    WHERE OMIM.CONTAINS(creation_date, '5/27/1999') = 1;

The following query uses a CONTAINS function to look up all of the OMIM entries that were created in 2005 and contain HEMERALOPIA-MYOPIA in the title:
SELECT mim_id, SUBSTR(title,1,40), creation_date FROM OMIM
    WHERE OMIM.CONTAINS(creation_date, '2005') = 1
    AND OMIM.CONTAINS(title, 'HEMERALOPIA-MYOPIA') = 1;

The following query uses a SEARCH_TERM function to look up all of the OMIM entries that contain HEMERALOPIA-MYOPIA in one of their fields:
SELECT mim_id, SUBSTR(title,1,40) FROM OMIM
    WHERE OMIM.SEARCH_TERM(mim_id, 'HEMERALOPIA-MYOPIA') = 1;

The following query uses a CONTAINS function to look up all of the OMIM entries that contain "CYSTIC FIBROSIS" in the title of the entry. This example shows how to SELECT values from nicknames in the OMIM hierarchy by joining the child nickname primary key and foreign key with the parent nickname:
SELECT o.mim_id, SUBSTR(o.title,1,20), SUBSTR(ot.text,1,40)
FROM omim o, omim_text ot
    WHERE OMIM.CONTAINS(o.title, 'CYSTIC FIBROSIS') = 1
    AND o.mim_id = ot.mim_id;

Queries for PubMed nicknames

The following query contains a single fetch key on a PubMed nickname:
SELECT PMID, ArticleTitle FROM pmarticles
    WHERE pmid = '12345';

The following query contains mixed fetch keys on a PubMed nickname:
SELECT PMID, ArticleTitle FROM pmarticles
    WHERE pmid = '12345' OR MedlineID = '12346';

The following query uses a CONTAINS function on a PubMed nickname:
SELECT PMID, ArticleTitle FROM pmarticles
    WHERE ENTREZ.CONTAINS (ArticleTitle, 'granulation') = 1
    AND ENTREZ.CONTAINS (PubDate, '1992') = 1;

The following query searches for the specified AuthorList and LanguageList on a PubMed nickname:
SELECT PMID, ArticleTitle FROM pmarticles
    WHERE ENTREZ.CONTAINS (AuthorList, 'Albarrak') = 1
    AND ENTREZ.CONTAINS (LanguageList, 'eng') = 1;

The following query uses a complex predicate on a PubMed nickname:
SELECT PMID, ArticleTitle FROM pmarticles
WHERE ENTREZ.CONTAINS (PublicationTypeList, 'Journal Article') = 1
AND ENTREZ.CONTAINS (MedlineTA, 'sun')=1
OR ENTREZ.CONTAINS (PersonalNameSubjectList, 'shine')=1;

Fixed columns for the Nucleotide nicknames
When you register the Nucleotide nicknames, information about the fields in the Nucleotide database is added to the federated database catalog. The information that is added to the catalog is represented as a set of fixed columns for each nickname.

List of the Nucleotide nicknames

The Nucleotide nicknames are:
• GBSeq
• GBReference
• GBFeatures
• GBIntervals
• GBQualifiers

The following tables provide information about the columns in each nickname:
• The name of the column.
• The data type for the column. You can override the default data type for a column when you create a nickname. For example, the default data type for the Sequence column in the GBSeq nickname is VARCHAR(32000). You can change this data type to CLOB(1 MB). The Entrez wrapper supports the CLOB data type, up to 5 megabytes in length.
• The description for the column.
• The search tag qualifier that is used by the wrapper to query the Nucleotide database. The only column names that you can specify in the ENTREZ.CONTAINS functions are the column names that have a search tag qualifier. The Qualifier column contains the valid search qualifiers for the column. For a list of valid search tags, see the following NCBI web site and locate the link to Search Field Descriptions and Qualifiers:
• If the column can return a null value. Most of the columns that return a null value are input columns.

GBSeq nickname fixed columns

The columns in the GBSeq nickname are described in the following table. The F column indicates columns that are designated fetch keys. Using the fetch keys might expedite query processing.

Table 29. Nucleotide GBSeq nickname

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
<th>Qualifier</th>
<th>Fetch key</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrimaryAccession</td>
<td>VARCHAR(16)</td>
<td>Primary accession number</td>
<td>PACC</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>NOT NULL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SequenceKey</td>
<td>VARCHAR(32)</td>
<td>The primary key column used to join the GBSeq nickname with child nicknames.</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOT NULL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
<td>Qualifier</td>
<td>Fetch key</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------</td>
<td>--------------------------------------------------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>LocusName</td>
<td>VARCHAR(16) NOT NULL</td>
<td>The name of the locus.</td>
<td>ACCN</td>
<td>No</td>
</tr>
<tr>
<td>SeqLength</td>
<td>INTEGER NOT NULL</td>
<td>The length of the sequence.</td>
<td>SLEN</td>
<td>No</td>
</tr>
<tr>
<td>Strandedness</td>
<td>VARCHAR(32)</td>
<td>not-set, single-stranded, double-stranded, mixed-stranded</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>MoleculeType</td>
<td>VARCHAR(16)</td>
<td>nucleic-acid, dna, rna, trna, rna, mrna, urna, snrna, snorna, peptide</td>
<td>PROP</td>
<td>No</td>
</tr>
<tr>
<td>Topology</td>
<td>VARCHAR(16)</td>
<td>linear, circular</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Division</td>
<td>CHAR(3) NOT NULL</td>
<td>The GenBank division.</td>
<td>PROP</td>
<td>No</td>
</tr>
<tr>
<td>UpdateDate</td>
<td>DATE NOT NULL</td>
<td>The date of the most recent update.</td>
<td>MDAT</td>
<td>No</td>
</tr>
<tr>
<td>CreateDate</td>
<td>DATE NOT NULL</td>
<td>The date that the record was created.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Definition</td>
<td>VARCHAR(7000) NOT NULL</td>
<td>The definition line of the sequence.</td>
<td>TITL</td>
<td>No</td>
</tr>
<tr>
<td>Version</td>
<td>INTEGER</td>
<td>The version ID of the sequence.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>GI</td>
<td>VARCHAR(16)</td>
<td>The GenInfo (GI) sequence ID.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>KeywordList</td>
<td>VARCHAR(7000)</td>
<td>A semicolon separated list</td>
<td>KYWD</td>
<td>No</td>
</tr>
<tr>
<td>Segment</td>
<td>VARCHAR(250)</td>
<td>The segment.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Source</td>
<td>VARCHAR(200) NOT NULL</td>
<td>The source.</td>
<td>ORGN</td>
<td>No</td>
</tr>
<tr>
<td>Organism</td>
<td>VARCHAR(7000) NOT NULL</td>
<td>The organism.</td>
<td>ORGN</td>
<td>No</td>
</tr>
<tr>
<td>Taxonomy</td>
<td>VARCHAR(7000) NOT NULL</td>
<td>The taxonomy.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Comment</td>
<td>VARCHAR(7000)</td>
<td>Comments</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Primary</td>
<td>VARCHAR(7000)</td>
<td>The primary.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>SourceDB</td>
<td>VARCHAR(250)</td>
<td>The source database.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Sequence</td>
<td>VARCHAR(32000)</td>
<td>The sequence.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>AllText</td>
<td>VARCHAR(1)</td>
<td>A pseudo-column that always returns NULL</td>
<td>ALL</td>
<td>No</td>
</tr>
<tr>
<td>FreeText</td>
<td>VARCHAR(1)</td>
<td>A pseudo-column that always returns NULL</td>
<td>WORD</td>
<td>No</td>
</tr>
</tbody>
</table>
**GBReference nickname fixed columns**

The columns in the GBReference nickname are described in the following table.

*Table 30. Nucleotide GBReference nickname*

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
<th>Qualifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>SequenceKey</td>
<td>VARCHAR(32) NOT NULL</td>
<td>The key that is used to join a GBReference child nickname with its parent nickname.</td>
<td>None</td>
</tr>
<tr>
<td>ReferenceNum</td>
<td>INTEGER NOT NULL</td>
<td>Parsed from GBReference_reference</td>
<td>None</td>
</tr>
<tr>
<td>RangeLow</td>
<td>INTEGER NOT NULL</td>
<td>Low base for reference (parsed from GBReference_reference)</td>
<td>None</td>
</tr>
<tr>
<td>RangeHigh</td>
<td>INTEGER NOT NULL</td>
<td>High base for reference (parsed from GBReference_reference)</td>
<td>None</td>
</tr>
<tr>
<td>Authors</td>
<td>VARCHAR(3200)</td>
<td>A semicolon-separated list of names in GenBank form</td>
<td>AUTH</td>
</tr>
<tr>
<td>Consortium</td>
<td>VARCHAR(250)</td>
<td>The consortium.</td>
<td>None</td>
</tr>
<tr>
<td>Title</td>
<td>VARCHAR(250)</td>
<td>The GenBank reference title.</td>
<td>None</td>
</tr>
<tr>
<td>Journal_Title</td>
<td>VARCHAR(250) NOT NULL</td>
<td>The title of the journal.</td>
<td>JOUR</td>
</tr>
<tr>
<td>MedlineID</td>
<td>INTEGER</td>
<td>The Medline ID</td>
<td>None</td>
</tr>
<tr>
<td>PubMedID</td>
<td>INTEGER</td>
<td>The PubMed ID</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>VARCHAR(3200)</td>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>

**GBFeatures nickname fixed columns**

The columns in the GBFeatures nickname are described in the following table.

*Table 31. Nucleotide GBFeatures nickname*

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
<th>Qualifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>SequenceKey</td>
<td>VARCHAR(32) NOT NULL</td>
<td>The key that is used to join a GBFeatures child nickname with its parent nickname.</td>
<td>None</td>
</tr>
<tr>
<td>FeatureJoinKey</td>
<td>VARCHAR(32) NOT NULL</td>
<td>The primary key column used to join the GBFeatures nickname with child nicknames.</td>
<td>None</td>
</tr>
<tr>
<td>FeatureKey</td>
<td>VARCHAR(20) NOT NULL</td>
<td>FKEY</td>
<td></td>
</tr>
<tr>
<td>FeatureLocation</td>
<td>VARCHAR(200) NOT NULL</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
**GBIntervals nickname fixed columns**

The columns in the GBIntervals nickname are described in the following table.

*Table 32. Nucleotide GBIntervals nickname*

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data type</th>
<th>Description</th>
<th>Qualifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>FeatureJoinKey</td>
<td>VARCHAR(32) NOT NULL</td>
<td>The key that is used to join a GBIntervals child nickname with its parent nickname.</td>
<td>None</td>
</tr>
<tr>
<td>IntervalFrom</td>
<td>INTEGER</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>IntervalTo</td>
<td>INTEGER</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>IntervalPoint</td>
<td>INTEGER</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>IntervalAccession</td>
<td>VARCHAR(32) NOT NULL</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**GBQualifiers nickname fixed columns**

The columns in the GBQualifiers nickname are described in the following table.

*Table 33. Nucleotide GBQualifiers nickname*

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
<th>Qualifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>FeatureJoinKey</td>
<td>VARCHAR(32) NOT NULL</td>
<td>The key that is used to join a GBQualifiers child nickname with its parent nickname.</td>
<td>None</td>
</tr>
<tr>
<td>QualifierName</td>
<td>VARCHAR(50)</td>
<td>The name of the qualifier</td>
<td>None</td>
</tr>
<tr>
<td>QualifierValue</td>
<td>VARCHAR(32000)</td>
<td>The value of the qualifier</td>
<td>None</td>
</tr>
</tbody>
</table>

**Fixed columns for the OMIM nicknames**

When you register the OMIM nicknames, information about the fields in the OMIM database is added to the federated database catalog. The information that is added to the catalog is represented as a set of fixed columns for each nickname.

You can retrieve data from any of the fixed columns by including the column name in your SELECT statement. However, the columns that have search tags are the only columns that can be specified in the WHERE clause of a query.

**List of the OMIM nicknames**

The OMIM nicknames are:

- OMIM
- OMIM_Alias
- OMIM_Text
- OMIM_Text_uid
- OMIM_See_Also
- OMIM_Allelic_Variant
- OMIM_Allelic_Variant_Description
- OMIM_Allelic_Variant_Description_uid
- OMIM_Clinical_Synopsis
- OMIM_Edit_History
• OMIM_References
• OMIM_Contributors
• OMIM_Links
• OMIM_Links_uid

The following tables provide information about the columns in each nickname:
• The name of the column.
• The data type for the column.
• Whether the column can return a null value. Most of the columns that return a null value are input columns.
• The search tag that is used by the wrapper to query the OMIM database. The column names that have a search tag are the only column names that you can specify in the OMIM.CONTAINS functions.
• The description for the column.

**OMIM nickname fixed columns**

The columns for the OMIM nickname are described in the following table.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIM_ID</td>
<td>VARCHAR(6)</td>
<td>N</td>
<td>ID</td>
<td>A six-digit number that uniquely identifies an OMIM database entry. A primary key column that is defined as a FOR BIT DATA column. The OMIM database entry numbering system is described at the following Web site: <a href="http://www.ncbi.nlm.nih.gov/omim/faq">OMIM Frequently Asked Questions (FAQs)</a></td>
</tr>
<tr>
<td>All.Fields</td>
<td>VARCHAR(1)</td>
<td>Y</td>
<td>ALL</td>
<td>An input column that is used to query all of the searchable columns in the OMIM database.</td>
</tr>
<tr>
<td>Chromosome</td>
<td>VARCHAR(1)</td>
<td>Y</td>
<td>CH</td>
<td>An input column that is used to specify the chromosome to which a gene or disorder is mapped in the OMIM Gene Map or Morbid Map.</td>
</tr>
<tr>
<td>Copyright</td>
<td>VARCHAR(60)</td>
<td>Y</td>
<td></td>
<td>The copyright statement for an OMIM database entry.</td>
</tr>
<tr>
<td>Created_By</td>
<td>VARCHAR(50)</td>
<td>Y</td>
<td></td>
<td>The full name of the person who created the OMIM database entry.</td>
</tr>
<tr>
<td>EC_RN_Number</td>
<td>VARCHAR(1)</td>
<td>Y</td>
<td>EC</td>
<td>An input column that is used to specify the number that designates a particular enzyme or chemical. The numbers for enzymes are assigned by the Enzyme Commission. The numbers for chemicals are assigned by the Chemical Abstract Service (CAS).</td>
</tr>
<tr>
<td>Filter</td>
<td>VARCHAR(1)</td>
<td>Y</td>
<td>FI</td>
<td>An input column that is used to retrieve records that contain crosslinks to other Entrez databases and LinkOuts to external (non-Entrez) resources. For example, &quot;omim protein&quot;, &quot;omim structure&quot;, &quot;lopovlocuslink&quot;.</td>
</tr>
<tr>
<td>Creation_Date</td>
<td>DATE</td>
<td>Y</td>
<td>CD</td>
<td>The date on which an OMIM database entry was created. The format must be in a format that is supported by the federated database, such as MM/DD/YYYY, DD.MM:YYYY, or YYYY-MM-DD.</td>
</tr>
<tr>
<td>Gene_Map</td>
<td>VARCHAR(1)</td>
<td>Y</td>
<td>GM</td>
<td>An input column that is used to search the cytogenetic map location that is represented in the OMIM Gene Map.</td>
</tr>
<tr>
<td>Gene_Map_Disorder</td>
<td>VARCHAR(1)</td>
<td>Y</td>
<td>DI</td>
<td>An input column that is used to search for text that appears in the Disorder column of the OMIM Gene Map.</td>
</tr>
</tbody>
</table>
Table 34. Columns for the OMIM nickname (continued)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gene_Map_Name</td>
<td>VARCHAR(1)</td>
<td>Y</td>
<td>GN</td>
<td>An input column that is used to search for the official gene symbol and alternate gene symbols that are associated with an OMIM database entry.</td>
</tr>
<tr>
<td>Has_Gm_Locus</td>
<td>VARCHAR(1)</td>
<td>Y</td>
<td>PR</td>
<td>An input column that is used to search for the OMIM database entries that have a gene map locus. For example: OMIM.CONTAINS (has_gm_locus,'has locus')</td>
</tr>
<tr>
<td>Has_Summary</td>
<td>CHAR(1)</td>
<td>Y</td>
<td></td>
<td>Specifies if an OMIM database entry has a summary. Valid values for this column are Y and N.</td>
</tr>
<tr>
<td>Has_Synopsis</td>
<td>CHAR(1)</td>
<td>Y</td>
<td>PR</td>
<td>A column that is used to search for the OMIM database entries that have a synopsis. For example: OMIM.CONTAINS (has_synopsis,'clinical synopsis')</td>
</tr>
<tr>
<td>Has_Variants</td>
<td>VARCHAR(1)</td>
<td>Y</td>
<td>PR</td>
<td>An input column that is used to search for OMIM database entries that have allelic variants. For example: OMIM.CONTAINS (has_variants,'allelic variants')</td>
</tr>
<tr>
<td>Locus</td>
<td>VARCHAR(10)</td>
<td>Y</td>
<td></td>
<td>The position that a given gene occupies on a chromosome.</td>
</tr>
<tr>
<td>MIM_ID_Display</td>
<td>VARCHAR(7)</td>
<td>N</td>
<td></td>
<td>Displays the six-digit MIM_ID with the symbol corresponding to the MIM_type_value. For example: '605491, %608583. See the following Web site for more information about the symbols that precede the MIM ID numbers, [OMIM Frequently Asked Questions (FAQs)](OMIM_Frequently Asked Questions (FAQs))</td>
</tr>
<tr>
<td>Number_Gene_Maps</td>
<td>INTEGER</td>
<td>Y</td>
<td></td>
<td>The number of gene maps.</td>
</tr>
<tr>
<td>Symbol</td>
<td>VARCHAR(10)</td>
<td>Y</td>
<td></td>
<td>The symbol for the OMIM database entry.</td>
</tr>
<tr>
<td>Text</td>
<td>VARCHAR(1000)</td>
<td>Y</td>
<td></td>
<td>A field in an OMIM database entry that contains information about the entry.</td>
</tr>
<tr>
<td>Title</td>
<td>VARCHAR(250)</td>
<td>Y</td>
<td>TI</td>
<td>The title for the OMIM database entry.</td>
</tr>
</tbody>
</table>

**OMIM_Alias nickname fixed columns**

The columns for the OMIM_Alias nickname are described in the following table.

Table 35. Columns for the OMIM_Alias nickname

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIM_ID</td>
<td>VARCHAR(6)</td>
<td>N</td>
<td></td>
<td>A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.</td>
</tr>
<tr>
<td>Alias</td>
<td>VARCHAR(100)</td>
<td>Y</td>
<td></td>
<td>Alternative titles and symbols for the OMIM database entry that are separated by semicolons.</td>
</tr>
</tbody>
</table>

**OMIM_Text nickname fixed columns**

The columns for the OMIM_Text nickname are described in the following table.
### Table 36. Columns for the OMIM_Text nickname

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIM_ID</td>
<td>VARCHAR(6)</td>
<td>N</td>
<td>ID</td>
<td>A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.</td>
</tr>
<tr>
<td>Neighbor_Link_uids</td>
<td>VARCHAR</td>
<td>Y</td>
<td></td>
<td>A list of unique IDs that are separated by commas.</td>
</tr>
<tr>
<td>Text</td>
<td>VARCHAR(3000)</td>
<td>N</td>
<td>TXT</td>
<td>The content of the text field.</td>
</tr>
<tr>
<td>Text_Join_Key</td>
<td>VARCHAR(32)</td>
<td>N</td>
<td></td>
<td>The primary key column for the OMIM_Text nickname. This column contains a unique ID that is generated by the wrapper. This column is defined as a FOR BIT DATA column.</td>
</tr>
<tr>
<td>Text_Label</td>
<td>VARCHAR(50)</td>
<td>N</td>
<td></td>
<td>The label for the text field. For example: Description, Clinical Features, or Cloning.</td>
</tr>
</tbody>
</table>

### OMIM_Text_uid nickname fixed columns

The columns for the OMIM_Text_uid nickname are described in the following table.

### Table 37. Columns for the OMIM_Text_uid nickname

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text_Join_Key</td>
<td>VARCHAR(32)</td>
<td>N</td>
<td></td>
<td>A foreign key that is used in a join condition with the parent nickname, OMIM_Text. This column is defined as a FOR BIT DATA column.</td>
</tr>
<tr>
<td>Uid</td>
<td>VARCHAR(10)</td>
<td>N</td>
<td></td>
<td>The unique ID that is associated with the Allelic_Join_Key for the OMIM_Text_uid nickname.</td>
</tr>
</tbody>
</table>

### OMIM_See_Also nickname fixed columns

The columns for the OMIM_See_Also nickname are described in the following table.

### Table 38. Columns for the OMIM_See_Also nickname

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIM_ID</td>
<td>VARCHAR(6)</td>
<td>N</td>
<td></td>
<td>A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.</td>
</tr>
<tr>
<td>Author</td>
<td>VARCHAR(50)</td>
<td>N</td>
<td></td>
<td>The last name of the primary author of the OMIM database entry.</td>
</tr>
<tr>
<td>Citation_Number</td>
<td>INTEGER</td>
<td>N</td>
<td></td>
<td>The citation number in the reference section of the OMIM database entry.</td>
</tr>
<tr>
<td>Others</td>
<td>VARCHAR(50)</td>
<td>Y</td>
<td></td>
<td>The last name of the second author if the OMIM database entry is authored by two authors. If there are more than two authors, the phrase et al is returned from the OMIM database entry. If there is only one author for the OMIM database entry, this column returns a null value.</td>
</tr>
</tbody>
</table>
### Table 38. Columns for the OMIM_See_Also nickname (continued)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>CHAR (4)</td>
<td>Y</td>
<td></td>
<td>The year that the OMIM database entry is published. The format for this value is YYYY.</td>
</tr>
</tbody>
</table>

#### OMIM_Allelic_Variant nickname fixed columns

The columns for the OMIM_Allelic_Variant nickname are described in the following table.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIM_ID</td>
<td>VARCHAR(6)</td>
<td>N</td>
<td>ID</td>
<td>A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.</td>
</tr>
<tr>
<td>Allelic_Join_Key</td>
<td>VARCHAR (32)</td>
<td>N</td>
<td></td>
<td>The primary key column for the OMIM_Allelic_Variant nickname. This column contains a unique ID that is generated by the wrapper. This column is defined as a FOR BIT DATA column.</td>
</tr>
<tr>
<td>Aliases</td>
<td>VARCHAR (100)</td>
<td>Y</td>
<td></td>
<td>A list of Allelic Variant aliases. The entries in the list are separated by commas.</td>
</tr>
<tr>
<td>Mutation</td>
<td>VARCHAR (40)</td>
<td>N</td>
<td>AV</td>
<td>The mutation of the Allelic Variant.</td>
</tr>
<tr>
<td>Name</td>
<td>VARCHAR (100)</td>
<td>N</td>
<td>AV</td>
<td>The name of the Allelic Variant.</td>
</tr>
<tr>
<td>Number</td>
<td>CHAR (4)</td>
<td>N</td>
<td></td>
<td>The number of the Allelic Variant. The Allelic Variant numbers are four digit numbers that start with 0001.</td>
</tr>
</tbody>
</table>

#### OMIM_Allelic_Variant_Description nickname fixed columns

The columns for the OMIM_Allelic_Variant_Description nickname are described in the following table.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allelic_Join_Key</td>
<td>VARCHAR (32)</td>
<td>N</td>
<td></td>
<td>A foreign key that is used in a join condition with the parent nickname, OMIM_Allelic_Variant. This column is defined as a FOR BIT DATA column.</td>
</tr>
<tr>
<td>Allelic_Description_Join_Key</td>
<td>VARCHAR (32)</td>
<td>N</td>
<td></td>
<td>The primary key column for the OMIM_Allelic_Variant_Description nickname. This column contains a unique ID that is generated by the wrapper. This column is defined as a FOR BIT DATA column.</td>
</tr>
<tr>
<td>Description</td>
<td>VARCHAR (3000)</td>
<td>N</td>
<td>AV</td>
<td>The description of the OMIM Allelic Variant.</td>
</tr>
<tr>
<td>Neighbor_Link_uids</td>
<td>VARCHAR (300)</td>
<td>Y</td>
<td></td>
<td>A list of unique IDs that are separated by commas.</td>
</tr>
</tbody>
</table>
OMIM_Allelic_Variant_Description_uid nickname fixed columns

The columns for the OMIM_Allelic_Variant_Description_uid nickname are described in the following table.

Table 41. Columns for the OMIM_Allelic_Variant_Description_uid nickname

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allelic_Description_Join_Key</td>
<td>VARCHAR(32)</td>
<td>N</td>
<td></td>
<td>A foreign key that is used in a join condition with the parent nickname, OMIM_Allelic_Variant_Description. This column is defined as a FOR BIT DATA column.</td>
</tr>
<tr>
<td>Uid</td>
<td>VARCHAR(10)</td>
<td>N</td>
<td></td>
<td>The unique ID that is associated with the Allelic_Join_Key for the OMIM_Allelic_Variant_Description_uid nickname.</td>
</tr>
</tbody>
</table>

OMIM_Clinical_Synopsis nickname fixed columns

The columns for the OMIM_Clinical_Synopsis nickname are described in the following table.

Table 42. Columns for the OMIM_Clinical_Synopsis nickname

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIM_ID</td>
<td>VARCHAR(6)</td>
<td>N</td>
<td></td>
<td>A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.</td>
</tr>
<tr>
<td>Author</td>
<td>VARCHAR (40)</td>
<td>Y</td>
<td></td>
<td>The full name of the author of the synopsis.</td>
</tr>
<tr>
<td>Creation_Date</td>
<td>DATE</td>
<td>Y</td>
<td></td>
<td>The date that the synopsis was created.</td>
</tr>
<tr>
<td>Synopsis_Key</td>
<td>VARCHAR(100)</td>
<td>Y</td>
<td></td>
<td>The clinical synopsis key. For example: Inheritance, Cardiovascular, Respiratory, Growth.</td>
</tr>
<tr>
<td>Synopsis_Terms</td>
<td>VARCHAR(500)</td>
<td>Y</td>
<td>CS</td>
<td>A list of terms that are used in the clinical synopsis. The list is separated by semicolons.</td>
</tr>
</tbody>
</table>

OMIM_Edit_History nickname fixed columns

The columns for the OMIM_Edit_History nickname are described in the following table.

Table 43. Columns for the OMIM_Edit_History nickname

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIM_ID</td>
<td>VARCHAR(6)</td>
<td>N</td>
<td></td>
<td>A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.</td>
</tr>
<tr>
<td>Editor_ID</td>
<td>VARCHAR (20)</td>
<td>N</td>
<td>ED</td>
<td>The ID for the editor of the OMIM database entry.</td>
</tr>
<tr>
<td>Date</td>
<td>DATE</td>
<td>N</td>
<td>MD</td>
<td>The date that the OMIM database entry was edited. You must specify the date in a format that is supported by the federated database, such as MM/DD/YYYY, DD.MM.YYYY, or YYYY-MM-DD.</td>
</tr>
</tbody>
</table>
### OMIM_References nickname fixed columns

The columns for the OMIM_References nickname are described in the following table.

*Table 44. Columns for the OMIM_References nickname*

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIM_ID</td>
<td>VARCHAR(6)</td>
<td>N</td>
<td></td>
<td>A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.</td>
</tr>
<tr>
<td>Authors_List</td>
<td>VARCHAR (1000)</td>
<td>Y</td>
<td>RE</td>
<td>The list of authors for the reference. The names are separated by a semicolon and use the format Last Name, First/Middle initials. For example: Stewart, L.K.; Wamhoff, J.</td>
</tr>
<tr>
<td>Ambiguous</td>
<td>CHAR(1)</td>
<td>Y</td>
<td></td>
<td>Specifies whether the reference is ambiguous. Valid values for this column are Y and N.</td>
</tr>
<tr>
<td>Citation_Title</td>
<td>VARCHAR (100)</td>
<td>Y</td>
<td>RE</td>
<td>The titles of the references that are cited in the OMIM database entry.</td>
</tr>
<tr>
<td>Citation_Type</td>
<td>VARCHAR(10)</td>
<td>Y</td>
<td></td>
<td>The type of citation.</td>
</tr>
<tr>
<td>Journal</td>
<td>VARCHAR(50)</td>
<td>Y</td>
<td></td>
<td>Journal</td>
</tr>
<tr>
<td>No_Link</td>
<td>CHAR(1)</td>
<td>Y</td>
<td></td>
<td>Specifies if an OMIM database entry has links to other references. Valid values for this column are Y and N.</td>
</tr>
<tr>
<td>Other_Authors</td>
<td>VARCHAR (20)</td>
<td>Y</td>
<td>RE</td>
<td>The names of other authors that are cited in the reference. The last name of the second author if the reference is authored by two authors. If there are more than two authors, the phrase et al is returned from the OMIM database entry. If there is only one author for the reference, this column returns a null value.</td>
</tr>
<tr>
<td>Page_From</td>
<td>INTEGER</td>
<td>Y</td>
<td></td>
<td>The first page number in the reference where the article begins.</td>
</tr>
<tr>
<td>Page_To</td>
<td>INTEGER</td>
<td>Y</td>
<td></td>
<td>The last page number in the reference where the article ends. If the article is contained on one page, this column returns a null value.</td>
</tr>
<tr>
<td>Publication_Date</td>
<td>DATE</td>
<td>Y</td>
<td></td>
<td>The date that the reference was published. If the day or the month are not present, the day or month is set to 1. For example, if the entry in the OMIM database is April 2004, then the date 2004/04/01 is returned.</td>
</tr>
<tr>
<td>Pubmed_ID</td>
<td>VARCHAR (10)</td>
<td>Y</td>
<td></td>
<td>The Pubmed ID (PMID). If no PMID is available, this column returns '0'.</td>
</tr>
<tr>
<td>Primary.Author</td>
<td>VARCHAR (20)</td>
<td>Y</td>
<td>RE</td>
<td>The last name of the primary author for the reference that is cited.</td>
</tr>
<tr>
<td>Ref_Orig_Number</td>
<td>INTEGER</td>
<td>Y</td>
<td></td>
<td>The original reference number for the OMIM database entry.</td>
</tr>
<tr>
<td>Ref_Number</td>
<td>INTEGER</td>
<td>N</td>
<td></td>
<td>The current reference number for the OMIM database entry.</td>
</tr>
<tr>
<td>Type</td>
<td>VARCHAR(20)</td>
<td>Y</td>
<td></td>
<td>The type of reference. For example citation, book, personal communication.</td>
</tr>
<tr>
<td>Volume</td>
<td>VARCHAR (10)</td>
<td>Y</td>
<td></td>
<td>The volume for the reference that is cited.</td>
</tr>
</tbody>
</table>
**OMIM_Contributors nickname fixed columns**

The columns for the OMIM_Contributors nickname are described in the following table.

Table 45. Columns for the OMIM_Contributors nickname

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIM_ID</td>
<td>VARCHAR(6)</td>
<td>N</td>
<td></td>
<td>A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.</td>
</tr>
<tr>
<td>Action</td>
<td>VARCHAR (10)</td>
<td>Y</td>
<td></td>
<td>The action that was performed on a record. For example: updated and recognized. The action is parsed from the Author and Action fields.</td>
</tr>
<tr>
<td>Author</td>
<td>VARCHAR (40)</td>
<td>N</td>
<td>AU</td>
<td>The name of the contributor name. This name is parsed from the Author and Action fields.</td>
</tr>
<tr>
<td>Date</td>
<td>DATE</td>
<td>Y</td>
<td></td>
<td>The date that the contribution was made.</td>
</tr>
</tbody>
</table>

**OMIM_Links nickname fixed columns**

The columns for the OMIM_Links nickname are described in the following table.

Table 46. Columns for the OMIM_Links nickname

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIM_ID</td>
<td>VARCHAR(6)</td>
<td>N</td>
<td></td>
<td>A six-digit number that uniquely identifies an OMIM database entry. A foreign key column that is used in a join condition with the parent nickname, OMIM. The MIM_ID column is defined as a FOR BIT DATA column.</td>
</tr>
<tr>
<td>Links_Join_Key</td>
<td>VARCHAR (32)</td>
<td>N</td>
<td></td>
<td>The primary key column for the OMIM_Links nickname. This column contains a unique ID that is generated by the wrapper. This column is defined as a FOR BIT DATA column.</td>
</tr>
<tr>
<td>Link_Number</td>
<td>INTEGER</td>
<td>Y</td>
<td></td>
<td>The link number in the OMIM database entry.</td>
</tr>
<tr>
<td>Link_Type</td>
<td>VARCHAR (10)</td>
<td>Y</td>
<td></td>
<td>The type of link in the OMIM database entry. The valid types are Medline, Protein, Nucleotide, or Structure.</td>
</tr>
<tr>
<td>Link_Uids</td>
<td>VARCHAR (300)</td>
<td>Y</td>
<td></td>
<td>A list of unique IDs that are associated with the Links_Join_Key for the OMIM_Links nickname. The unique IDs are separated by commas.</td>
</tr>
</tbody>
</table>

**OMIM_Links_uid nickname fixed columns**

The columns for the OMIM_Links_uid nickname are described in the following table.

Table 47. Columns for the OMIM_Links_uid nickname

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Links_Join_Key</td>
<td>VARCHAR (32)</td>
<td>N</td>
<td></td>
<td>A foreign key that is used in a join condition with the parent nickname, OMIM_Links. This column is defined as a FOR BIT DATA column.</td>
</tr>
</tbody>
</table>
Table 47. Columns for the OMIM_Links_uid nickname (continued)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Null</th>
<th>Search tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uid</td>
<td>VARCHAR(10)</td>
<td>N</td>
<td></td>
<td>The unique ID that is associated with the Links_Join_Key for the OMIM_Links_uid nickname.</td>
</tr>
</tbody>
</table>

**Fixed columns for PubMed nicknames**

When you register the PubMed nicknames, information about the fields in the PubMed database is added to the federated database catalog. The information that is added to the catalog is represented as a set of fixed columns for each nickname.

**List of the PubMed nicknames**

The PubMed nicknames are:

- PMArticles
- PMAccession
- PMChemical
- PMMeSHHeading
- PMComments
- PMArticleID

The following tables provide information about the columns in each nickname:

- The name of the column.
- The data type for the column.
- The description for the column.
- The search tag that is used by the wrapper to query the PubMed database. The column names that have a search tag are the only column names that you can specify in the ENTREZ.CONTAINS functions.
- If the column can return a null value. Most of the columns that return a null value are input columns.

You can override the default length for a column when you create a nickname. For example, some columns can return a large amount of data, such as the Abstract column in the PMArticles nickname. The default length for this column is VARCHAR(32000). To return the first 100 bytes of the column, you can define the column with the data type VARCHAR(100). Only the first 100 bytes will be returned.

For a list of valid search tags, see the following NCBI web site and locate the link to Search Field Descriptions and Qualifiers: [www.ncbi.nlm.nih.gov/entrez/query/static/help/pmhelp.html](http://www.ncbi.nlm.nih.gov/entrez/query/static/help/pmhelp.html)

The Entrez wrapper supports the CLOB data type, up to 5 megabytes in length.

**PMArticles nickname fixed columns**

The columns in the PMArticles nickname are described in the following table. The F column indicates columns that are designated fetch keys. Using the fetch keys might expedite query processing.
### Table 48. PubMed PMArticles nickname

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
<th>Tags</th>
<th>Fetch key</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMID</td>
<td>VARCHAR(10)</td>
<td>PRIMARY KEY column used to join the PMArticles nickname with child nicknames</td>
<td>UID</td>
<td>Yes</td>
</tr>
<tr>
<td>MedlineID</td>
<td>VARCHAR(10)</td>
<td>Medline ID</td>
<td>UID</td>
<td>Yes</td>
</tr>
<tr>
<td>Owner</td>
<td>VARCHAR(8)</td>
<td>Owner of the publication entry; values are defined by NCBI and might be NLM, NASA, PIP, KIE, HSR, HMD, SIS, NOTNLM. If not present, then the default is NLM.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Status</td>
<td>VARCHAR(32)</td>
<td>Publication status as defined by NCBI. Values might include: In-Process, Completed, Out-of-scope, PubMed-not_MEDLINE.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>DateCreated</td>
<td>DATE</td>
<td>Date that the publication entry was created.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>DateCompleted</td>
<td>DATE</td>
<td>Date that the publication entry was completed.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>DateRevised</td>
<td>DATE</td>
<td>Date that the publication entry was revised.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>ArticleTitle</td>
<td>VARCHAR(250)</td>
<td>Title of the article.</td>
<td>TI</td>
<td>No</td>
</tr>
<tr>
<td>Pagination</td>
<td>VARCHAR(32)</td>
<td>Full pagination of the article.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Abstract</td>
<td>VARCHAR(32000)</td>
<td>Abstract for the article.</td>
<td>TIAB</td>
<td>No</td>
</tr>
<tr>
<td>Affiliation</td>
<td>VARCHAR(250)</td>
<td>Affiliation and address of the first author</td>
<td>AD</td>
<td>No</td>
</tr>
<tr>
<td>AuthorList</td>
<td>VARCHAR(3200)</td>
<td>List of authors; canonized</td>
<td>AU</td>
<td>No</td>
</tr>
<tr>
<td>LanguageList</td>
<td>VARCHAR(250)</td>
<td>Semicolon-separated list</td>
<td>LA</td>
<td>No</td>
</tr>
<tr>
<td>PublicationnTypeList</td>
<td>VARCHAR(250)</td>
<td>Semicolon-separated list</td>
<td>PT</td>
<td>No</td>
</tr>
<tr>
<td>VernacularTitle</td>
<td>VARCHAR(250)</td>
<td>Vernacular title for the article.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>DateOfElectronicPublication</td>
<td>VARCHAR(32)</td>
<td>NCBI schema specifies no structure for this column</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Country</td>
<td>VARCHAR(128)</td>
<td>Country or region of publication that is cited in the journal.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>MedlineTA</td>
<td>VARCHAR(250)</td>
<td>Medline title abbreviation.</td>
<td>TA</td>
<td>No</td>
</tr>
<tr>
<td>NlmUniqueID</td>
<td>VARCHAR(32)</td>
<td>Contains MedlineCode if NlmUniqueID is not present</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>GeneSymbolList</td>
<td>VARCHAR(250)</td>
<td>A semicolon-separated list; not used since 1996</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>NumberOfReferences</td>
<td>INTEGER</td>
<td>Number of bibliographic references for the review article.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>PersonalNameSubjectList</td>
<td>VARCHAR(250)</td>
<td>Canonized as a semicolon-separated list of names</td>
<td>PS</td>
<td>No</td>
</tr>
<tr>
<td>KeywordList</td>
<td>VARCHAR(3200)</td>
<td>A semicolon-separated list</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>SpaceFlightMissionList</td>
<td>VARCHAR(250)</td>
<td>A semicolon-separated list</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>InvestigatorList</td>
<td>VARCHAR(250)</td>
<td>Canonized as a semicolon-separated list of names</td>
<td>None</td>
<td>No</td>
</tr>
</tbody>
</table>
### Table 48. PubMed PMArticles nickname (continued)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
<th>Tags</th>
<th>Fetch key</th>
</tr>
</thead>
<tbody>
<tr>
<td>PublicationStatus</td>
<td>VARCHAR(32)</td>
<td>Status of the publication.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>ProviderID</td>
<td>VARCHAR(32)</td>
<td>Publication provider ID.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>CitationSubset List</td>
<td>VARCHAR(250)</td>
<td>A semicolon-separated list</td>
<td>SB</td>
<td>No</td>
</tr>
<tr>
<td>AllFields</td>
<td>VARCHAR(1)</td>
<td>Pseudo-column; always returns NULL</td>
<td>ALL</td>
<td>No</td>
</tr>
<tr>
<td>TextWords</td>
<td>VARCHAR(1)</td>
<td>Pseudo-column; always returns NULL</td>
<td>TW</td>
<td>No</td>
</tr>
<tr>
<td>PubDate</td>
<td>DATE</td>
<td>Includes the journal and book publication date + medline date</td>
<td>DP</td>
<td>No</td>
</tr>
<tr>
<td>PubDateString</td>
<td>VARCHAR(32)</td>
<td>Includes the journal and book publication date + medline date</td>
<td>DP</td>
<td>No</td>
</tr>
<tr>
<td>Title</td>
<td>VARCHAR(250)</td>
<td>Book or journal title</td>
<td>TA</td>
<td>No</td>
</tr>
<tr>
<td>Journal_ISSN</td>
<td>CHAR(9)</td>
<td>ISSN for the journal.</td>
<td>TA</td>
<td>No</td>
</tr>
<tr>
<td>Journal_Volume</td>
<td>VARCHAR(10)</td>
<td>Volume of the journal.</td>
<td>VI</td>
<td>No</td>
</tr>
<tr>
<td>Journal_Issue</td>
<td>VARCHAR(10)</td>
<td>Issue of the journal.</td>
<td>IP</td>
<td>No</td>
</tr>
<tr>
<td>Journal_Coden</td>
<td>VARCHAR(32)</td>
<td>Code number (coden) of the journal.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Journal_ISO Abbreviation</td>
<td>VARCHAR(32)</td>
<td>ISO abbreviation for the journal.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Book_Publisher</td>
<td>VARCHAR(128)</td>
<td>Publisher the book.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Book_Authors</td>
<td>VARCHAR(250)</td>
<td>Canonized as other author lists</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Book_Collection Title</td>
<td>VARCHAR(128)</td>
<td>Collection title of the book.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Book_Volume</td>
<td>VARCHAR(10)</td>
<td>Volume of the book.</td>
<td>None</td>
<td>No</td>
</tr>
</tbody>
</table>

#### PMAccession nickname fixed columns

The columns in the PMAccession nickname are described in the following table.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMID</td>
<td>VARCHAR(10)</td>
<td>Key that is used to join a PMAccession child nickname with its parent nickname.</td>
<td>None</td>
</tr>
<tr>
<td>DataBankName</td>
<td>VARCHAR(250)</td>
<td>Name of the data bank.</td>
<td>SI</td>
</tr>
<tr>
<td>Accession</td>
<td>VARCHAR(32)</td>
<td>Accession number.</td>
<td>SI</td>
</tr>
</tbody>
</table>

#### PMChemical nickname fixed columns

The columns in the PMChemical nickname are described in the following table.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMID</td>
<td>VARCHAR(10)</td>
<td>Key that is used to join a PMChemical child nickname with its parent nickname.</td>
<td>None</td>
</tr>
<tr>
<td>NameOfSubstance</td>
<td>VARCHAR(128)</td>
<td>Name of the substance.</td>
<td>NM</td>
</tr>
</tbody>
</table>
Table 50. PubMed PMChemical nickname (continued)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegistryNumber</td>
<td>VARCHAR(32) NOT</td>
<td>Might be a CAS or other registry number</td>
<td>RN</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CASRegistry</td>
<td>CHAR</td>
<td>Y or N</td>
<td>None</td>
</tr>
</tbody>
</table>

**PMMeshHeading nickname fixed columns**

The columns in the PMMeSHHeading nickname are described in the following table.

Table 51. PubMed PMMeSHHeading nickname

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMID</td>
<td>VARCHAR(10) NOT</td>
<td>Key that is used to join a PMMeSHHeading child</td>
<td>ID</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td>nickname with its parent nickname.</td>
<td></td>
</tr>
<tr>
<td>DescriptorOrName</td>
<td>VARCHAR(128) NOT</td>
<td>Name or descriptor of the MeSH.</td>
<td>MH1</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DescriptorIsMajor</td>
<td>CHAR NOT NULL</td>
<td>Y if the descriptor is major</td>
<td>None</td>
</tr>
<tr>
<td>QualifierOrSubhead</td>
<td>VARCHAR(128)</td>
<td>Qualifier or subheading of the MeSH.</td>
<td>SH</td>
</tr>
<tr>
<td>QSIsMajor</td>
<td>CHAR</td>
<td>Y if qualifier or subhead is major</td>
<td>None</td>
</tr>
</tbody>
</table>

**Note:**
1. If the predicate "DescriptorIsMajor = Y" is included in the query, then the search term is MAJR.

**PMComments nickname fixed columns**

The columns in the PMComments nickname are described in the following table.

Table 52. PubMed PMComments nickname

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMID</td>
<td>VARCHAR(10) NOT</td>
<td>Key that is used to join a PMComments child nickname with its parent nickname.</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RefSource</td>
<td>VARCHAR(128) NOT</td>
<td>Source of the reference.</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>VARCHAR(32) NOT</td>
<td>Examples of valid types are: CommentOn, CommentIn, ErratumIn, ErratumFor, RepublishedFrom, RepublishedIn, RetractionOf, RetractionIn, UpdateIn, UpdateOf, SummaryForPatents, OriginalReportIn</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note</td>
<td>VARCHAR(3200)</td>
<td>Notes</td>
<td>None</td>
</tr>
</tbody>
</table>

**PMArticleID nickname fixed columns**

The columns in the PMArticleID nickname are described in the following table.

Table 53. PubMed PMArticleID nickname

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMID</td>
<td>VARCHAR(10) NOT</td>
<td>Key that is used to join a PMArticleID child nickname with its parent nickname.</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 53. PubMed PMArticleID nickname (continued)

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArticleID</td>
<td>VARCHAR(32) NOT NULL</td>
<td>ID of the article.</td>
<td>None</td>
</tr>
<tr>
<td>IdType</td>
<td>VARCHAR(8) NOT NULL</td>
<td>Examples of valid article identification types are: doi, pii, pmcpid, pmpid, sici, pubmed, medline, pmcid</td>
<td>None</td>
</tr>
</tbody>
</table>
Chapter 6. Configuring access to Excel data sources

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

Procedure

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

Excel wrapper

An Excel workbook is a file that is created using the Microsoft Excel application and has a file extension of xls. The Excel wrapper is used to perform searches on the Excel files.

You use Excel files to store information that is best displayed in a table, with corresponding rows and columns. Excel workbooks consist of one or more spreadsheet pages, or worksheets. Worksheets are often used to perform calculations.

The following figure shows how the Excel wrapper connects your worksheets to the federated system.

![Diagram showing how the Excel wrapper connects worksheets to the federated system]

Figure 7. How the Excel wrapper works

The Excel wrapper uses the CREATE NICKNAME statement to map the columns in your Excel worksheets to columns in your federated system. The following table shows a sample of worksheet data that is stored in a file called Compound_Master.xls.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>COMPOUND_NAME</td>
<td>WEIGHT</td>
<td>MOL_COUNT</td>
<td>WAS_TESTED</td>
</tr>
</tbody>
</table>
Table 54. Sample worksheet for Compound_Master.xls (continued)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>compound_A</td>
<td>1.23</td>
<td>367</td>
</tr>
<tr>
<td>3</td>
<td>compound_G</td>
<td></td>
<td>210</td>
</tr>
<tr>
<td>4</td>
<td>compound_F</td>
<td>0.000425536</td>
<td>174</td>
</tr>
<tr>
<td>5</td>
<td>compound_Y</td>
<td>1.00256</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>compound_Q</td>
<td></td>
<td>1024</td>
</tr>
<tr>
<td>7</td>
<td>compound_B</td>
<td>33.5362</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>compound_S</td>
<td>0.96723</td>
<td>67</td>
</tr>
<tr>
<td>9</td>
<td>compound_O</td>
<td>1.2</td>
<td></td>
</tr>
</tbody>
</table>

The information in an Excel worksheet is usually not available to you through standard SQL commands. When the Excel wrapper is installed and registered on your federated server, you can access this information as if it were a typical relational data source. For example, if you wanted to know all the compound data where the molecular count is greater than 100, you would run the following SQL query:

```
SELECT * FROM compound_master WHERE mol_count > 100
```

The results of the query are shown in the following table.

Table 55. Query results

<table>
<thead>
<tr>
<th>COMPOUND_NAME</th>
<th>WEIGHT</th>
<th>MOL_COUNT</th>
<th>WAS_TESTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>compound_A</td>
<td>1.23</td>
<td>367</td>
<td>tested</td>
</tr>
<tr>
<td>compound_G</td>
<td></td>
<td>210</td>
<td></td>
</tr>
<tr>
<td>compound_F</td>
<td>0.000425536</td>
<td>174</td>
<td>tested</td>
</tr>
<tr>
<td>compound_Q</td>
<td></td>
<td>1024</td>
<td></td>
</tr>
</tbody>
</table>

Methods of accessing Excel data

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

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

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

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

Each wrapper imposes some requirements on the location and layout of the data in the Excel workbooks. With the Excel wrapper, only the data in the first worksheet in the workbook can be accessed. With the ODBC wrapper, you can access data from any worksheet in the workbook.
The following examples show the worksheet layout requirements for these two wrappers.

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

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

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>Compound Analysis</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><strong>Compound Name</strong></td>
<td>Weight</td>
<td>Molecular Count</td>
<td>Tested?</td>
</tr>
<tr>
<td>4</td>
<td>compound_A</td>
<td>1.23</td>
<td>367</td>
<td>tested</td>
</tr>
<tr>
<td>5</td>
<td>compound_B</td>
<td>210</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>compound_F</td>
<td>0.000425536</td>
<td>174</td>
<td>tested</td>
</tr>
<tr>
<td>7</td>
<td>compound_Y</td>
<td>1.000256</td>
<td></td>
<td>tested</td>
</tr>
<tr>
<td>8</td>
<td>compound_Q</td>
<td>1024</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>compound_B</td>
<td>33.5382</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>compound_S</td>
<td>0.96723</td>
<td>67</td>
<td>tested</td>
</tr>
<tr>
<td>11</td>
<td>compound_O</td>
<td>1.2</td>
<td></td>
<td>tested</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td><strong>Total Compounds Tested</strong></td>
<td></td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 8. A worksheet that contains several rows of labels and a formula*

**If you use the Excel wrapper**

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

**If you use the ODBC wrapper**

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

Example of a worksheet that contains one row of labels

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

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Compound Name</td>
<td>Weight</td>
<td>Molecular Count</td>
<td>Tested?</td>
</tr>
<tr>
<td>2</td>
<td>compound_A</td>
<td>1.23</td>
<td>367</td>
<td>tested</td>
</tr>
<tr>
<td>3</td>
<td>compound_G</td>
<td></td>
<td>210</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>compound_F</td>
<td>0.000425536</td>
<td>174</td>
<td>tested</td>
</tr>
<tr>
<td>5</td>
<td>compound_Y</td>
<td>1.000256</td>
<td></td>
<td>tested</td>
</tr>
<tr>
<td>6</td>
<td>compound_Q</td>
<td></td>
<td>1024</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>compound_B</td>
<td>33.5362</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>compound_S</td>
<td>0.96723</td>
<td>67</td>
<td>tested</td>
</tr>
<tr>
<td>9</td>
<td>compound_Q</td>
<td>1.2</td>
<td></td>
<td>tested</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 9. A worksheet that contains one row of column labels in row 1

If you use the Excel wrapper

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

If you use the ODBC wrapper

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

Example of a worksheet that contains only data

This example shows a worksheet that contains only data. There are no rows of column labels, no blank rows, and no cells with formulas.
If you use the Excel wrapper

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

If you use the ODBC wrapper

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

Adding Excel data sources to a federated server

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

Before you begin

- The data in Excel worksheets must be structured correctly so that the Excel wrapper can access the data.
- Federation must be installed on a server that will act as the federated server.
- A database must exist on the federated server.

Restrictions

- The Excel wrapper is available only for versions of the Microsoft Windows operating system that are supported by the federated server.
The Excel application must be installed on the federated server.
The Excel workbook must be on the same computer as the federated server or on a network accessible drive.
The data in only the first worksheet in the Excel workbook can be accessed by the Excel wrapper.
The federated database codepage set must match the Excel file character set, otherwise you could get unexpected results from your queries.
Pass-through sessions are not allowed.

**About this task**

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

**Procedure**

To add the Excel data sources to a federated server:
1. "Registering the Excel wrapper."
2. "Registering the server definition for an Excel data source" on page 137.

**Registering the Excel wrapper**

You must register a wrapper to access Excel data sources.

**About this task**

Wrappers are used by federated servers to communicate with and retrieve data from the data sources. Wrappers are implemented as a set of library files.

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

**Procedure**

To register the Excel wrapper:

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the <strong>Federated Database Objects</strong> folder and click <strong>Create Federated Objects</strong>.</td>
</tr>
</tbody>
</table>
| Using the command line | Issue the CREATE WRAPPER statement. For example:  
CREATE WRAPPER excel_wrapper 
LIBRARY 'db2lsxls.dll';  
You must specify the LIBRARY parameter in the CREATE WRAPPER statement. |
Excel wrapper library files
The Excel wrapper library files are added to the federated server when you install the federated server.

When you install the federated server, three library files are added to the default directory path. For example, if the federated server is running on Windows, the wrapper library files that are added to the directory path are db2lsxls.dll, db2lsxlsF.dll, and db2lsxlsU.dll. The default wrapper library file is db2lsxls.dll. The other wrapper library files are used with specific wrapper options.

When you register the Excel wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

The default directory path and default wrapper library file name are listed in the following table.

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Wrapper library file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>%DB2PATH%\bin</td>
<td>db2lsxls.dll</td>
</tr>
</tbody>
</table>

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

Registering the server definition for an Excel data source
You must register a server definition because the hierarchy of the federated objects requires that the Excel workbook files, which are identified by nicknames, are associated with a specific server definition object.

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

Procedure
To register a server definition for an Excel data source:

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Using the command line</td>
<td>Issue the CREATE SERVER statement. For example: CREATE SERVER server_definition_name WRAPPER excel_wrapper;</td>
</tr>
</tbody>
</table>
CREATE SERVER statement - Examples for the Excel wrapper

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

The following example shows you how to register a server definition called biochem_lab for a workbook that contains biochemical data. The CREATE SERVER statement that you issue is:

```
CREATE SERVER biochem_lab WRAPPER excel_wrapper;
```

- `biochem_lab` - A name that you assign to the Excel server definition. Duplicate server definition names are not allowed.

- `WRAPPER Excel_wrapper` - The wrapper name that you specified in the CREATE WRAPPER statement.

Registering nicknames for Excel data sources

For each Excel server definition that you register, you must register a nickname for each Excel worksheet that you want to access. Use these nicknames, instead of the names of the worksheets, when you query the Excel data sources.

About this task

When you create a nickname for an Excel worksheet, the information in the worksheet data is mapped to a relational table.

Blank cells in the worksheet are interpreted as NULL.

Up to 10 consecutive blank rows can exist in the worksheet and will be included in the data set. More than 10 consecutive blank rows are interpreted as the end of the data set.

Blank columns can exist in the worksheet. However, these columns must be registered and described as valid fields even if they are not used.

Procedure

To register a nickname for an Excel worksheet:

Choose the method that you want to use to register the nickname. Nicknames can be up to 128 characters in length.

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects. Follow the steps in the wizard.</td>
</tr>
<tr>
<td>Method</td>
<td>Procedure</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>From command line</td>
<td>Issue the CREATE NICKNAME statement. For example:</td>
</tr>
<tr>
<td></td>
<td>CREATE NICKNAME nickname</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>column_name data_type</td>
</tr>
<tr>
<td></td>
<td>OPTIONS (nickname_column_options),</td>
</tr>
<tr>
<td></td>
<td>column_name data_type</td>
</tr>
<tr>
<td></td>
<td>OPTIONS (nickname_column_options),</td>
</tr>
<tr>
<td></td>
<td>column_name data_type</td>
</tr>
<tr>
<td></td>
<td>OPTIONS (nickname_column_options)</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>FOR SERVER server_definition_name OPTIONS (nickname_options);</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Repeat this step for each Excel worksheet that you want to create a nickname for.</td>
</tr>
</tbody>
</table>

**CREATE NICKNAME statement - examples for the Excel wrapper**

Use the CREATE NICKNAME statement to register a nickname for an Excel worksheet that you want to access. These examples show the required parameters and optional nickname options.

```
CREATE NICKNAME Compounds
{
  Compound_ID INTEGER,
  CompoundName VARCHAR(50),
  MolWeight FLOAT
}
FOR SERVER biochem_lab
OPTIONS (FILE_PATH 'C:\My Documents\CompoundMaster.xls',
          RANGE 'B2:D25');
```

*Compounds*

A unique nickname that is used to identify the Excel worksheet.

**Important:** The nickname is a two-part name the consists of the schema and the name of the nickname. If you omit the schema when you register the nickname, the authorization ID of the user who registers the nickname is used for the nickname schema.

*Compound_ID INTEGER*

The name and data type for a worksheet column that contains the compound identifiers.

*CompoundName VARCHAR(50)*

The name and data type for a worksheet column that contains the compound names.

*MolWeight FLOAT*

The name and data type for a worksheet column that contains the molecular weight of the compounds.

*FOR SERVER biochem_lab*

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

*FILE_PATH 'C:\My Documents\CompoundMaster.xls'*

Specifies the fully qualified directory path and file name for the Excel workbook that contains the data you want to access. The data must be in the first worksheet in the workbook.
OPTIONS (RANGE 'B2:D25')
Specifies the range of cells that you want to access in the workbook that you specified in the FILE_PATH nickname option.

Any syntax or semantic error in the range option value results in an SQL1882E message. Errors might include:

- The range is not a valid range. For example, if the top-left cell specified in the range is either below or to the right of the bottom-right cell.
- The number of columns designated by the range value does not correspond to the number of columns specified in the CREATE NICKNAME statement.
- A nonvalid character or other syntax error has been found.

Excel data sources - example queries
To access Excel data, you use the nickname and the defined nickname columns in your SQL statements in the same manner as you would use a regular table name and table columns.

These examples show you how to structure the queries to access Excel data using the nickname compounds.

Selecting a specific column of information
The following query displays all compound_IDs where the molecular weight is greater than 200:
SELECT compound_ID FROM compounds
    WHERE molweight > 200;

Using an OR condition in your SELECT statements
The following query displays all records where the compound name or molecular weight is null:
SELECT * FROM compounds
    WHERE compoundname IS NULL OR molweight IS NULL;

Using LIKE and AND conditions in your SELECT statements
The following query displays all records where the compound name contains the string ase and the molecular weight is greater than or equal to 300:
SELECT * FROM compounds
    WHERE compoundname LIKE '%ase%' AND molweight >= 300;

Excel data source – sample scenario
This scenario shows the SQL statements that are necessary to register the federated objects that are used to access an Excel worksheet. Included in this scenario are several queries that you can run using the nickname that you create.

Excel worksheet information
This scenario starts with a worksheet that contains information about various compounds. The name of the workbook that the worksheet is in is Compound_Master.xls and the workbook was created in Excel. The fully-qualified path name to the workbook is C:\Data\Compound_Master.xls.
The first worksheet in the workbook contains four columns and nine rows of data. The columns list the names of the compounds, the weight of the compounds, the molecular count of the compound, and if the compound has been tested.

The contents of the worksheet are shown in the following table.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>COMPOUND_NAME</td>
<td>WEIGHT</td>
<td>MOL_COUNT</td>
</tr>
<tr>
<td>2</td>
<td>compound_A</td>
<td>1.23</td>
<td>367</td>
</tr>
<tr>
<td>3</td>
<td>compound_G</td>
<td>210</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>compound_F</td>
<td>0.000425536</td>
<td>174</td>
</tr>
<tr>
<td>5</td>
<td>compound_Y</td>
<td>1.00256</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>compound_Q</td>
<td>1024</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>compound_B</td>
<td>33.5362</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>compound_S</td>
<td>0.96723</td>
<td>67</td>
</tr>
<tr>
<td>9</td>
<td>compound_O</td>
<td>1.2</td>
<td></td>
</tr>
</tbody>
</table>

**Register the federated objects**

To access the worksheet using the Excel wrapper you must register the objects on the federated server:

1. Register the Excel wrapper.
   For example:
   
   ```sql
   CREATE WRAPPER Excel LIBRARY 'db2lsxls.dll';
   ```

2. Register the server definition:
   For example:

   ```sql
   CREATE SERVER biochem_lab WRAPPER Excel;
   ```

3. Register a nickname that refers to the Excel worksheet:
   For example:

   ```sql
   CREATE NICKNAME Compound_Master
   (compound_name VARCHAR(40),
    weight FLOAT,      
    mol_count INTEGER, 
    was_tested VARCHAR(20))
   FOR SERVER biochem_lab
   OPTIONS (FILE_PATH 'C:\Data\Compound_Master.xls');
   ```

The registration process is complete. The Excel worksheet is now part of the federated system, and can be used in SQL queries.

The following examples show the SQL queries and the results that are returned from the Compound_Master nickname.

**A query that returns all data that matches a specific WHERE clause condition**

To return all of the data for the compounds that have a molecular count that is greater than 100, issue this query:

```sql
SELECT * FROM Compound_Master
WHERE mol_count > 100;
```
**A query that returns specific columns from the worksheet**

To return the names and molecular counts for all of the compounds where the molecular count has not yet been determined, issue this query:

```sql
SELECT compound_name, mol_count FROM Compound_Master
WHERE mol_count IS NULL;
```

All of the columns from rows 2, 3, 4, 6, and 8 are returned.

The `compound_name` and `mol_count` columns from rows 5, 7, and 10 are returned.

**A query that counts the number of rows that match specific WHERE clause conditions**

To return the number of compounds that have a weight that is greater than 1 and that have not been tested, issue this query:

```sql
SELECT count(*) FROM Compound_Master
WHERE was_tested IS NULL AND weight > 1
```

The record count of 1 is returned. The compound in row 7 matches the query criteria.

**A query that returns specific columns from the worksheet and includes a subselect statement**

To return the names and molecular counts for all of the compound where the molecular count has been determined and the molecular count is less than the average molecular count, issue this query:

```sql
SELECT compound_name, mol_count FROM Compound_Master
WHERE mol_count IS NOT NULL
AND mol_count <
(SELECT AVG(mol_count) FROM Compound_Master
WHERE mol_count IS NOT NULL AND was_tested IS NOT NULL);
```

The subquery returns 368 for the molecular count average. The main query uses the average to return the query results that are shown in the following table:

<table>
<thead>
<tr>
<th>COMPOUND_NAME</th>
<th>MOL_COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>compound_A</td>
<td>367</td>
</tr>
<tr>
<td>compound_G</td>
<td>210</td>
</tr>
<tr>
<td>compound_F</td>
<td>174</td>
</tr>
<tr>
<td>compound_S</td>
<td>67</td>
</tr>
</tbody>
</table>

**File access control model for the Excel wrapper**

To access an Excel file, the wrapper needs a user identity for security purposes. The Excel wrapper uses the user identity that is associated with the federated database service. The name of the federated database service depends on the name of the database instance. For example, if the database instance name is DB2, then the service name is DB2 - DB2. To determine the user identity that is associated with federated database service, use the Control Panel in Windows to display the services. Double-click the service name and display the Log On properties page.
Chapter 7. Configuring access to HMMER data sources

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

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

HMMER wrapper

HMMER is a application package that you can use to search gene sequence databases that use statistical models or profile hidden Markov models (HMMs). The HMMER wrapper transforms the query statements into a format that the HMMER application package can interpret and starts the hmmpfam program or the hmmssearch program to run the query.

You can download the HMMER application package at no charge from [http://hmmer.wustl.edu/](http://hmmer.wustl.edu/) You can install the HMMER application package on a separate HMMER server or on the federated server.

An HMM is a statistical model of the primary structure consensus of a gene sequence family. An HMM is based upon probability models. You can train an HMM to recognize patterns from unaligned gene sequences if a trusted alignment is not yet known. You need less skill and manual intervention to train and use a successful HMM than to carefully construct a profile. You can use a trained HMM to access libraries of hundreds of profile HMMs and apply them on a very large scale to whole genome or Expressed Sequence Tag (EST) analyses.

PFAM (Protein Families Database of Alignments and HMMs) is a database of protein domain models. The HMMER application package is tightly tied to the construction and use of the PFAM database.

The HMMER application package contains the nine programs, but only two of these programs are supported by the IBM WebSphere Federation Server, the hmmpfam and the hmmssearch programs.

Table 59. The HMMER programs supported by the HMMER wrapper

<table>
<thead>
<tr>
<th>HMMER program</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hmmpfam</td>
<td>Uses a specific gene sequence to search an HMM database and determine the family that the test gene sequence might belong to. Calculates how well each model matches a specified sequence and a database of models. The match is expressed in terms of statistical significance.</td>
</tr>
<tr>
<td>hmmssearch</td>
<td>Uses a specific HMM profile to search a sequence database for significantly similar sequence matches.</td>
</tr>
</tbody>
</table>

Users or applications issue SQL query statements with HMMER-specific predicates to the federated server. The predicates in these statements map to command-line options in the hmmpfam or hmmssearch programs.
A special daemon program runs on the server where the HMMER application package is installed. This daemon receives the query request from the federated server and sends it to the HMMER application package. The HMMER application package runs the query on a profile database, such as PFAM.

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

![Diagram showing HMMER workflow](image)

**Figure 11. How the HMMER wrapper works**

The daemon returns the results to the HMMER wrapper. The wrapper transforms the data into a relational table, and returns this table to the user or application.

The following example shows how information is extracted from profile databases, which are constructed by HMMER programs, and displayed as a relational table. The HMMER User's Guide [http://hmmer.wustl.edu/](http://hmmer.wustl.edu/) provides examples of creating profile databases and a HMMER tutorial.

The example shows a sample query that uses the 7LES_DROME gene sequence. You specify sequences in the WHERE clause of the query.
The HMMER wrapper transforms the results from the query into the relational table as shown in the following table.

**Table 60. The HMMER results are transformed into a relational table**

<table>
<thead>
<tr>
<th>Model</th>
<th>ModelScore</th>
<th>DomainNumber</th>
<th>DomainScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>pkinase</td>
<td>+3.04100000000000E+002</td>
<td>1</td>
<td>+3.04100000000000E+002</td>
</tr>
<tr>
<td>fn3</td>
<td>+1.76300000000000E+002</td>
<td>1</td>
<td>+4.90000000000000E+001</td>
</tr>
<tr>
<td>fn3</td>
<td>+1.76300000000000E+002</td>
<td>2</td>
<td>+1.36000000000000E+001</td>
</tr>
<tr>
<td>fn3</td>
<td>+1.76300000000000E+002</td>
<td>3</td>
<td>+1.62000000000000E+001</td>
</tr>
<tr>
<td>fn3</td>
<td>+1.76300000000000E+002</td>
<td>4</td>
<td>+6.35000000000000E+001</td>
</tr>
<tr>
<td>fn3</td>
<td>+1.76300000000000E+002</td>
<td>5</td>
<td>+1.46000000000000E+001</td>
</tr>
<tr>
<td>fn3</td>
<td>+1.76300000000000E+002</td>
<td>6</td>
<td>+1.94000000000000E+001</td>
</tr>
<tr>
<td>rrm</td>
<td>-4.45000000000000E+001</td>
<td>1</td>
<td>-4.45000000000000E+001</td>
</tr>
</tbody>
</table>

**Figure 12. Sample query run on 7LES_DROME data**

Chapter 7. Configuring access to HMMER data sources 145
The data is now in a relational format and can be joined with data from other data sources.

**Adding HMMER data sources to a federated server**

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

**Before you begin**

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

**About this task**

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

**Procedure**

To add HMMER data sources to a federated server:

1. Verify that the correct version of the HMMER program executable files are installed.
2. Configure the HMMER daemon.
3. Start the HMMER daemon.
4. Register the wrapper.
5. Register the server definitions.
6. Optional: Create the user mappings.
7. Register the nickname.

**Verifying the version of the HMMER program executable**

You must have a supported version of the hmmpfam and hmmsearch executable files installed on the server where the HMMER application program is installed.

**About this task**

The HMMER application program can be installed on the federated server or a separate HMMER server.

**Procedure**

To check the version level of the HMMER executable files:

1. Issue a command that returns the version number.
   - For the hmmpfam program, the command is `hmmpfam -h`.
   - For the hmmsearch program, the command is `hmmsearch -h`.
2. In the output file, check the version of the executable files. You must have HMMER version 2.2g or later.
   - If you do not have the correct version, download the files from [http://hmmer.wustl.edu/](http://hmmer.wustl.edu/)
Configuring the HMMER daemon

A HMMER daemon is required by the HMMER wrapper to access HMMER data sources. The HMMER daemon must be configured before you register the HMMER wrapper.

Before you begin

The HMMER daemon must have:

- Execute access to the hmmpfam and hmmsearch executable files so that it can run HMMER searches.
- Write access to a directory in which it can write temporary files.
- Read access to at least one profile database on which you can run HMMER searches.

Restrictions

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

About this task

The HMMER wrapper requires a HMMER daemon. The HMMER daemon must be running on a server that you can access through TCP/IP from your federated system. This can be the same server that operates as the federated server, or a separate HMMER server.

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

During the installation of WebSphere Federation Server, a sample daemon configuration file is installed on the federated server. The name of the sample daemon configuration file is HMMER_DAEMON.config.

Procedure

To configure the HMMER daemon:

1. Ensure that the required files are on the server where HMMER is installed.
   During the installation of IBM WebSphere Federation Server, some of the required files are installed on the federated server. You must provide the other required files.
   On federated servers that run UNIX, the required files are:
   - The HMMER daemon executable file, $DB2PATH/bin/db2hmer_daemon.
   - The HMMER daemon configuration file, %DB2PATH%/samples/lifesci/HMMER_DAEMON.config.
   - The conversion utility, %DB2PATH%/bin/db2h2x.
   - The shell script, %DB2PATH%/bin/db2runpfam.ksh.
   - The HMMER program executable files, hmmpfam and hmmsearch. These files are not supplied by IBM.
   - The HMMER database files. These files are not supplied by IBM.
   On federated servers that run Windows, the required files are:
- The daemon executable files, `%DB2PATH%\bin\db2hmer_daemon.exe` and
  `%DB2PATH%\bin\db2hmer_daemon_svc.exe`.
- The HMMER daemon configuration file, `%DB2PATH%\samples\lifesci\HMMER_DAEMON.config`.
- The conversion utility, `%DB2PATH%\bin\db2h2x.exe`.
- The HMMER program executable files, hmmpfam.exe and hmmsearch.exe. These files are not supplied by IBM.
- The HMMER database files. These files are not supplied by IBM.

%DB2PATH% is the path where the federated database is installed.

By default, the daemon expects to find the configuration file in the working directory from which the daemon is started. You can copy the configuration file to another location. If you use a HMMER server, you must copy the daemon configuration file from the directory on the federated server to a directory on the HMMER server. You can copy the daemon configuration file to any directory on the HMMER server that the daemon can access.

2. On federated servers that run UNIX, ensure that the HMMER daemon executable file, conversion utility, and the shell script are executable.

To make the files executable, run the following command:
```
chmod a+x db2hmer_daemon db2h2x db2runpfam.ksh
```

3. Edit the daemon configuration file to work with your data source.
   a. Optional: You can rename the configuration file.
   b. Ensure that the first line in the configuration file is an equal sign. If the
      equal sign is missing, the daemon will not start. An error message will
      indicate that the DAEMON_PORT was not specified.
   c. Ensure that the last line in the configuration file ends with a new line.
      The sample configuration file that is provided with WebSphere Federation
      Server ends with a new line. If the last line does not end with a new line,
      you will receive an error message when you attempt to run your first
      HMMER query using the data source that is listed on the last line.
   d. Specify the following options in the configuration file. For options that
      require paths, you can specify the relative paths. Relative paths are relative
      to the directory from which the daemon process was started.

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

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

   **DAEMON_LOGFILE_DIR**
   This is the directory in which the daemon creates its log file. This
   file contains useful status and error information generated by the
   HMMER daemon.
Q_SEQ_DIR_PATH
This is the directory in which a temporary query sequence data file
is created by the daemon. This temporary file is cleaned up once the
HMMER job completes.

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

RUNPFAM_PATH
This is the fully-qualified name of the db2runpfam.ksh shell script
provided with WebSphere Federation Server. This option is ignored
if it is specified on Windows.

HMMERPFAM_PATH
This is the fully-qualified name of the HMMER executable file on
the computer that is running the daemon. On UNIX, the name of
the file is hmmpfam. On Windows, the name of the file is
hmmpfam.exe.

HMMSEARCH_PATH
This is the fully-qualified name of the HMMER executable file on
the computer that is running the daemon. On UNIX, the name of
the file is hmmssearch. On Windows, the name of the file is
hmmssearch.exe.

H2X_PATH
This is the fully-qualified name of the conversion program
(HMMER to XML) provided with the daemon. On UNIX, the name
of the program is db2h2x. On Windows, the name of the program is
db2h2x.exe.

database specification entry
Specifies the location of a profile database or sequence file. Make
note of the database data_source_name that you specify in the
configuration file. For the daemon to function properly, you must
specify the database data_source_name when you create the nickname
for the data source. The name is case-sensitive. The database
data_source_name is specified in:

• The DATASOURCE option of the CREATE NICKNAME
  statement (for hmmpfam)
• The MODEL predicate of the CREATE NICKNAME statement
  (for hmmssearch)

The configuration file must contain at least one database
specification entry that specifies the fully qualified path for the
profile or database. For example:

data_source_name=profile_or_sequence_database_name

On federated servers that run UNIX
For example, to specify the MYHMM profile database you
would add the following line to the daemon configuration
file:

myhmm=/home/user_ID/myhmm
On federated servers that run Windows
For example, to specify MYHMMS profile database you would add the following line to the daemon configuration file:

```
myhmms=c:\hmmer\tutorial\myhmms
```

HMMER daemon configuration file - examples
The examples show the required settings and specifications for PFAM and SEARCH data sources. There are daemon configuration file examples for both UNIX and Windows servers.

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

HMMER_DAEMON.config file for UNIX

This example shows the required options and profile database specification for a server that runs UNIX.

```
=  
DAEMON_PORT=4098
MAX_PENDING_REQUESTS=10
DAEMON_LOGFILE_DIR=./
Q_SEQ_DIR_PATH=./
HMMER_OUT_DIR_PATH=./
RUNPFAM_PATH=./db2runpfam.ksh
HMMPFAM_PATH=./db2runpfam.ksh /home/user_id/hmmer/bin/hmmpfam
HMMSEARCH_PATH=./db2runpfam.ksh /home/user_id/hmmer/bin/hmmsearch
H2X_PATH=/home/user_id/sql\lib\bin\db2h2x
myhmms=/home/user_id/hmmer/tutorial/myhmms
globin=/home/user_id/hmmer/tutorial/globin.hmm
pfams=/home/user_id/hmmer/pfam/Pfam_ls
```

HMMER_DAEMON.config file for Windows

This example shows the required options and profile database specification for a server that runs Windows.

```
=  
DAEMON_PORT=4098
MAX_PENDING_REQUESTS=10
DAEMON_LOGFILE Dir=.
Q_SEQ_DIR_PATH=.
HMMER_OUT_DIR_PATH=.
HMMPFAM_PATH=c:\hmmer\bin\hmmpfam.exe
HMMSEARCH_PATH=c:\hmmer\bin\hmmsearch.exe
H2X_PATH=\db2h2x.exe
myhmms=c:\hmmer\tutorial\myhmms
globin=c:\hmmer\tutorial\globin.hmm
pfamseq=c:\hmmer\pfam\pfamseq
```

Starting the HMMER daemon
Before you can access HMMER data sources, you must start the HMMER daemon.

Before you begin
Before you start the HMMER daemon, you must have write access to all paths listed under the DAEMON_LOGFILE_DIR, HMMER_OUT_DIR_PATH, and Q_SEQ_DIR_PATH entries in the configuration file.
The executable file starts a new process in which the HMMER daemon runs.

Procedure

To start the HMMER daemon:
1. Open the directory where the daemon executable file is located.
2. Issue the `db2hmmer_daemon` command to run the executable files with the parameters that you need.
   
   There are options that you can specify when you issue the command.

`db2hmmer_daemon command - options and examples`

The `db2hmmer_daemon` command can be used on UNIX and Windows servers. Some of the options listed in the syntax can be used only on Windows servers.

**Options - db2hmmer_daemon command**

The options for the `db2hmmer_daemon` command are:

```
db2hmmer_daemon -a action -c config_file -d debug_level
     -u user_id -p password
```

- **-a action**
  
  Performs the specified activity. Valid actions are `status`, `install`, `start`, `stop`, and `remove`.
  
  You can specify this option only on Windows servers.

- **-c config_file**
  
  Instructs the daemon to use the specified configuration file. If you do not specify the configuration file, the daemon searches for the HMMER_DAEMON.config file in the directory where the daemon executable files are installed.
  
  You can specify this option on UNIX and Windows servers. On Windows servers, this option can be specified only with the `install` or `start` actions.

- **-d debug_level**
  
  Sets the daemon debug level to the specified value. The valid values are 1, 2, or 3. You can use this option with the `install` and `start` actions.
  
  You can specify this option on UNIX and Windows servers.

- **-u user_id**
  
  Sets the daemon to run under the specified user ID. You can use this option with the `install` action.
  
  You can specify this option only on Windows servers.

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

The options that are specified with the `start` action apply only until the daemon is stopped, and override the values that are specified with the `install` action. If the daemon is stopped and then started without the option, then the value of the option is whatever was specified with the install action, or the built-in default for the daemon.
For example, if the commands are issued in the following order, the HMMER1.config file is specified with the *install* action. The HMMER2.config file is specified with the *start* action and used until the daemon is stopped. When the daemon is started again, the HMMER1.config file is used.

\[1\] db2hmer_daemon -a install -c HMMER1.config  
\[2\] db2hmer_daemon -a start -c HMMER2.config  
\[3\] db2hmer_daemon -a stop  
\[4\] db2hmer_daemon -a start

**Examples - db2hmer_daemon command**

The following examples show you how to use the options with the db2hmer_daemon command.

**Start the daemon**

To start the daemon, issue the following command:

```
  db2hmer_daemon -a start
```

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

**Daemon configuration file**

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

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

```
  dbhmer_daemon -c cfg/HMMER_D.config
```

**Debugging**

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

```
  db2hmer_daemon -a install -d 2  
  db2hmer_daemon -a start
```

**Status of the daemon (Windows)**

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

```
  db2hmer_daemon -a status
```

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

**Stop the daemon**

To stop the daemon on Windows, issue the following command:

```
  db2hmer_daemon -a stop
```

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

To stop the HMMER daemon on a UNIX server, determine the process ID of the db2hmer_daemon by issuing the `ps -ef | grep db2hmer` command. Issue the command `kill nnnn`, where `nnnn` is the process ID of the db2hmer_daemon.
Remove the daemon

To remove the daemon, issue the following command:

db2hammer_daemon -a remove

Registering the HMMER wrapper

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

If you use a proxy server to access HMMER files, you must specify the proxy information as options when you register the wrapper. If you also specify proxy options when you register a server definition, the server options take precedence.

Procedure

To register the HMMER wrapper:

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>From the command line</td>
<td>Issue the CREATE WRAPPER statement.</td>
</tr>
</tbody>
</table>

CREATE WRAPPER wrapper_name
LIBRARY library_name;

If you use a proxy server to access HMMER data sources, you issue this statement:

CREATE WRAPPER wrapper_name
LIBRARY library_name
OPTIONS (PROXY_TYPE 'type',
PROXY_SERVER_NAME 'server_name',
PROXY_SERVER_PORT 'port_number');

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

HMMER wrapper library files

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

When you install IBM WebSphere Federation Server, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2lshmmer.a, libdb2lshmmerF.a, and libdb2lshmmerU.a. The default wrapper library file is libdb2lshmmer.a. The other wrapper library files are used with specific wrapper options.

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

The default directory paths and default wrapper library file names are listed in the following table.
<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Wrapper library file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/&lt;install_path&gt;/lib32/</td>
<td>libdb2lshmmer.a</td>
</tr>
<tr>
<td></td>
<td>/usr/opt/&lt;install_path&gt;/lib64/</td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib32</td>
<td>libdb2lshmmer.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib64</td>
<td></td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib32</td>
<td>libdb2lshmmer.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib64</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>%DB2PATH%in</td>
<td>db2lshmmer.dll</td>
</tr>
</tbody>
</table>

<install_path> is the directory path where the federated server is installed on UNIX or Linux.

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

**CREATE WRAPPER statement - Examples for the HMMER wrapper**

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

**Example of registering a wrapper**

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

```
CREATE WRAPPER hammer_wrapper LIBRARY 'libdb2lshmmer.a';
```

`hammer_wrapper` is a name that you assign to the HMMER wrapper. Duplicate wrapper names are not allowed.

`LIBRARY 'libdb2lshmmer.a'` is the name of the wrapper library file for federated servers that use AIX operating systems.

**Example of registering a wrapper for a SOCKS proxy server**

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

```
CREATE WRAPPER hammer_wrapper LIBRARY 'libdb2lshmmer.so'
  OPTIONS (PROXY_TYPE 'SOCKS',
            PROXY_SERVER_NAME 'proxy_socks',
            PROXY_SERVER_PORT '1081');
```

`LIBRARY 'libdb2lshmmer.so'` is the name of the wrapper library file for federated servers that use Linux and Solaris operating systems.

`PROXY_TYPE 'SOCKS'` specifies the proxy type that is used to access the Internet when behind a firewall.
PROXY_SERVER_NAME 'proxy_socks'
  Specifies the proxy server name or IP address. This option is required if the
  value for the PROXY_TYPE server option is 'SOCKS'.

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

Registering the server definitions for a HMMER data source

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

About this task

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

Procedure

To register a server definition for a HMMER data source:
1. Choose the method that you want to use to register the server definition:

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>From the command line</td>
<td>Issue the CREATE SERVER statement. For example:</td>
</tr>
<tr>
<td></td>
<td>CREATE SERVER server_definition_name</td>
</tr>
<tr>
<td></td>
<td>TYPE HMMER_search_type</td>
</tr>
<tr>
<td></td>
<td>VERSION version_number</td>
</tr>
<tr>
<td></td>
<td>WRAPPER wrapper_name</td>
</tr>
<tr>
<td></td>
<td>OPTIONS (NODE 'node_name',</td>
</tr>
<tr>
<td></td>
<td>DAEMON_PORT 'port_number');</td>
</tr>
<tr>
<td></td>
<td>If you use a proxy server to access HMMER data sources, the statement that you issue is:</td>
</tr>
<tr>
<td></td>
<td>CREATE SERVER server_definition_name</td>
</tr>
<tr>
<td></td>
<td>VERSION version_number</td>
</tr>
<tr>
<td></td>
<td>WRAPPER wrapper_name</td>
</tr>
<tr>
<td></td>
<td>OPTIONS (PROXY_TYPE 'type',</td>
</tr>
<tr>
<td></td>
<td>PROXY_SERVER_NAME 'server_name',</td>
</tr>
<tr>
<td></td>
<td>PROXY_SERVER_PORT 'port_number');</td>
</tr>
<tr>
<td></td>
<td>When you register the server definition, you specify server options in the</td>
</tr>
<tr>
<td></td>
<td>CREATE SERVER statement. There are required server options and optional</td>
</tr>
<tr>
<td></td>
<td>server options. The NODE and DAEMON_PORT server options are required</td>
</tr>
<tr>
<td></td>
<td>for HMMER data sources.</td>
</tr>
</tbody>
</table>

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

After the server definition is registered, use the ALTER SERVER statement to add
or drop server options.
CREATE SERVER statement - Examples for HMMER wrapper

Use the CREATE SERVER statement to register server definitions for the HMMER wrapper. This topic includes an example with the required parameters and an example with additional server options.

**Required parameters**

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

```
CREATE SERVER hmmpfam_server TYPE PFAM
  VERSION 2.2 WRAPPER hmmer_wrapper
  OPTIONS(NODE 'someserver.someschool.edu', DAEMON_PORT '4422');
```

- **hmmpfam_server**
  A name that you assign to the HMMER server. Duplicate server definition names are not allowed.

- **TYPE PFAM**
  The type of HMMER search that this associated with the hmmpfam_server server definition. The valid values for TYPE are PFAM (for hmmpfam) or SEARCH (for hmmsearch).

- **VERSION 2.3**
  The version of the hmmpfam or hmmsearch program executable file that you are using. The supported versions are HMMER 2.2g (or later).

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

- **NODE 'someserver.someschool.edu'**
  Specifies the host name or IP address of the server on which the HMMER daemon process is running. This value is case sensitive.

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

- **DAEMON_PORT '4422'**
  The port number on which the daemon listens for HMMER job requests. The port number must be the same number that you specified in the DAEMON_PORT option for the HMMER daemon configuration file. The default is port number 4098.

**A server definition for the hmmpfam program with optional parameters**

This example shows a server definition for a hmmpfam program:

```
CREATE SERVER hmmpfam_server TYPE PFAM
  VERSION 2.2 WRAPPER hmmer_wrapper
  OPTIONS(NODE 'someserver.someschool.edu', DAEMON_PORT '4422',
    PROCESSORS '2', HMMPFAM_OPTIONS '--null2 --pvm');
```

- **PROCESSORS '2'**
  Specifies the number of processors that the HMMER program uses. This option is equivalent to the --cpu option for the hmmpfam and hmmsearch programs.

- **HMMPFAM_OPTIONS '--null2 --pvm'**
  Use this server option to specify options that are used by the hmmpfam program. These options cannot be specified in a predicate. In this example, the --null2 and --pvm options will be used whenever a query is run that
uses the `hmmpfam_server` server definition. The HMMPFAM_OPTIONS server option is only valid with server definition that specify TYPE PFAM.

A server definition for the hmmsearch program with optional parameters

This example shows a server definition for a hmmsearch program:

```plaintext
CREATE SERVER hmmsearch_server TYPE SEARCH
    VERSION 2.2 WRAPPER hmmer_wrapper
    OPTIONS(NODE 'someserver.someschool.edu', DAEMON_PORT '4422',
        PROCESSORS '2', HMMSEARCH_OPTIONS '--null2 --pvm');
```

**PROCESSORS '2'**

 Specifies the number of processors that the HMMER program uses. This option is equivalent to the `--cpu` option for the hmmpfam and hmmsearch programs.

**HMMSEARCH_OPTIONS ' --null2 --pvm'**

 Use this server option to specify options that are used by the hmmsearch program. These options cannot be specified in a predicate. In this example, the `--null2` and `--pvm` options will be used whenever a query is run that uses the `hmmsearch_server` server definition. The HMMSEARCH_OPTIONS server option is only valid with server definition that specify TYPE SEARCH.

Server definitions when a proxy server is used

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

- You want to retrieve data by using a URI.
- The URI retrieves data from behind a firewall, through a proxy.

Check with your network administrator for information about the settings that you should specify in the proxy options.

Registering a server definition for a SOCKS proxy server

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

```plaintext
CREATE SERVER hmmer_serversocks
    WRAPPER hmmer_wrapper
    OPTIONS (PROXY_TYPE 'SOCKS', PROXY_SERVER_NAME 'proxy_socks',
        PROXY_SERVER_PORT '1081');
```

**hmmer_serversocks**

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

**WRAPPER hmmer_wrapper**

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

**PROXY_TYPE 'SOCKS'**

 Specifies the proxy type that is used to access the Internet when behind a firewall.

**PROXY_SERVER_NAME 'proxy_socks'**

 Specifies the proxy server name or IP address. This option is required if the value for the PROXY_TYPE server option is 'SOCKS'.
PROXY_SERVER_PORT '1081'
  Specifies the proxy server port number. This option is required if the value for the PROXY_TYPE server option is 'SOCKS'.

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

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

```
CREATE SERVER xml_server_socks
  WRAPPER hmmer_wrapper
  OPTIONS (PROXY_TYPE 'SOCKS', PROXY_SERVER_NAME 'proxy_socks',
    PROXY_SERVER_PORT '1081', PROXY_AUTHID 'Laura',
    PROXY_PASSWORD 'not4me');
```

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

PROXY_PASSWORD 'not4me'
  Specifies the password on the proxy server that is associated with the user name 'Laura'. This server option is required when the value for the PROXY_TYPE server option is 'SOCKS'.

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

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

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

About this task

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

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

Procedure

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

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Method</td>
<td>Procedure</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Using the command line</td>
<td>Issue the CREATE USER MAPPING statement. For example:</td>
</tr>
<tr>
<td></td>
<td>CREATE USER MAPPING FOR local_userID</td>
</tr>
<tr>
<td></td>
<td>SERVER server_definition_name</td>
</tr>
<tr>
<td></td>
<td>OPTIONS (REMOTE_AUTHID 'remote_userID',</td>
</tr>
<tr>
<td></td>
<td>REMOTE_PASSWORD 'remote_password')</td>
</tr>
<tr>
<td></td>
<td>PROXY_AUTHID 'proxy_server_userID',</td>
</tr>
<tr>
<td></td>
<td>PROXY_PASSWORD 'proxy_server_password'</td>
</tr>
</tbody>
</table>

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

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

**CREATE USER MAPPING statement - Examples for the HMMER wrapper**

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

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

**Proxy server example**

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

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

```sql
CREATE USER MAPPING FOR katherine SERVER hmmer_proxy
    OPTIONS (REMOTE_AUTHID 'kathy',
              REMOTE_PASSWORD 'all4one',
              PROXY_AUTHID 'kate',
              PROXY_PASSWORD 'them2us');
```

- **katherine** specifies the local user ID that you are mapping to a user ID that is defined at the HMMER proxy server.
- **SERVER hmmer_server** specifies the server definition name that you registered in the CREATE SERVER statement for the HMMER server.
- **REMOTE_AUTHID 'kathy'** specifies the user ID at the HMMER server to which you are mapping `katherine`. This remote ID must be in a format that is expected by the HMMER server.
- **REMOTE_PASSWORD 'all4one'** specifies the password that is associated with 'kathy'.

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

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

Registering nicknames for HMMER data sources
For each HMMER server definition that you register, you must register a nickname for each HMMER database that you want to access. Use these nicknames when you query the HMMER databases.

About this task
When you register a HMMER nickname, a set of fixed input and output columns for the profile database are automatically created with the nickname. The fixed columns are created in the federated database system catalog. You reference the fixed columns in SQL queries.

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

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

Procedure
To register a nickname for a HMMER database:

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Using the command line</td>
<td>Issue the CREATE NICKNAME statement. For example: CREATE NICKNAME nickname FOR SERVER server_definition_name OPTIONS (DATASOURCE data_source_name);</td>
</tr>
</tbody>
</table>

The data_source_name must match an existing data_source_name in the HMMER_DAEMON.config file on the HMMER server.

Repeat this step for each HMMER database that you want to create a nickname for.

CREATE NICKNAME statement - examples for HMMER wrapper
Use the CREATE NICKNAME statement to register a nickname for a HMMER profile database that you want to access. This topic includes an example with the required parameters and an example with additional nickname options.
Required parameters example

The following example shows you how to register a nickname for searches that use the hmmpfam program. This example uses the hmmpfam_server server definition to access the 'myhmms' data source.

```
CREATE NICKNAME hmmpfam_nickname
FOR SERVER hmmpfam_server
  OPTIONS(DATASOURCE 'myhmms')
```

- **hmmpfam_nickname**
  - A name that you assign to the nickname. This name must be unique.

- **SERVER hmmpfam_server**
  - The name of the server definition that you want the nickname to be associated with.

- **DATASOURCE 'myhmms'**
  - The name of the database that you will run HMMER searches on. This database must be listed in the HMMER daemon configuration file.

  Although the DATASOURCE nickname option is specified as an option in the CREATE NICKNAME statement, it is required for HMMER data sources.

Optional nickname options example

You can specify how long the HMMER wrapper waits for the results from the HMMER daemon. The following example shows you how to register a nickname for searches that use the hmmpfam program and how to specify the TIMEOUT nickname option.

```
CREATE NICKNAME hmmpfam_nickname
FOR SERVER hmmpfam_server
  OPTIONS(DATASOURCE 'myhmms', TIMEOUT '30')
```

- **TIMEOUT '30'**
  - The maximum time, in minutes, that the HMMER wrapper waits for results from the daemon. The default time is 5 minutes.

Fixed columns for HMMER nicknames

When you issue the CREATE NICKNAME statement for a HMMER data source, a set of fixed columns are automatically created with the nickname. You can reference the fixed columns in SQL queries.

There are fixed input columns and fixed output columns that are automatically created when you register a HMMER nickname. The fixed columns are added to the federated database system catalog.

If you want to change the default data type that is assigned to a fixed column, you can specify the column name and data type in the CREATE NICKNAME statement.

For example, to limit the output for the AlignmentConsensus column to no more than the first 100 characters, you issue the following statement:

```
CREATE NICKNAME nucleol
  (AlignmentConsensus VARCHAR(100))
FOR SERVER searchtest
  OPTIONS (DATASOURCE 'nucleol', TIMEOUT '1');
```
**Fixed input columns for HMMER nicknames**

The fixed input columns are used as parameter-passing predicates in SQL queries.

The input columns pass standard HMMER switches to either the hmmpfam or hmmsearch programs. The HMMER program then runs on the specified data source using these switches. Fixed input columns can also be referenced in the query SELECT list and are returned as part of the results table.

Each nickname that you create is associated with a specific server definition. The server definitions are designated as either a PFAM type of server or a SEARCH type of server. There are different fixed input columns for each type of server. You can specify these columns in the WHERE clause of your SQL statements.

**Fixed input columns for PFAM**

The following table lists the fixed input columns that you can use with the hmmpfam program.

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
<th>Operator</th>
<th>Switches</th>
<th>Returned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HmmQSeq</td>
<td>VARCHAR(32000)</td>
<td>Input gene sequence that is used to search</td>
<td>=</td>
<td>-E n</td>
<td>Same as the input value that you specify. This column is required.</td>
</tr>
<tr>
<td>ModelEValue</td>
<td>DOUBLE</td>
<td>Estimated e-value</td>
<td>&lt;</td>
<td>-E n</td>
<td>See the query output.</td>
</tr>
<tr>
<td>ModelScore</td>
<td>DOUBLE</td>
<td>Raw score</td>
<td>&gt;</td>
<td>-T n</td>
<td>See the query output.</td>
</tr>
<tr>
<td>DBSize</td>
<td>INTEGER</td>
<td>Calculate e-values as if the database had 'n' gene sequences</td>
<td>=</td>
<td>-Z n</td>
<td>Same as the input value that you specify. Uses hmmpfam default if not specified.</td>
</tr>
<tr>
<td>CutMode</td>
<td>CHAR(2)</td>
<td>Cutoff mode; can be ga, tc or nc (case sensitive)</td>
<td>=</td>
<td>--cut_ga --cut_tc --cut_nc</td>
<td>Same as the input value that you specify. NULL if not specified.</td>
</tr>
<tr>
<td>DomainScore</td>
<td>DOUBLE</td>
<td>Domain score</td>
<td>&gt;</td>
<td>--domT n</td>
<td>See the query output.</td>
</tr>
<tr>
<td>DomainEValue</td>
<td>DOUBLE</td>
<td>Domain e-value</td>
<td>&lt;</td>
<td>--domE n</td>
<td>See the query output.</td>
</tr>
<tr>
<td>ForwardAlgorithm</td>
<td>CHAR</td>
<td>Use Forward algorithm rather than Viterbi; value can be 'Y' or 'N'</td>
<td>=</td>
<td>--forward</td>
<td>Same as the input value that you specify. 'N' is the default.</td>
</tr>
</tbody>
</table>

**Fixed input columns for SEARCH**

The following table lists the fixed input columns that you can use with the hmmsearch program.
Table 63. Fixed input columns for servers of type SEARCH

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
<th>Operator</th>
<th>Options</th>
<th>Returned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>VARCHAR(32000)</td>
<td>Name of the HMM profile file used in the search.</td>
<td>=</td>
<td></td>
<td>Same as the input value that you specify. This column is required.</td>
</tr>
<tr>
<td>SequenceEValue</td>
<td>DOUBLE</td>
<td>Estimated e-value</td>
<td>&lt;</td>
<td>-E n</td>
<td>See the query output.</td>
</tr>
<tr>
<td>SequenceScore</td>
<td>DOUBLE</td>
<td>Raw score</td>
<td>&gt;</td>
<td>-T n</td>
<td>See the query output.</td>
</tr>
<tr>
<td>DBSize</td>
<td>INTEGER</td>
<td>Calculate e-values as if the database had 'n' gene sequences</td>
<td>=</td>
<td>-Z n</td>
<td>Same as the input value that you specify. Uses hmmpfam default if not specified.</td>
</tr>
<tr>
<td>CutMode</td>
<td>CHAR(2)</td>
<td>Cutoff mode; can be ga, tc or nc (case sensitive)</td>
<td>=</td>
<td>--cut_ga</td>
<td>Same as the input value that you specify. NULL if not specified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>--cut_tc</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>--cut_nc</td>
<td></td>
</tr>
<tr>
<td>DomainScore</td>
<td>DOUBLE</td>
<td>Domain score</td>
<td>&gt;</td>
<td>--domT n</td>
<td>See the query output.</td>
</tr>
<tr>
<td>DomainEValue</td>
<td>DOUBLE</td>
<td>Domain e-value</td>
<td>&lt;</td>
<td>--domE n</td>
<td>See the query output.</td>
</tr>
<tr>
<td>ForwardAlgorithm</td>
<td>CHAR</td>
<td>Use Forward algorithm rather than Viterbi; value can be 'Y' or 'N'</td>
<td>=</td>
<td>--forward</td>
<td>Same as the input value that you specify. 'N' is the default.</td>
</tr>
</tbody>
</table>

**Fixed output columns for HMMER nicknames**

The fixed output columns are specified in the SELECT list in your queries. You can also specify fixed output columns as predicates in the WHERE clause.

**Fixed output columns for PFAM**

Each nickname that you create is associated with a specific server definition. The server definitions are designated as either a PFAM type of server or a SEARCH type of server.

The following tables list the fixed output columns for each type of server.

The following table lists the fixed columns that are returned as output for PFAM.

Table 64. Fixed output columns for servers of type PFAM

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>VARCHAR(32)</td>
<td>Name of model.</td>
</tr>
<tr>
<td>ModelDescription</td>
<td>VARCHAR(64)</td>
<td>Text description of model.</td>
</tr>
<tr>
<td>ModelScore</td>
<td>DOUBLE</td>
<td>Raw score (”bit score”).</td>
</tr>
<tr>
<td>ModelEValue</td>
<td>DOUBLE</td>
<td>Estimated e-value.</td>
</tr>
</tbody>
</table>
### Table 64. Fixed output columns for servers of type PFAM (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModelHits</td>
<td>INTEGER</td>
<td>Number of domains hit within the model.</td>
</tr>
<tr>
<td>DomainNumber</td>
<td>INTEGER</td>
<td>Specific domain (within one model).</td>
</tr>
<tr>
<td>SequenceFrom</td>
<td>INTEGER</td>
<td>Starting point of gene sequence.</td>
</tr>
<tr>
<td>SequenceFromGlobal</td>
<td>CHAR</td>
<td>'Y' if the alignment starts at the beginning of the gene sequence.</td>
</tr>
<tr>
<td>HmmFrom</td>
<td>INTEGER</td>
<td>Starting point of consensus model.</td>
</tr>
<tr>
<td>HmmFromGlobal</td>
<td>CHAR</td>
<td>'Y' if the alignment starts at the beginning of the consensus model.</td>
</tr>
<tr>
<td>HmmTo</td>
<td>INTEGER</td>
<td>Ending point in consensus model.</td>
</tr>
<tr>
<td>HmmToGlobal</td>
<td>CHAR</td>
<td>'Y' if the alignment ends at the end of the consensus model.</td>
</tr>
<tr>
<td>DomainScore</td>
<td>DOUBLE</td>
<td>Raw score (&quot;bit score&quot;) for the isolated domain.</td>
</tr>
<tr>
<td>DomainEValue</td>
<td>DOUBLE</td>
<td>Expected value for the isolated domain.</td>
</tr>
<tr>
<td>AlignmentConsensus</td>
<td>VARCHAR(32000)</td>
<td>The HMM consensus. The amino acid shown for the consensus is the highest probability amino acid at that position according to the HMM, not necessarily the highest scoring amino acid.</td>
</tr>
<tr>
<td>AlignmentExactMatch</td>
<td>VARCHAR(32000)</td>
<td>Matches the highest probability residue in the HMM.</td>
</tr>
<tr>
<td>AlignmentSubSequence</td>
<td>VARCHAR(32000)</td>
<td>Shows the gene sequence itself.</td>
</tr>
</tbody>
</table>

### Fixed output columns for SEARCH

The following table lists the fixed columns that are returned as output for SEARCH.

### Table 65. Fixed output columns for servers of type SEARCH

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence</td>
<td>VARCHAR(32)</td>
<td>The sequence identifier.</td>
</tr>
<tr>
<td>SequenceDescription</td>
<td>VARCHAR(64)</td>
<td>Text description of the sequence.</td>
</tr>
<tr>
<td>SequenceScore</td>
<td>DOUBLE</td>
<td>Raw score (&quot;bit score&quot;).</td>
</tr>
<tr>
<td>SequenceEValue</td>
<td>DOUBLE</td>
<td>Estimated e-value.</td>
</tr>
<tr>
<td>SequenceHits</td>
<td>INTEGER</td>
<td>Number of domains hit within the sequence.</td>
</tr>
<tr>
<td>DomainNumber</td>
<td>INTEGER</td>
<td>Specific domain (within one sequence).</td>
</tr>
<tr>
<td>SequenceFrom</td>
<td>INTEGER</td>
<td>Starting point of gene sequence.</td>
</tr>
<tr>
<td>SequenceFromGlobal</td>
<td>CHAR</td>
<td>'Y' if the alignment starts at the beginning of the gene sequence.</td>
</tr>
<tr>
<td>HmmFrom</td>
<td>INTEGER</td>
<td>Starting point of consensus model.</td>
</tr>
<tr>
<td>HmmFromGlobal</td>
<td>CHAR</td>
<td>'Y' if the alignment starts at the beginning of the consensus model.</td>
</tr>
<tr>
<td>HmmTo</td>
<td>INTEGER</td>
<td>Ending point in consensus model.</td>
</tr>
<tr>
<td>HmmToGlobal</td>
<td>CHAR</td>
<td>'Y' if the alignment ends at the end of the consensus model.</td>
</tr>
<tr>
<td>DomainScore</td>
<td>DOUBLE</td>
<td>Raw score (&quot;bit score&quot;) for the isolated domain.</td>
</tr>
<tr>
<td>DomainEValue</td>
<td>DOUBLE</td>
<td>Expected value for the isolated domain.</td>
</tr>
</tbody>
</table>
Table 65. Fixed output columns for servers of type SEARCH (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlignmentConsensus</td>
<td>VARCHAR(32000)</td>
<td>The HMM consensus. The amino acid shown for the consensus is the highest probability amino acid at that position according to the HMM, not necessarily the highest scoring amino acid.</td>
</tr>
<tr>
<td>AlignmentExactMatch</td>
<td>VARCHAR(32000)</td>
<td>Matches the highest probability residue in the HMM.</td>
</tr>
<tr>
<td>AlignmentSubSequence</td>
<td>VARCHAR(32000)</td>
<td>Shows the gene sequence itself.</td>
</tr>
</tbody>
</table>

**HMMER data source - complete example**

All the SQL statements that you need to issue to add HMMER data sources to a federated server are shown in this example. This examples also shows you how to use a nickname in a query.

The SQL queries for HMMER data sources must contain special input predicates that are used to pass standard HMMER options to the program executable file.

To be valid, every query passed to the HMMER wrapper must contain at least the HmmQSeq input predicate (for TYPE PFAM) or model predicate (for TYPE SEARCH). All other predicates are optional.

To construct a query against a HMMER nickname, specify the input columns in the WHERE clause and the output columns in the SELECT list.

**Example for the hmmpfam program**

This example creates a wrapper, server definition, and nickname on an AIX federated server for the hmmpfam program. This example also runs a query that uses a string literal for the search sequence. The SQL statements to register the federated objects are:

```sql
CREATE WRAPPER hmmer_wrapper LIBRARY 'libdb2lshmmer.a';

CREATE SERVER hmmpfam_server TYPE pfam
    VERSION 2.2 WRAPPER hmmer_wrapper
    OPTIONS(NODE 'HMMERServ.MyCompany.com');

CREATE NICKNAME hmmpfam_nickname
    FOR SERVER hmmpfam_server
    OPTIONS(DATASOURCE 'myhmms', TIMEOUT '1');
```

Run the 7LES_DROME gene sequence on the hmmpfam_nickname. The SQL query is:

```sql
SELECT Model, substr(ModelDescription,1,50) as ModelDescription,
    ModelScore, ModelEValue, ModelHits, DomainNumber,
    SequenceFrom, SequenceTo, SequenceFromGlobal, SequenceToGlobal,
    HmmFrom, HmmTo, HmmFromGlobal, HmmToGlobal, DomainScore, DomainEValue,
    length(HmmQSeq) as "length(HmmQSeq)",
    length(AlignmentConsensus) as "length(AConsensus)",
    length(AlignmentMatch) as "length(AMatch)",
    length(AlignmentSubSeq) as "length(ASubSeq)",
    substr(HmmQSeq,1,64) as HmmQSeq,
    substr(AlignmentConsensus,1,64) as AlignmentConsensus,
    substr(AlignmentMatch,1,64) as AlignmentMatch,
    substr(AlignmentSubSeq,1,64) as AlignmentSubSeq
FROM hmmpfam_nickname
WHERE HmmQSeq = 'MTMFWQNNVHDQSDEQDKQAAPTKRLNISFNVKIAVNVNTKMTTTHINQQAPGTSS';
```

Chapter 7. Configuring access to HMMER data sources 165
Example for the hmmsearch program

This example creates a wrapper, server definition, and nickname on a Windows federated server for the hmmsearch program. This example also runs a query that uses a string literal for the search sequence.

The SQL statements to register the federated objects are:

```sql
CREATE WRAPPER hmm_wrapper LIBRARY 'db2lshmmer.dll';
CREATE SERVER hmmsearch_serv TYPE search VERSION 2.2
  WRAPPER hmm_wrapper
    OPTIONS(NODE 'localhost');
CREATE NICKNAME artemia FOR SERVER hmmsearch_server
  OPTIONS(DATASOURCE 'artemia', TIMEOUT '1');
```

The SQL query is:

```sql
SELECT Model, Sequence, substr(SequenceDescription,1,50) as SequenceDescription,
  SequenceScore, SequenceEValue, SequenceHits, DomainNumber,
  SequenceFrom, SequenceTo, SequenceFromGlobal, SequenceToGlobal,
  HmmFrom, HmmTo, HmmFromGlobal, HmmToGlobal, DomainScore, DomainEValue,
  length(AlignmentConsensus) as "length(AConsensus)",
  length(AlignmentMatch) as "length(AMatch)",
  length(AlignmentSubSeq) as "length(ASubSeq)",
  substr(AlignmentConsensus,1,200) as AlignmentConsensus,
  substr(AlignmentMatch,1,200) as AlignmentMatch,
  substr(AlignmentSubSeq,1,200) as AlignmentSubSeq
FROM artemia
WHERE Model = 'globin' and DomainScore > 50;
```

Construct new HMMER queries with samples

The following samples show you how to construct queries for HMMER data sources.

In these queries, the nickname is a name that describes the type of HMMER search and the data source, for example hmmpfam_nickname. Some examples also show how to use the HMMER wrapper with other data sources.

**Query 1.**

When this SQL statement runs, the wrapper uses the indicated sequence and the HMM database defined by the nickname to run the hmmpfam program. The wrapper returns the columns that are listed in the SELECT statement.

```sql
SELECT Model, ModelScore, ModelEValue, DomainNumber, DomainScore, DomainEValue
  FROM hmmpfam_nickname
  WHERE HmmQSeq = 'MTMFWQQNVDHSDEQDKQAKGAAPTKRLNISFNV-KI-AVNVWVT'...';
```

**Query 2.**

When this SQL statement runs, the wrapper performs an hmmpfam search of hmmpfam_nickname that uses the indicated gene sequence. In addition, the wrapper passes the -T 0 option to the hmmpfam command. This option is from the list of fixed input columns for HMMER nicknames. The wrapper returns the three columns that are listed after SELECT.

```sql
SELECT Model, ModelScore, ModelEValue
  FROM hmmpfam_nickname
  WHERE HmmQSeq = 'MTMFWQQNVDHSDEQDKQAKGAAPTKRLNISFNV-KI-AVNVWVT'...'
    AND ModelScore > 0;
```
Query 3.

When this SQL statement runs, the wrapper performs an hmmpfam search of hmmpfam_nickname that uses the indicated gene sequence. In addition, the wrapper passes the -E 1 option to the hmmpfam command. This option is from the list of fixed input columns for HMMER nicknames. The wrapper returns the four columns that are listed after SELECT and sorts the result from highest to lowest by the DomainScore.

```sql
SELECT Model, DomainNumber, DomainScore, DomainEValue
FROM hmmpfam_nickname
WHERE HmmQSeq = 'MTMFQQVQDOHQSDQKDQIAKAPTKRLNISFDVNTKMTTTINQ...
AND ModelEValue < 1
ORDER BY DomainScore DESC;
```

Query 4.

When this SQL statement runs, the wrapper runs hmmsearch against the sequence file artemia, using the HMM specified by globin. The rows with a DomainScore greater than 50 are returned, because the wrapper passes the --domT 50 option to the hmmsearch command. The wrapper returns the columns specified after SELECT. Column values that are longer than 200 characters are truncated. Only the first 200 characters in these columns are returned.

```sql
CREATE WRAPPER hmmer_wrapper
  LIBRARY 'db2ishmmer.dll';

CREATE SERVER hmmsearch_server
  TYPE search VERSION 2.2
  WRAPPER hmmer_wrapper
  OPTIONS(NODE 'HMMERserv.MyCompany.com');

CREATE NICKNAME artemia_nickname
  FOR SERVER hmmsearch_server
  OPTIONS(DATASOURCE 'artemia', TIMEOUT '1');

SELECT Model, Sequence, substr(SequenceDescription,1,50)
  AS SequenceDescription, SequenceScore, SequenceEValue,
  SequenceHits, DomainNumber, SequenceFrom,
  SequenceTo, SequenceFromGlobal, SequenceToGlobal,
  HmmFrom, HmmTo, HmmFromGlobal, HmmToGlobal, DomainScore,
  DomainEValue,
  length(AlignmentConsensus) AS "length(AConsensus)",
  length(AlignmentMatch) AS "length(AMatch)",
  length(AlignmentSubSeq) AS "length(ASubSeq)",
  substr(AlignmentConsensus,1,200) AS AlignmentConsensus,
  substr(AlignmentMatch,1,200) AS AlignmentMatch,
  substr(AlignmentSubSeq,1,200) AS AlignmentSubSeq
FROM artemia_nickname
WHERE Model = 'globin'
AND DomainScore > 50;
```
Chapter 8. Configuring access to Informix data sources

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

Before you begin

- The Informix Client SDK software must be installed and configured on the server that will act as the federated server.
- Check the setup of the federated server.
- Check the federated parameter to ensure that federation is enabled.
- On AIX federated servers, the AIX Base Application Development Math Library must be installed. You can determine if the Library is installed by issuing the AIX command lslpp -l bos.adt.libm.

You can configure a federated server to access data that is stored in Informix data sources by using the DB2 Control Center or by issuing SQL statements on the DB2 command line. The DB2 Control Center includes a wizard to guide you through the steps that are necessary to configure the required federated objects.

Procedure

To add Informix data sources to a federated server:

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

Setting up and testing the Informix client configuration file

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

Before you begin

The Informix Client SDK software must be installed on the federated server.

The client configuration file specifies the location of each Informix database server and type of connection (protocol) for the database server.

The default location of the client configuration file depends on the operating system that is used by the federated server.

- On federated servers that run UNIX, the default location and name of the file is $INFORMIXDIR/etc/sqlhosts. The sqlhosts file is installed with the Informix client SDK.
On federated servers that run Windows, the default location of the sqlhosts registry is the local computer.

The format of sqlhosts is described in the Administrator’s Guide for Informix Dynamic Server.

Procedure

To set up and test the Informix client configuration file:

1. Configure the Informix Client SDK.
   - On federated servers that run UNIX, you can configure the Informix Client SDK by editing the sqlhosts file. You can also copy the sqlhosts file from another system that has Informix Connect or Informix Client SDK installed.
   - On federated servers that run Windows, you can configure the Informix Client SDK with the Informix Setnet32 utility. The Setnet32 utility sets up the sqlhosts registry.

2. Verify the location of the sqlhosts file or registry.
   - On federated servers that run UNIX, the sqlhosts file is located in the $INFORMIXDIR/etc/ directory.
   - On federated servers that run Windows, the sqlhosts information is kept in the following key in the Windows registry:
     HKEY_LOCAL_MACHINE\SOFTWARE\INFORMIX\SQLHOSTS

3. If you want to place the sqlhosts file or registry in a path other than the default search path, set the INFORMIXSQLHOSTS environment variable to specify the file location. Use one of the following options to set the INFORMIXSQLHOSTS environment variable.
   - On federated servers that run UNIX, set the INFORMIXSQLHOSTS environment variable to the fully-qualified name of the sqlhosts file.
   - On federated servers that run Windows, use the Setnet32 utility to set the INFORMIXSQLHOSTS environment variable to the name of the Windows computer that stores the registry.

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

Setting the Informix environment variables

The Informix environment variables must be set in the db2dj.ini file on the federated server.

There are required and optional environment variables for Informix data sources. If you installed the Informix client software before you installed the Informix wrapper, the required Informix environment variables are set in the db2dj.ini file.

You must set the environment variables using the steps in this task if you did not install the Informix client software before you installed the Informix wrapper or if you want to set any of the optional environment variables.

To set the Informix environment variables:

1. Use one of the following methods to set the Informix environment variables that you want to use:
Method | Step
--- | ---
To set the environment variables using the DB2 Control Center | Use the Federated Objects wizard in the DB2 Control Center. To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.

To set the environment variables automatically | Run the DB2 Setup program again and specify the Custom installation option. Follow the instructions in the wizard.

**Important:** Running the installation program again will set only the required environment variables. The optional environment variables must be set manually.

To set the environment variables manually | Edit the db2dj.ini file.

- On federated servers that run UNIX, the db2dj.ini file is located in the sqllib/cfg directory.
- On federated servers that run Windows, the db2dj.ini file is located in the %DB2PATH%\cfg directory.

The db2dj.ini file contains configuration information about the Informix client software that is installed on your federated server. If the file does not exist, you can create a file with the name db2dj.ini by using any text editor. In the db2dj.ini file, you must specify the fully qualified path for the environment variables; otherwise you will encounter errors. The following environment variables show what an entry in the db2dj.ini file on UNIX might look like:

```
INFORMIXDIR=/informix/csdk
INFORMIXSERVER=inf10
```

2. Set the Informix code page conversion environment variables (as necessary).

3. To ensure that the environment variables are set on the federated server, recycle the federated database instance.

Issue the following commands to recycle the federated database instance:

```
db2stop
db2start
```

**Informix environment variables**

There are required and optional environment variables for Informix data sources. These variables are set in the db2dj.ini file.

The valid environment variables for Informix are:

- INFORMIXDIR
- INFORMIXSERVER
- INFORMIXSQLHOSTS (optional)
- CLIENT_LOCALE (optional)
- DB_LOCALE (optional)
- DBNLS (optional)

The CLIENT_LOCALE, DB_LOCALE, and DBNLS environment variables are code page environment variables.
Variable descriptions

INFORMIXDIR
Specifies the directory path where the Informix Client SDK software is installed.

For example:
• On federated servers that run UNIX, set the path to:
  INFORMIXDIR=/informix/csdk
• On federated servers that run Windows, set the path to:
  INFORMIXDIR=C:\informix\csdk

INFORMIXSERVER
Identifies the name of the default Informix server. This setting must be a valid entry in the sqlhosts file (UNIX) or the SQLHOSTS registry key (Windows). To get a value for INFORMIXSERVER, read the sqlhosts file. Select one of the dbservername values. The dbservername is the first value in each entry in the sqlhosts file.

For example:
INFORMIXSERVER=inf10

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

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

• On federated servers that run UNIX, the default path is $INFORMIXDIR/etc/sqlhosts.
• On federated servers that run Windows, if the SQLHOSTS registry key does not reside on the local computer, then the value for the INFORMIXSQLHOSTS environment variable is the name of the Windows computer that stores the registry.

A UNIX example of setting this environment variable to another path is:
INFORMIXSQLHOSTS=/informix/csdk/etc/my_sqlhosts

Informix code page conversion
Each time that the Informix wrapper connects to an Informix data source, the wrapper determines which code page value to use for that connection. You can have the Informix wrapper set the code page value or you can designate a code page by setting the CLIENT_LOCALE environment variable.

The environment variables that specify Informix code page conversion are set in the db2dj.ini file on your federated server.

For Informix code page conversion, you can set the following optional environment variables:
• CLIENT_LOCALE
• DB_LOCALE
The Informix code page environment variables are:

**CLIENT_LOCALE**
Specifies the Informix locale that you want to use. Use this variable when you do not want the Informix wrapper to automatically determine the variable setting.

For example:

```
CLIENT_LOCALE=Informix_client_locale_value
```

- If the `CLIENT_LOCALE` variable is set in the db2dj.ini file on the federated server, then the wrapper uses the code page value in the db2dj.ini file.
- If the `CLIENT_LOCALE` variable is not set on the federated server, the wrapper determines the territory and the code page of the federated database. The wrapper sets the `CLIENT_LOCALE` variable to the closest matching Informix locale. If there is no matching Informix locale, the wrapper sets the `CLIENT_LOCALE` variable to the en_us.8859-1 locale for UNIX systems and to the en_us.CP1252 locale for Windows systems.

You can see the list of valid Informix locales by issuing the `glfiles` command on the Informix server.

Refer to the *Informix Guide to GLS Functionality* for more information about code page conversions.

**DB_LOCALE**
Specifies that the Informix database uses a different code page than your client locale. Use this variable when you want Informix to perform conversions between the two code pages. Set the `DB_LOCALE` environment variable to the name of the Informix database locale.

For example:

```
DB_LOCALE=Informix_db_locale_value
```

**DBNLS**
Specifies that Informix verifies that the `DB_LOCALE` setting matches the actual locale of the Informix database. Set this environment variable to 1.

For example:

```
DBNLS=1
```

**Force Informix to perform code page conversion**

The Informix database uses a different code page than your client locale and you want Informix to perform conversions between the two code pages. You need to:

1. Set Informix environment variable `DB_LOCALE` to the name of the Informix database locale. You set this variable in the db2dj.ini file on the federated server.
2. To verify that the `DB_LOCALE` setting matches the actual locale of the Informix database, set the Informix environment variable `DBNLS` to 1. You set this variable in the db2dj.ini file on the federated server.
Informix data that uses the Chinese code page GB 18030

To access data that uses the Chinese code page GB 18030, use the UTF-8 code page on your federated database and add the following setting to your db2dj.ini file, so that Informix correctly translates the GB 18030 data to unicode.

DB_LOCALE=zh_cn.GB18030-2000

Registering the Informix wrapper

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

Procedure

To register the Informix wrapper:

Issue the CREATE WRAPPER statement and specify the default name for the Informix wrapper.

For example:

CREATE WRAPPER INFORMIX;

Recommendation: Use the default wrapper name. The default wrapper name for Informix is INFORMIX. When you register the wrapper by using the default name, the federated server automatically uses the appropriate Informix wrapper library for the operating system that your federated server is running on.

If the default wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name that you choose. When you use a name that is different from the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement.

For example, to register a wrapper with the name informix_wrapper on a federated server that uses the AIX operating system, issue the following statement:

CREATE WRAPPER informix_wrapper LIBRARY 'libdb2informix.a';

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

Informix wrapper library files

The Informix wrapper library files are added to the federated server when you install federation.

When you install federation, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2informix.a, libdb2informixF.a, and libdb2informixU.a. The default wrapper library file is libdb2informix.a. The other wrapper library files are used internally by the default wrapper library.

If you decide not to use the default wrapper name when you register the wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.
The default directory paths and default wrapper library file names are listed in the following table.

### Table 66. Informix wrapper library locations and file names

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Library file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/install_path/lib32/</td>
<td>libdb2informix.a</td>
</tr>
<tr>
<td></td>
<td>/usr/opt/install_path/lib64/</td>
<td></td>
</tr>
<tr>
<td>HP-UX</td>
<td>/opt/IBM/db2/install_path/lib32</td>
<td>libdb2informix.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/install_path/lib64</td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/IBM/db2/install_path/lib32</td>
<td>libdb2informix.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/install_path/lib64</td>
<td></td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/IBM/db2/install_path/lib32</td>
<td>libdb2informix.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/install_path/lib64</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>%DB2PATH%\bin</td>
<td>db2informix.dll</td>
</tr>
</tbody>
</table>

- `install_path` is the directory path where federation is installed on UNIX or Linux.
- `%DB2PATH%` is the environment variable that is used to specify the directory path where federation is installed on Windows. The default Windows directory path is `C:\Program Files\IBM\SQLLIB`.

### Registering the server definitions for an Informix data source

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

#### Procedure

To register a server definition for an Informix data source:

1. Locate the node name in the Informix sqlhosts file or registry.
   
   Sample sqlhosts file:
   ```
   inf10an onsoctcp anaconda inmx10
   inf10bo onsoctcp boa ifmx10
   inf10py onsoctcp python ifmx10
   ```
   - The first value in each line is the `node_name`, such as `inf10an`.
   - The second value in each line is the `nettype`, or type of connection. In this example `onsoctcp` indicates this is a TCP/IP connection.
   - The third value in each line is the host name, such as `anaconda`, `boa`, and `python`.
   - The fourth value in each line is the service name, such as `inmx10`. The service name field depends on the `nettype` listed in the second value.
   
   For more information about the format of the sqlhosts file and the meaning of these fields, see the Informix manual *Administrators Guide for Informix Dynamic Server*.

2. Use one of the following methods to create the server definition.
   - Use the Federated Objects wizard in the DB2 Control Center. To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.
   - Issue the `CREATE SERVER` statement.

For example:
CREATE SERVER server_definition_name TYPE informix
    VERSION version_number WRAPPER INFORMIX
    OPTIONS (NODE 'node_name', DBNAME 'database_name');

Although the 'node_name' and 'database_name' variables are specified as options in the CREATE SERVER statement, these options are required for Informix data sources.

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

### CREATE SERVER statement - Examples for the Informix wrapper

Use the CREATE SERVER statement to register server definitions for the Informix wrapper. This topic includes a complete example with the required parameters, and examples with additional server options.

#### Complete example

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

```sql
CREATE SERVER asia TYPE informix VERSION 10 WRAPPER INFORMIX
    OPTIONS (NODE 'abc', DBNAME 'sales');
```

- `asia` A name that you assign to the Informix database server. Duplicate server definition names are not allowed.
- **TYPE informix**
  Specifies the type of data source server to which you are configuring access. For the Informix wrapper, the server type must be informix.
- **VERSION 10**
  The version of the Informix database server that you want to access.
- **WRAPPER INFORMIX**
  The wrapper name that you specified in the CREATE WRAPPER statement.
- **NODE 'abc'**
  The name of the node where the Informix database server resides. Obtain the node name from the sqlhosts file. This value is case sensitive.
  Although the name of the node is specified as an option in the CREATE SERVER statement, it is required for Informix data sources.
- **DBNAME 'sales'**
  The name of the Informix database that you want to access. This value is case sensitive.
  Although the name of the database is specified as an option in the CREATE SERVER statement, it is required for Informix data sources.

#### Additional server options

When you create a server definition, you can specify additional server options in the CREATE SERVER statement. The server options can be general server options and Informix-specific server options.
**FOLD_ID and FOLD_PW server options**

When the federated server connects to a data source, the federated server tries to connect using all possible combinations of uppercase and lowercase for the user ID and password, as well as the current case. The federated server might make up to nine connect attempts before successfully connecting to the data source server. These attempts can slow down connect times and might result in the user ID being locked out. You can prevent lock outs by specifying values for the FOLD_ID and FOLD_PW server options. You can set the FOLD_ID and FOLD_PW server options to ‘N’ (do not fold the user ID or password).

If you set the FOLD_ID and FOLD_PW server options to ‘N’, you must specify the user ID and password in the correct case. The advantage to setting these server options to ‘N’ is that when an invalid user ID or password is specified, the wrapper will not keep trying the various uppercase and lowercase combinations. These two server options can reduce the chance of exceeding the maximum number of failed login attempts and the ID getting locked out.

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

```sql
CREATE SERVER asia TYPE informix VERSION 10 WRAPPER INFORMIX
   OPTIONS (NODE 'abc', DBNAME 'sales', FOLD_ID 'N', FOLD_PW 'N');
```

**IUD_APP_SVPT_ENFORCE server option example**

The IUD_APP_SVPT_ENFORCE option specifies whether the federated server should enforce detecting or building of application savepoint statements. Informix does not support application savepoint statements. When set to ‘N’, the federated server will not roll back transactions when an error is encountered. Your application must handle the error recovery.

The IUD_APP_SVPT_ENFORCE server option must be set to ‘N’ to enable replication to or from Informix data sources. The following example shows an Informix server definition with the IUD_APP_SVPT_ENFORCE server option.

```sql
CREATE SERVER asia TYPE informix VERSION 10 WRAPPER INFORMIX
   OPTIONS (NODE 'abc', DBNAME 'sales', IUD_APP_SVPT_ENFORCE 'N');
```

**INFORMIX_DB_LOCALE and INFORMIX_CLIENT_LOCALE options**

The INFORMIX_DB_LOCALE option sets the database locale environment variable (DB_LOCALE) that is used for the connection between the federated server and the data source server. If the INFORMIX_DB_LOCALE option is not specified, the Informix DB_LOCALE environment variable is set to the value that is specified in the db2dj.ini file. If the db2dj.ini file does not specify the DB_LOCALE environment variable, the Informix DB_LOCALE environment variable is not set. A valid value is any valid Informix locale. This option is optional. The default setting is None.

The INFORMIX_CLIENT_LOCALE option sets the client locale environment variable (CLIENT_LOCALE) to be used for the connection between the federated server and the data source server. If the INFORMIX_CLIENT_LOCALE option is not specified, the Informix CLIENT_LOCALE environment variable is set to the value specified in the db2dj.ini file. If db2dj.ini does not specify CLIENT_LOCALE, then the Informix CLIENT_LOCALE environment variable is set to the Informix locale.
locale that most closely matches the code page and territory of the federated database. A valid value is any valid Informix locale. This option is optional. The default setting is None.

The following example shows an Informix server definition with the INFORMIX_DB_LOCALE and INFORMIX_CLIENT_LOCALE options.

CREATE SERVER asia TYPE informix VERSION 10 WRAPPER INFORMIX
  OPTIONS (NODE 'abc', DBNAME 'sales', INFORMIX_DB_LOCALE 'en_us.8859-1', INFORMIX_CLIENT_LOCALE 'en_us.CP1252');

Creating the user mappings for an Informix data source

When you attempt to access an Informix server, the federated server establishes a connection to the Informix server by using a user ID and password that are valid for that data source. You must define an association (a user mapping) between each federated server user ID and password and the corresponding data source user ID and password.

Create a user mapping for each user ID that will access the federated system to send distributed requests to the Informix data source.

Procedure

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

Issue a CREATE USER MAPPING statement.

For example:

CREATE USER MAPPING FOR local_userID SERVER server_definition_name
  OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password');

Although the REMOTE_AUTHID and REMOTE_PASSWORD variables are specified as options in the CREATE USER MAPPING statement, these options are required to access Informix data sources.

CREATE USER MAPPING statement - Examples for the Informix wrapper

Use the CREATE USER MAPPING statement to map a federated server user ID to an Informix server user ID and password. This topic includes a complete example with the required parameters, and an example that shows you how to use the DB2 special register USER with the CREATE USER MAPPING statement.

Complete example

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

CREATE USER MAPPING FOR VINCENT SERVER asia
  OPTIONS (REMOTE_AUTHID 'vinnie', REMOTE_PASSWORD 'close2call');

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

SERVER asia
  Specifies the server definition name that you registered in the CREATE SERVER statement for the Informix server.
REMOTE_AUTHID 'vinnie'
Specifies the user ID at the Informix database server to which you are mapping VINCENT. The value is case-sensitive, unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

Although the remote user ID is specified as an option in the CREATE USER MAPPING statement, it is required for Informix data sources.

REMOTE_PASSWORD 'close2call'
Specifies the password that is associated with 'vinnie'. The value is case-sensitive, unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

Although the remote password is specified as an option in the CREATE USER MAPPING statement, it is required for Informix data sources.

Special register example
You can use the DB2 special register USER to map the authorization ID of the person who is issuing the CREATE USER MAPPING statement to the data source authorization ID that is specified in the REMOTE_AUTHID user option.

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

```
CREATE USER MAPPING FOR USER SERVER asia
  OPTIONS (REMOTE_AUTHID 'vinnie', REMOTE_PASSWORD 'close2call');
```

Testing the connection to the Informix server
Test the connection to the Informix data source server to determine if the federated server is properly configured to access Informix data sources.

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

Procedure
To test the connection to the Informix server:

Open a pass-through session and issue a SELECT statement on the Informix system tables. If the SELECT statement returns a count, your server definition and your user mapping are set up properly.

For example:
```
SET PASSTHRU server_definition_name
SELECT count(*) FROM informix.systables
SET PASSTHRU RESET
```

If the SELECT statement returns an error, you should troubleshoot the connection errors.

Troubleshooting data source connection errors
A test connection to the data source server might return an error for several reasons. There are actions that you can take to determine why the error occurred.
Symptom

An error is returned when you attempt to connect to the data source.

Cause

There are several possible causes for a connection problem.

Resolving the problem

To troubleshoot data source connection errors, check the following items for problems:

- Verify that the data source is available.
- If applicable, ensure that the data source server is configured for incoming connections.
- Ensure that your user mapping settings for the REMOTE_AUTHID and REMOTE_PASSWORD options are valid for the connections to the data source. Alter the user mapping, or create another user mapping as necessary.
- If applicable, ensure that the data source client software on the federated server is installed and configured correctly to connect to the data source.
- For ODBC data sources, ensure that the ODBC driver on the federated server is installed and configured correctly to connect to the ODBC data source server. On federated servers that run Windows, use the ODBC Data Source Administrator tool to check the driver. On federated servers that run UNIX, consult the ODBC client vendor’s documentation.
- Verify that the settings for the variables set on the federated server are correct for the data source. These variables include the system environment variables, the variables in the db2dj.ini file, and the DB2 Profile Registry (db2set) variables.
- Check your server definition. If necessary, drop the server definition and create it again.

Performance tuning for the Informix wrapper

You can use the FOLD_ID and FOLD_PW server options to improve connectivity between the federated server and Informix data sources.

When the federated server connects to a data source, the server tries to connect using all possible combinations of uppercase and lowercase for the user ID and password. The server might make up to nine connect attempts before successfully connecting to the data source server. These attempts can slow down connect times and might result in the user ID getting locked out.

You can improve performance by specifying the values for the FOLD_ID and FOLD_PW server options.

- If all your Informix user IDs and passwords are in lowercase, setting the FOLD_ID and FOLD_PW server options with the value ‘L’ can improve your connect time.
  For example:
  ```sql
  ALTER SERVER TYPE INFORMIX
  OPTIONS (ADD FOLD_ID 'L');
  ALTER SERVER TYPE INFORMIX
  OPTIONS (ADD FOLD_PW 'L');
  ```

- The federated server attempts each combination of uppercase and lowercase values for the user ID and password. You can reduce the chance of the
maximum number of failed login attempts being exceeded by setting these options to 'N' (do not fold the user ID and the password). If you establish these settings, then you need to always specify the user ID and password in the correct case. If an invalid user ID and password are specified, the wrapper will not keep trying the various combinations.

For example:

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

---

**Registering nicknames for Informix tables, views, and synonyms**

For each Informix server definition that you register, you must register a nickname for each table, view, or synonym that you want to access. Use these nicknames, instead of the names of the data source objects, when you query the Informix servers.

**Before you begin**

Update the statistics at the Informix data source before you register a nickname. The federated database relies on the data source catalog statistics to optimize query processing. You can use the Informix UPDATE STATISTICS command, which is equivalent to the DB2 RUNSTATS command to update the data source statistics.

**Procedure**

To register a nickname for an Informix table, view, or synonym:

Issue the CREATE NICKNAME statement. Nicknames can be up to 128 bytes in length.

For example:

```
CREATE NICKNAME nickname FOR server_definition_name."remote_schema"."remote.table";
```

When you create the nickname, the federated server queries the data source catalog using the nickname. This query tests the connection to the data source table, view, or synonym. If the connection does not work, you will receive an error message.

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

---

**CREATE NICKNAME statement - Examples for the Informix wrapper**

Use the CREATE NICKNAME statement to register a nickname for an Informix table, view, or synonym that you want to access. This topic includes a complete example with the required parameters.

**Complete example**

```
CREATE NICKNAME JPSALES FOR asia."vinnie"."japan";
```

**JPSALES**

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

---

Chapter 8. Configuring access to Informix data sources  181
**Important:** The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authorization ID of the user who registers the nickname.

```
asia."vinnie"."japan"
```

A three-part identifier for the remote object:

- *asia* is the server definition name that you assigned to the Informix database server in the CREATE SERVER statement.
- *vinnie* is the name of the owner to which the table, view, or synonym belongs unless the database is ANSI-compliant. In an ANSI-compliant database, it is the schema name.
- *japan* is the name of the remote table, view, or synonym that you want to access.

The federated server folds the names of the Informix schemas and tables to uppercase unless you enclose the names in quotation marks.
Chapter 9. Configuring access to Microsoft SQL Server data sources

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

**Before you begin**

- The ODBC driver must be installed and configured on the federated server.
- Federation must be installed on a server that acts as the federated server.
- Check the setup of the federated server.
- Check the federated parameter to ensure that federation is enabled.

You can configure the federated server to access Microsoft SQL Server data sources by using the DB2 Control Center or by issuing SQL statements on the DB2 command line.

**Procedure**

To configure access to Microsoft SQL Server data sources:

1. Use one of the following methods to prepare the federated server and federated database depending on your operating system.
   - [Prepare the federated server and federated database (Windows)].
   - [Prepare the federated server and federated database (UNIX)].
2. Set the environment variables for the Microsoft SQL Server wrapper.
3. Register the wrapper.
4. Register the server definition.
5. Create the user mappings.
6. Test the connection to the Microsoft SQL Server remote server.
7. Register nicknames for Microsoft SQL Server tables and views.

**Preparing the federated server to access Microsoft SQL Server data sources (Windows)**

On federated servers that run Windows, the federated server must be able to access Microsoft SQL Server data sources. To prepare the federated server, you must verify the settings in the ODBC System DSN and test the connection to Microsoft SQL Server data sources.

**Procedure**

To prepare the federated server to access Microsoft SQL Server data sources:

1. Verify that the ODBC System DSN is set to connect to the Microsoft SQL Server data source. In the Control Panel, locate the existing DSN entry for the Microsoft SQL Server remote server or create a DSN entry.
   - The DSN entry for the Microsoft SQL Server remote server is the value that you will use for the NODE server option when you register the server definition in the federated database.
2. Use one of the following methods to test the connection to the Microsoft SQL Server data source:
   - Select **Configure** from the ODBC Data Source Administrator window.
   - Use the Microsoft SQL Server query tool.

After you complete this task, you can set the environment variables.

---

### Preparing the federated server to access Microsoft SQL Server data sources (Linux, UNIX)

On federated servers that run Linux or UNIX, the federated server must be able to access Microsoft SQL Server data sources. To prepare the federated server, you must verify the settings in the odbc.ini file, create symbolic links, and test the connection to Microsoft SQL Server data sources.

**Procedure**

To prepare the federated server to access Microsoft SQL Server data sources:

1. Verify that the odbc.ini file is updated on the federated server. If the odbc.ini file does not exist on the federated server, you can create it in a text editor. Consult the documentation from the ODBC client vendor for information about the odbc.ini file.

   **Remember:** Place the odbc.ini file or a copy of this file in the home directory of the DB2 instance owner to ensure that it can be accessed if the instance owner is not the root user.

2. Verify that the path to the odbc.ini is in the ODBCINI environment variable. From an operating system command prompt, issue the following command:

```
export ODBCINI=$HOME/.odbc.ini
```
3. Create the appropriate symbolic links:

<table>
<thead>
<tr>
<th>Federated server operating system</th>
<th>Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>Create the following symbolic links:</td>
</tr>
<tr>
<td></td>
<td><code>ln -s $DJX_ODBC_LIBRARY_PATH/..</code></td>
</tr>
<tr>
<td></td>
<td><code>/locale/usr/local/locale</code></td>
</tr>
<tr>
<td></td>
<td><code>ln -s $DJX_ODBC_LIBRARY_PATH/libodbcinst.so/</code></td>
</tr>
<tr>
<td></td>
<td><code>usr/lib/libodbcinst.so</code></td>
</tr>
<tr>
<td></td>
<td>If you are using the DataDirect Technologies Connect for the ODBC driver, verify the library name and create the symbolic link. The name of the library varies depending on the version of the driver and whether you are using a 32-bit or a 64-bit driver.</td>
</tr>
<tr>
<td></td>
<td>For example, if you’re using DataDirect Version 4.2, you would create the following link:</td>
</tr>
<tr>
<td></td>
<td><code>ln -s $DJX_ODBC_LIBRARY_PATH/libivicu19.so/</code></td>
</tr>
<tr>
<td></td>
<td><code>usr/lib/libivicu19.so</code></td>
</tr>
<tr>
<td></td>
<td>If you’re using DataDirect Version 5.0, you would create the following link:</td>
</tr>
<tr>
<td></td>
<td><code>ln -s $DJX_ODBC_LIBRARY_PATH/libivicu20.so/</code></td>
</tr>
<tr>
<td></td>
<td><code>usr/lib/libivicu20.so</code></td>
</tr>
<tr>
<td></td>
<td>If you use DataDirect and do not include the symbolic link, CREATE WRAPPER MSSQLODBC3 might fail with the following error message:</td>
</tr>
<tr>
<td></td>
<td><code>SQL10013N The specified library name could not be loaded.</code></td>
</tr>
<tr>
<td>Solaris</td>
<td>Create the following symbolic link:</td>
</tr>
<tr>
<td></td>
<td><code>ln -s $DJX_ODBC_LIBRARY_PATH/..</code></td>
</tr>
<tr>
<td></td>
<td><code>/locale $HOME/sqllib/locale</code></td>
</tr>
<tr>
<td></td>
<td><code>$HOME</code> is the home directory of the DB2 instance owner.</td>
</tr>
</tbody>
</table>

4. Run the `/opt/odbc/odbc.sh` script. This script sets up several operating system specific environment variables.

5. Test the connection from the federated server to the Microsoft SQL server data source by using the DataDirect Connect ODBC demoodbc utility. The demoodbc utility is located in the `/demo` subdirectory of the DataDirect Connect ODBC libraries.

After you complete this task, you can set the environment variables.
Setting the Microsoft SQL Server environment variables

The Microsoft SQL Server environment variables must be set in the db2dj.ini file on the federated server.

Restrictions

Review the restrictions for the db2dj.ini file before you begin this task.

The db2dj.ini file contains configuration information about the Microsoft SQL Server ODBC driver that is installed on your federated server.

There are required and optional environment variables for Microsoft SQL Server data sources.

If you installed the Microsoft SQL Server client software before you installed the Microsoft SQL Server wrapper, the required Microsoft SQL Server environment variables are set in the db2dj.ini file.

You must set the environment variables by using the steps in this task if you did not install the Microsoft SQL Server client software before you installed the Microsoft SQL Server wrapper or if you want to set any of the optional environment variables.

Procedure

To set the Microsoft SQL Server environment variables:

1. Use one of the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click <strong>Create Federated Objects</strong>.</td>
</tr>
<tr>
<td>Automatically set the environment variables.</td>
<td>Run the WebSphere Federation Server installation wizard. Follow the instructions in the wizard. <strong>Important:</strong> Set the required environment variables by running the installation wizard. The optional environment variables must be set manually.</td>
</tr>
<tr>
<td>Manually set the environment variables.</td>
<td>Edit the db2dj.ini file.</td>
</tr>
<tr>
<td></td>
<td>• On federated servers that run UNIX, this file is located in the sqllib/cfg directory.</td>
</tr>
<tr>
<td></td>
<td>• On federated servers that run Windows, this file is located in the %DB2PATH%\cfg directory.</td>
</tr>
<tr>
<td></td>
<td>If the file does not exist, you can create a file with the name db2dj.ini by using any text editor. In the db2dj.ini file, you must specify the fully qualified path in the value of the environment variables; otherwise you will encounter errors.</td>
</tr>
</tbody>
</table>
|                                                 | * **Example:** DJX_ODBC_LIBRARY_PATH=/opt/odbc/lib
|                                                 | * **Example:** DJX_ODBCINI=/opt/odbc/.odbc.ini                      |

2. To ensure that the environment variables are set on the federated server, recycle the DB2 instance with these commands:
After you complete this task, you can register the wrapper.

**Microsoft SQL Server environment variables**

There are required and optional environment variables for Microsoft SQL Server data sources. These variables are set in the db2dj.ini file.

The following environment variables are valid for Microsoft SQL Server:
- DJX_ODBC_LIBRARY_PATH
- ODBCINI
- LD_LIBRARY_PATH (Solaris only)

**Variable descriptions**

**DJX_ODBC_LIBRARY_PATH**
Specifies the directory path to the ODBC library files. This variable must also be specified on federated servers that run Solaris.

For example:
```
DJX_ODBC_LIBRARY_PATH=ODBC_driver_directory/lib
```

*ODBC_driver_directory* is the directory path where the ODBC driver is installed.

**ODBCINI**
Specifies the directory path where your ODBC configuration file (odbc.ini) is located.

For example:
```
ODBCINI=/home/db2inst1/.odbc.ini
```

Do not set the ODBCINI environment variable as a system variable.

**LD_LIBRARY_PATH (Solaris only)**
On federated servers that run Solaris, specifies the directory path to the ODBC library files.

For example:
```
LD_LIBRARY_PATH=ODBC_driver_directory/lib
```

---

**Registering the Microsoft SQL Server wrapper**

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

**Procedure**

To register the Microsoft SQL Server wrapper:
Use one of the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click <strong>Create Federated Objects</strong>.</td>
</tr>
</tbody>
</table>
| Issue the CREATE WRAPPER statement and specify the default name for the Microsoft SQL Server wrapper. | For example:  
CREATE WRAPPER MSSQL0DBC3;  
**Remember:** When you register the wrapper by using the default name, MSSQL0DBC3, the federated server automatically uses the appropriate Microsoft SQL Server wrapper library for the operating system that your federated server is running on.  
If the default wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name that you choose. When you use a name that is different from the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement.  
For example, to register a wrapper with the name sqlserver_wrapper on a federated server that uses AIX, issue the following statement:  
CREATE WRAPPER sqlserver_wrapper LIBRARY 'libdb2mssql3.a';  
The wrapper library file that you specify depends on the operating system of the federated server. |

After you complete this task, you can register the server definition.

**Microsoft SQL Server wrapper library files**

The Microsoft SQL Server wrapper library files are added to the federated server when you install the wrapper.

When you install the Microsoft SQL Server wrapper, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2mssql3.a, libdb2mssql3Fa, and libdb2mssql3U.a. The default wrapper library file is libdb2mssql3.a. The other wrapper library files are used internally by the Microsoft SQL Server wrapper.

If you do not use the default wrapper name when you register a wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

The default directory paths and default wrapper library file names are listed in the following table.
Table 67. Microsoft SQL Server client library locations and file names

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Library file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/install_path/lib32/</td>
<td>libdb2mssql3.a</td>
</tr>
<tr>
<td></td>
<td>/usr/opt/install_path/lib64/</td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/IBM/db2/install_path/lib32</td>
<td>libdb2mssql3.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/install_path/lib64</td>
<td></td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/IBM/db2/install_path/lib32</td>
<td>libdb2mssql3.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/install_path/lib64</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>%DB2PATH%\bin</td>
<td>db2mssql3.dll</td>
</tr>
</tbody>
</table>

install_path is the directory path where the federated server is installed on UNIX or Linux.

Registering the server definitions for a Microsoft SQL Server data source

You must register each Microsoft SQL Server remote server that you want to access in the federated database.

Procedure

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

1. Locate the node name for the Microsoft SQL Server.
   - On federated servers that run Windows, the node name is the System DSN name that you specified for the Microsoft SQL Server remote server that you are accessing.
   - On federated servers that run UNIX, the node name is defined in the .odbc.ini file.

At the top of the .odbc.ini file, there is a section labeled ODBC Data Sources, which lists the nodes. Each of the nodes has a section in the .odbc.ini file that describes the node.

The following example is a .odbc.ini file on AIX. The node names are [rawilson] and [medusa].

[ODBC Data Sources]
rawilson=MS SQL Server 2000
medusa=MS SQL Server 2000
[rawilson]
Driver=/opt/odbc/lib/ddmss20.so
Description=MS SQL Server Driver for AIX
Address=9.112.39.1433
[medusa]
Driver=/opt/odbc/lib/ddmss20.so
Description=MS SQL Server Driver for AIX
Address=9.112.98.123,1433
[ODBC]
InstallDir=/opt/odbc

2. Use one of the following methods to create the server definition.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| Issue the CREATE SERVER statement | For example:  
CREATE SERVER server_definition_name  
TYPE MSSQLSERVER  
VERSION version_number  
WRAPPER wrapper_name  
OPTIONS (NODE 'node_name',  
DBNAME 'database_name');  
Although the 'node_name' and 'db_name' variables are specified as options in the CREATE SERVER statement, these options are required for Microsoft SQL Server data sources.  
After the server definition is registered, use the ALTER SERVER statement to add or drop server options. |

After you complete this task, you can create user mappings.

**Creating the user mappings for a Microsoft SQL Server data source**

When you attempt to access an Microsoft SQL Server remote server, the federated server establishes a connection to the Microsoft SQL Server remote server by using a user ID and password that are valid for that data source.

**Procedure**

To map a local authorization ID to a remote Microsoft SQL Server user ID and password:

Issue a CREATE USER MAPPING statement.

For example:

CREATE USER MAPPING FOR local_userID SERVER server_definition_name  
OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password');

After you complete this task, you can test the connection to the Microsoft SQL Server tables and views.

**Testing the connection to the Microsoft SQL Server remote server**

Test the connection to the Microsoft SQL Server remote server to determine if the federated server is properly configured to access Microsoft SQL Server data sources.

You can test the connection to the Microsoft SQL Server remote server by using the server definition and user mappings that you defined.

**Procedure**

To test the connection to the Microsoft SQL Server remote server:
Open a pass-through session and issue a SELECT statement on the Microsoft SQL Server system tables. If the SELECT statement returns a count, your server definition and your user mapping are set up properly.

For example:

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

After you complete this task, you can register nicknames.

### Registering nicknames for Microsoft SQL Server tables and views

For each Microsoft SQL Server remote server definition that you register, you must register a nickname for each table or view that you want to access. Use these nicknames, instead of the names of the data source objects, when you query the Microsoft SQL Server remote servers.

#### Before you begin

To ensure that the federated database has current and complete statistics, execute the Microsoft SQL Server sp_createstats stored procedure and the Microsoft SQL Server CREATE STATISTICS command from the Microsoft SQL Server database before creating the nickname.

The sp_createstats stored procedure gathers statistics on all of the default columns in a table in an Microsoft SQL Server data source, but does not gather statistics for columns that appear first within an index. To ensure that the federated database has complete statistics on the Microsoft SQL Server table, you also must use the Microsoft SQL Server CREATE STATISTICS command to gather statistics for each column that appears first in an index.

When you use the CREATE STATISTICS command from the Microsoft SQL Server database, you must give the statistic the same name for the column on which the statistics are being collected. By giving the statistic the same name as the column, you ensure that when you register the nickname with the CREATE NICKNAME statement, the federated database reads the statistics collected by the Microsoft SQL Server CREATE STATISTICS command.

#### Procedure

To register a nickname for a Microsoft SQL Server table or view:

Use one of the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Control Center.</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects. Follow the steps in the wizard.</td>
</tr>
</tbody>
</table>
| Issue the CREATE NICKNAME statement. | For example:  
CREATE NICKNAME nickname  
FOR server_definition_name."remote_schema"."remote.table"; |
When you create the nickname, the federated server queries the data source catalog using the nickname. This query tests the connection to the data source table, view, or synonym. If the connection does not work, you receive an error message.

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

**CREATE NICKNAME statement - Examples for the Microsoft SQL Server wrapper**

Use the CREATE NICKNAME statement to register a nickname for a Microsoft SQL Server table or view that you want to access. This topic provides a complete example with the required parameters.

The following example shows how to register a nickname for a Microsoft SQL Server table or view using the CREATE NICKNAME statement.

```
CREATE NICKNAME cust_africa FOR sqlserver."vinnie"."egypt"
```

`cust_africa`

A unique nickname that is used to identify the Microsoft SQL Server table or view.

**Important:** The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authorization ID of the user who registers the nickname.

`sqlserver."vinnie"."egypt"`

A three-part identifier for the remote object:

- `sqlserver` is the server definition name that you assigned to the Microsoft SQL Server remote server in the CREATE SERVER statement.
- `vinnie` is the user ID of the owner to which the table or view belongs.
- `egypt` is the name of the remote table or view that you want to access.

The federated server folds the names of the Microsoft SQL Server schemas and tables to uppercase unless you enclose the names in quotation marks.

---

**Using ODBC tracing information to troubleshoot connections to Microsoft SQL Server data sources**

If you experience problems connecting to the data source, you can obtain ODBC tracing information to analyze and resolve the problems.

**Symptom**

However, activating a trace does impact system performance. You should turn off tracing after you have resolved the connectivity problems.

If you are unable to connect to the data source with the Microsoft SQL Server wrapper, running a trace might help you diagnose the problem.

**Cause**

The cause of the problem might be an error in the wrapper configuration.
Diagnosing the problem

To diagnose the problem on a federated server running Windows:
1. In the Control Panel, open the Administrative Tools folder.
2. Click Data Sources (ODBC) to open the ODBC Data Source Administrator window.
3. Click the Tracing tab.
4. Click Start Tracing Now to start the trace utility.

On a federated server running UNIX:
   For example, if you use the DataDirect ODBC 3.x driver, find the example of the odbc.ini file in the client directory. The odbc.ini file contains a sample of the settings that are necessary to activate the trace files:
   ```
   [ODBC]
   Trace=1
   TraceFile=/home/user1/trace_dir/filename.xxx
   TraceDll=/ODBC_driver_directory/odbctrac.so
   InstallDir=/opt/odbc
   ```
   To turn tracing on, set the first line to Trace=1. To turn tracing off, set the first line to Trace=0. The value of the TraceFile setting is the path and file name that the federated database instance has write access to.

Resolving the problem

Check the trace log file for problems.

On Windows, open the ODBC Data Source Administrator and click the Tracing tab. The path to the trace log file is shown in the Log File Path field.

On UNIX, open the odbc.ini file. The path to the trace log file is indicated by the TraceFile setting.
Chapter 10. Configuring access to ODBC data sources

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

Before you begin

- The ODBC driver must be installed and configured on the server that acts as the federated server.
- Federation must be installed on the server that acts as the federated server.
- Check the setup of the federated server.
- Check the federated parameter to ensure that federation is enabled.
- The proper setup of the system environment variables, db2dj.ini file variables, and DB2 Profile Registry (db2set) variables. Check the documentation provided by the ODBC data source for the variables that are required for your ODBC client. The LIBPATH environment variable might be required.

Restrictions

- The ODBC wrapper cannot be used to access any DB2 family data sources. Use the DRDA wrapper to access DB2 family data sources.
- The ODBC wrapper does not support the following functions and statements:
  - LOCK TABLE statements on nicknames
  - Features deprecated in ODBC 3.x
  - X/Open or SQL/CLI drivers
  - Stored procedure nicknames
  - Statement-level atomicity enforcement using remote savepoint statements
  - WITH HOLD cursors
- For data sources that do not support positioned update and delete operations, positioned UPDATE and DELETE statements and certain searched UPDATE and DELETE statements on a nickname will fail if a unique index on non-nullable columns does not exist on the nickname or its corresponding remote table. The error SQL30090 with the reason code 21 is returned when these statements fail.
- The ODBC wrapper does not support INSERT, UPDATE, or DELETE statements against data sources that restrict the number of active statements for each connection. Consult the documentation for your data source to determine if the data source restricts the number of active statements for each connection. One of the ODBC data sources that this restriction applies to is IBM Red Brick™ Warehouse.
- The ODBC wrapper does not support operations on tables that contain columns with data types that use driver-specific SQL data type indicators. The type of operations that are not supported included the CREATE NICKNAME and SELECT statements in the pass-through mode. The ODBC wrapper supports only the SQL data type indicators that are defined by the ODBC standard in the Microsoft ODBC Programmer’s Reference.

About this task

You can configure the federated server to access ODBC data sources by using the DB2 Control Center or by issuing SQL statements on the DB2 command line. The
DB2 Control Center includes a wizard to guide you through the steps that are required to configure the required federated objects.

The data sources that are accessed through the ODBC API are referred to in this text as ODBC data sources.

Depending on your needs, you can access Excel data using the ODBC wrapper instead of using the Excel wrapper. The specific steps to configure the ODBC wrapper to access Excel data are documented in a separate topic.

**Procedure**

To add ODBC data sources to a federated server:

1. Use one of the following methods to prepare the federated server and federated database depending on your operating system:
   - Prepare the federated server to access data sources through ODBC (Windows)
   - Prepare the federated server to access data sources through ODBC (Linux, UNIX)

2. Register the ODBC wrapper
3. Register the server definitions for an ODBC data source
4. Create a user mapping for an ODBC data source
5. Test the connection to the ODBC data source server
6. Register nicknames for ODBC data source tables and views

---

### Preparing the federated server to access data sources through ODBC (Windows)

On federated servers that run Windows, the federated server must be able to access ODBC data sources. To prepare the federated server, you must verify the settings in the ODBC System DSN.

**Procedure**

To prepare the federated server to access data sources through ODBC:

Verify that the ODBC 3.x driver has been installed and configured on the federated server.

The node name for the ODBC data source must be defined in the System DSN. See the ODBC driver documentation for the installation and configuration procedures.

If you used the Microsoft ODBC Data Source Administrator window to configure the DSN, you can check this setting in the Control Panel. Ensure that ODBC data source is registered as a System DSN. Otherwise the federated server might not be able to find the DSN.

After you complete this task, you can register the wrapper.
Preparing the federated server to access data sources through ODBC
(Linux, UNIX)

On federated servers that run Linux or UNIX, the federated server must be able to access ODBC data sources. To prepare the federated server, you must verify the settings in the odbc.ini file, create symbolic links, and test the connection to ODBC data sources.

Procedure

To prepare the federated server to access data sources through ODBC:

1. Verify that the odbc.ini file has been updated on the federated server. If the file does not exist, you can create it in a text editor. Consult the documentation from the ODBC client vendor for information about the odbc.ini file.

2. Configure the ODBC client.
   Consult the documentation from the ODBC client vendor for instructions on how to configure the ODBC client.

3. If the client is DataDirect ODBC or RedBrick, verify that the appropriate symbolic links are created. In the following symbolic links, ODBC_CLIENT_DIR is the directory where the ODBC client is installed.

<table>
<thead>
<tr>
<th>Federated server operating system</th>
<th>Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>Create the following symbolic links:</td>
</tr>
<tr>
<td></td>
<td><code>ln -s $ODBC_CLIENT_DIR/../locale /usr/local/locale</code></td>
</tr>
<tr>
<td></td>
<td><code>ln -s $ODBC_CLIENT_DIR/libodbcinst.so /usr/lib/libodbcinst.so</code></td>
</tr>
<tr>
<td></td>
<td>If you are using the DataDirect Technologies Connect for ODBC 4.2 driver, you must also create the following symbolic link:</td>
</tr>
<tr>
<td></td>
<td><code>ln -s $ODBC_CLIENT_DIR/libivicu19.so /usr/lib/libivicu19.so</code></td>
</tr>
<tr>
<td>Solaris</td>
<td>Create the following symbolic link:</td>
</tr>
<tr>
<td></td>
<td><code>ln -s $ODBC_CLIENT_DIR/../locale $HOME/sql/lib/locale</code></td>
</tr>
<tr>
<td></td>
<td><code>$HOME</code> is the home directory of the DB2 instance owner.</td>
</tr>
</tbody>
</table>

4. If the client is DataDirect ODBC you can test the connection from the federated server to the data source by using the DataDirect Connect ODBC demoodbc tool.
   a. Run the `/opt/odbc/odbc.sh` script. This script sets up several operating system specific environment variables.
   b. Test the connection to the ODBC data source by using the DataDirect Connect ODBC demoodbc tool. The demoodbc tool is located in the `/demo` subdirectory of the Connect ODBC libraries.

After you complete this task, you can register the wrapper.

Registering the ODBC wrapper

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

Procedure
To register the ODBC wrapper:

Use one of the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click <strong>Create Federated Objects</strong>.</td>
</tr>
</tbody>
</table>
| Issue the CREATE WRAPPER statement and specify the default name for the ODBC wrapper. | For example:  
CREATE WRAPPER ODBC;  

**Remember:** When you register the wrapper by using the default name, ODBC, the federated server automatically uses the appropriate ODBC wrapper library for the operating system that your federated server is running on.  
You must specify the MODULE wrapper option on federated servers that run Linux or UNIX. The MODULE wrapper option specifies the full path of the library that contains the ODBC Driver Manager.  
For AIX and Solaris, if you are configuring access to WebSphere Classic Federation Server on z/OS data sources, you must specify the DB2_FENCED and DB2_SOURCE_CLIENT_MODE options as shown in the following example.  
CREATE WRAPPER ODBC  
LIBRARY 'libdb2rcodbc.a'  
OPTIONS (DB2_FENCED 'Y', DB2_SOURCE_CLIENT_MODE '32BIT', MODULE '/opt/IBM/DB2IIClassic82/cli/lib/cacsqlcli.so') |
| Issue the CREATE WRAPPER statement and specify an alternative name for the ODBC wrapper. | If the default wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name that you choose. When you use a name that is different from the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement.  
For example, to register a wrapper with the name odbc_wrapper on a federated server that uses AIX, issue the following statement:  
CREATE WRAPPER odbc_wrapper  
LIBRARY 'libdb2rcodbc.a';  
The wrapper library file that you specify depends on the operating system of the federated server. |

After you complete this task, you can register the server definitions.

**ODBC wrapper library files**

The ODBC wrapper library files are added to the federated server when you install the wrapper.

When you install the ODBC wrapper, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files added to the directory path are libdb2rcodbc.a, libdb2rcodbcF.a, and libdb2rcodbcU.a. The default wrapper library file is libdb2rcodbc.a. The other wrapper library files are used internally by the ODBC wrapper.
If you do not use the default wrapper name when you register a wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

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

Table 68. ODBC client library locations and file names

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Wrapper library file</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/install_path/</td>
<td>libdb2rcodbc.a</td>
</tr>
<tr>
<td></td>
<td>lib32/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/usr/opt/install_path/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>lib64/</td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/IBM/db2/install_path/</td>
<td>libdb2rcodbc.so</td>
</tr>
<tr>
<td></td>
<td>lib32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/install_path/</td>
<td>libdb2rcodbc.so</td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/IBM/db2/install_path/</td>
<td>libdb2rcodbc.so</td>
</tr>
<tr>
<td></td>
<td>lib32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/install_path/</td>
<td>libdb2rcodbc.so</td>
</tr>
<tr>
<td></td>
<td>lib64</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>%DB2PATH%\bin</td>
<td>db2rcodbc.dll</td>
</tr>
</tbody>
</table>

*install_path* is the directory path where federation is installed on UNIX or Linux.

**CREATE WRAPPER statement - Examples for the ODBC wrapper**

Use the CREATE WRAPPER statement to register the ODBC wrapper. This topic provides examples for Linux, UNIX and Windows.

In the following examples, *odbc_wrapper* is the name that you assign to the wrapper that you are registering in the federated database.

**Linux and Solaris federated server**

The following example shows you how to register a wrapper with the default name on a federated server that runs Linux or Solaris:

```sql
CREATE WRAPPER odbc OPTIONS (MODULE '/opt/lib/odbc.so');
```

You must specify the MODULE wrapper option on federated servers that run Linux or UNIX. The MODULE wrapper option specifies the full path of the library that contains the ODBC Driver Manager.

The following example shows you how to register a wrapper with an alternative name on a federated server that runs Linux or Solaris:

```sql
CREATE WRAPPER odbc_wrapper LIBRARY 'libdb2rcodbc.so'
    OPTIONS (MODULE '/opt/lib/odbc.so');
```

**AIX federated server**

The following example shows you how to register a wrapper with the default name on a federated server that runs AIX:

```sql
CREATE WRAPPER odbc
    OPTIONS ('/usr/lib/odbc.a');
```

You must specify the MODULE wrapper option on federated servers that run UNIX. The MODULE wrapper option specifies the full path of the library that contains the ODBC Driver Manager.
The following example shows you how to register a wrapper with an alternative name on a federated server that runs AIX:
CREATE WRAPPER odbc_wrapper LIBRARY 'libdb2rcodbc.a'
   OPTIONS (MODULE '/usr/lib/odbc.a');

**WebSphere Classic Federation Server for z/OS (AIX)**

The following example shows you how to register an ODBC wrapper by issuing the CREATE WRAPPER statement on an AIX operating system. On AIX and Solaris, the DB2_FENCED and DB2_SOURCE_CLIENT_MODE options must be specified as shown in the following example.
CREATE WRAPPER odbc
   OPTIONS (DB2_FENCED 'Y', DB2_SOURCE_CLIENT_MODE '32BIT',
            MODULE '/opt/IBM/DB2IIClassic82/cli/lib/cacsqlcli.so');

You must specify the MODULE wrapper option on federated servers that run UNIX. The MODULE wrapper option specifies the full path of the library that contains the ODBC Driver Manager.

The following example shows you how to register a wrapper for accessing WebSphere Classic Federation Server for z/OS data sources with an alternative name:
CREATE WRAPPER odbc_wrapper LIBRARY 'libdb2rcodbc.a'
   OPTIONS (DB2_FENCED 'Y', DB2_SOURCE_CLIENT_MODE '32BIT',
            MODULE '/opt/IBM/DB2IIClassic82/cli/lib/cacsqlcli.so');

**Windows federated server**

The following example shows you how to register a wrapper with the default name on a federated server that runs Windows:
CREATE WRAPPER odbc;

The following example shows you how to register a wrapper with an alternative name on a federated server that runs Windows:
CREATE WRAPPER odbc_wrapper LIBRARY 'db2rcodbc.dll';

**Registering the server definitions for an ODBC data source**

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

**Procedure**

To register a server definition for an ODBC data source:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Issue the CREATE SERVER statement. | For example:  
CREATE SERVER server_definition_name TYPE data_source_type  
    VERSION version_number WRAPPER wrapper_name  
    OPTIONS (NODE 'node_name', DBNAME 'database_name');  
Although the name of the node is specified as an option in the CREATE SERVER statement, it is required for ODBC data sources.  
After the server definition is created, use the ALTER SERVER statement to add or drop server options. |

After you complete this task, you can create a user mapping.

**CREATE SERVER statement - Examples of the ODBC wrapper**

Use the CREATE SERVER statement to register server definitions for the ODBC wrapper. This topic provides a complete example with the required parameters, and an example with additional server options.

The following example shows you how to register a server definition for a MySQL data source by issuing the CREATE SERVER statement:

```
CREATE SERVER mysql_server TYPE mysql  
    VERSION 4.0 WRAPPER wrapper_name  
    OPTIONS (NODE 'odbc_node', DBNAME 'venice')
```

*mysql_server*  
A name that you assign to the ODBC data source server. Duplicate server definition names are not allowed.

*TYPE mysql*  
Specifies the type of data source server to which you are configuring access. This parameter is optional.

*VERSION 4.0*  
The version of the ODBC data source that you want to access. This parameter is optional.

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

*NODE 'odbc_node'*  
The name of the node (the system DSN name) that was assigned to the ODBC data source when the DSN was defined. On federated servers that run Windows, this value must be the name of a system DSN in the ODBC Data Source Administrator window. On federated servers that run UNIX, the name of the node is the DSN defined in the ODBC configuration file. The ODBC configuration file is usually called odbc.ini.  
Although the name of the node is specified as an option in the CREATE SERVER statement, it is required for ODBC data sources.

*DBNAME 'venice'*  
Optional. The name of the ODBC data source that you want to access.
**Server options**

When you create the server definition, you can specify additional server options in the CREATE SERVER statement. The server options can be general server options and ODBC-specific server options.

Some ODBC data sources (for example, MySQL) cannot process quotation marks around table names and column names in SQL statements. To access these data sources, you must include the following server options in the CREATE SERVER statement:

- `DB2_TABLE_QUOTE_CHAR` '``'
- `DB2_ID_QUOTE_CHAR` '``'
- `DB2_AUTHID_QUOTE_CHAR` '``'

The `` character is the delimiter for identifiers such as schema names, table names, and column names.

For example:

```sql
CREATE SERVER mysql_server TYPE mysql
    VERSION 4.0 WRAPPER wrapper_name
    OPTIONS (NODE 'mysql_node', DB2_TABLE_QUOTE_CHAR '``',
              DB2_ID_QUOTE_CHAR '``', DB2_AUTHID_QUOTE_CHAR '``')
```

**Creating a user mapping for an ODBC data source**

When you attempt to access an ODBC server, the federated server establishes a connection to the ODBC server by using a user ID and password that are valid for that data source. For data sources that require a user mapping, you must define an association (a user mapping) between each federated server user ID and password and the corresponding data source user ID and password.

**About this task**

Create a user mapping for each user ID that will access the federated system to send distributed requests to the ODBC data source.

**Procedure**

To map a local user ID to the ODBC data source user ID and password:

Issue a CREATE USER MAPPING statement.

For example:

```sql
CREATE USER MAPPING FOR local_userID SERVER server_definition_name
    OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password');
```

Although the `remote_userID` and `remote_password` values are specified as options in the CREATE USER MAPPING statement, these options are required to access ODBC data sources.

After you complete this task, you can test the connection to the ODBC data source.
CREATE USER MAPPING statement - Examples for the ODBC wrapper

Use the CREATE USER MAPPING statement to map a federated server user ID to an ODBC data source user ID and password. This topic provides a complete example with the required parameters, and an example that shows you how to use the DB2 special register USER with the CREATE USER MAPPING statement.

The following example shows how to map a federated authorization ID to an ODBC data source user ID and password:

```sql
CREATE USER MAPPING FOR arturo SERVER mysql_server
  OPTIONS (REMOTE_AUTHID 'art', REMOTE_PASSWORD 'red4blue')
```

- **arturo**: Specifies the local authorization ID that you are mapping to the remote user ID and password, which are defined at the ODBC data source.
- **mysql_server**: Specifies the server definition name that you defined in the CREATE SERVER statement for the ODBC data source.
- **'art'**: Specifies the remote user ID to which you are mapping arturo. The value is case-sensitive, unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.
- **'red4blue'**: Specifies the remote password that is associated with 'art'. The value is case-sensitive, unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

**DB2 special register USER**

You can use the DB2 special register USER to map the authorization ID of the person who is issuing the CREATE USER MAPPING statement to the data source authorization ID that is specified in the REMOTE_AUTHID user option.

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

```sql
CREATE USER MAPPING FOR USER SERVER mysql_server
  OPTIONS (REMOTE_AUTHID 'art', REMOTE_PASSWORD 'red4blue');
```

**Testing the connection to the ODBC data source server**

Test the connection to the ODBC data source server to determine if the federated server is properly configured to access ODBC data sources.

**About this task**

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

**Procedure**

To test the connection to the ODBC data source server:

Open a pass-through session and issue a SELECT statement on the ODBC data source system tables. If the SELECT statement returns a count, your server definition and your user mapping are set up properly.
SET PASSTHRU server_definition_name
SELECT count(*) FROM schema_name.table_name
SET PASSTHRU RESET

If the SELECT statement returns an error, you should troubleshoot the connection errors.

After you complete this task, you can register nicknames for the ODBC data source tables and views.

### Registering nicknames for ODBC data source tables and views

For each ODBC server definition that you register, you must register a nickname for each table or view that you want to access. Use these nicknames, instead of the names of the data source objects, when you query the ODBC data sources.

#### Before you begin

Update the statistics at the ODBC data source before you register a nickname. The federated database relies on the data source catalog statistics to optimize query processing. Use the data source command that is equivalent to the DB2 RUNSTATS command to update the data source statistics.

#### Procedure

To register a nickname for an ODBC data source table or view, use one of the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
</tbody>
</table>
| Issue the CREATE NICKNAME statement. Nicknames can be up to 128 characters in length. | For example: 
CREATE NICKNAME nickname
FOR server_definition_name."remote_schema"."remote.table"; |

When you create the nickname, the federated server queries the data source catalog using the nickname. This query tests the connection to the data source table or view. If the connection does not work, you receive an error message.

Repeat this step for each ODBC table or view that you want to create a nickname for.

### CREATE NICKNAME statement - Examples for the ODBC wrapper

Use the CREATE NICKNAME statement to register a nickname for an ODBC table or view that you want to access. This topic provides a complete example with the required parameters.

The following example shows how to register a nickname for an ODBC table or view using the CREATE NICKNAME statement.

CREATE NICKNAME cust_europe FOR mysql_server."vinnie"."italy"
A unique nickname that is used to identify the ODBC table or view. The nickname must be unique within the schema.

**Important:** The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authorization ID of the user who registers the nickname.

If your ODBC data source does not support schemas, omit the schema from the CREATE NICKNAME statement.

A three-part identifier for the remote object:

- `mysql_server` is the server definition name that you assigned to the ODBC data source server in the CREATE SERVER statement.
- `vinnie` is the user ID of the owner to which the table or view belongs.
- `italy` is the name of the remote table or view that you want to access.

The federated server folds the names of the ODBC schemas and tables to uppercase unless you enclose the names in quotation marks.

---

**Optimizing ODBC wrapper performance with the ODBC tuning utility (db2fedsvrcfg)**

You can optimize the performance of the ODBC wrapper with the ODBC tuning utility, `db2fedsvrcfg`. This utility runs a set of predefined queries against the data source and tests the results for accuracy. The utility creates a set of ALTER SERVER statements that you can run against the server to set the server options for optimal performance.

**Before you begin**

The following items must be configured on your federated server:

- Federation
- The ODBC wrapper
- The ODBC client software
- Variables for the system environment. The ODBCINI and LIBPATH variables might be required.

For information about installation, configuration, and ODBC requirements, see the documentation that is provided with each of these products.

**Procedure**

To optimize ODBC wrapper performance with the ODBC tuning utility:

1. If your ODBC server has not been defined, use the DB2 Control Center or DB2 Command Line Processor to connect to the federated server and create an ODBC wrapper and server.
2. Optional: If the test tables already exist on your ODBC data source, connect to your ODBC data source and drop them.
3. Run the ODBC tuning utility from the DB2 command line with the options that you want to use. The utility might take some time to complete.
4. Verify that the utility ran successfully. If the utility ran successfully, you see
the following message in the command window or in the output file that you
specified:

```
ALTER SERVER "DS1" OPTIONS (ADD option1, 'value1')
ALTER SERVER "DS1" OPTIONS (ADD option2, 'value2')
ALTER SERVER "DS1" OPTIONS (ADD option3, 'value3')
```

The db2fedsvrcfg command completed successfully.

If the command ran unsuccessfully, you see an error message indicating the
reason for the error. Correct the problem and run the command again.

You must create the test tables manually under the following circumstances:

- If you want to use table names that are different from the default
- If the data source that you are accessing through ODBC is read-only
- If the ODBC tuning utility is unable to create the test tables that are
  required by the utility

5. Connect to the federated server where the data source is defined.

6. Use the db2look utility to save your existing server settings before you run the
   file that is created by the ODBC tuning utility. See the documentation for the
   db2look utility for information about saving your existing server settings.

7. Optional: If your ODBC server is defined, you can connect to the federated
   server and drop the server options. The utility creates ALTER SERVER
   statements in the format described in step 4. If these server options have
   already been added, the ALTER SERVER statements will fail.

8. Use the following command to run the ALTER SERVER statements that were
   generated by the utility against the federated database. The ALTER SERVER
   statements that were created by the ODBC tuning utility are contained in the
   db2fedsvrcfg.sql file.

   `db2 -tvf db2fedsvrcfg.sql`

9. Verify that the results of the ALTER SERVER statements. If any of the
   statements failed, you can modify the statements in the db2fedsvrcfg.sql file
   and run the statements again until they succeed.

10. After you have completed tuning the server with the ODBC tuning utility, set
    the PUSHDOWN server option to ‘Y’ to complete the optimization process.

**db2fedsvrcfg command syntax - ODBC tuning utility**

Use the db2fedsvrcfg command to improve the performance of the ODBC wrapper.

**Syntax**

```
db2fedsvrcfg -s serverName [-m odbcDriverManagerLibrary] -dsn odbcDSNname
    tableNameSuffix] [-dscp codePage] [-v] [-o outputFile] [-h]
```

**Parameters**

Use the command db2fedsvrcfg32 when you use 32-bit ODBC drivers on AIX or
Solaris. Otherwise, use the command db2fedsvrcfg.

- **-dbname dsDBname**
  The database name of the data source.
-dscp codePage
The code page identifier for the data source. If this option is not specified, the utility uses the code page of the user's environment. This parameter is optional.

-dsn odbcDSNname
The system data source name (DSN) for the data source.

-h Causes detailed help to be displayed. This parameter is optional.

-m odbcDriverManagerLibrary
The fully qualified file name of the ODBC driver manager library. The ODBC driver manager library file name is optional for Windows.

-noPrep
Prevents the testing tables from being created at the data source before testing. This parameter is optional.

-o outputFile
The fully qualified file name for the ODBC tuning utility output file. The output file contains the ALTER SERVER statements that are used to tune the ODBC wrapper performance. This parameter is optional. If this parameter is not specified, the output is displayed in the command window.

-p password
The password for connecting to the data source. This parameter is optional.

-prefix tableNamePrefix
The prefix of the ODBC data source table names that the utility uses for the analysis. If a prefix is not specified, the default prefix, IITEST, is used. This parameter is optional.

-s serverName
The name of the federated server.

-suffix tableNameSuffix
The suffix of the ODBC data source table names that the utility uses for the analysis. If a suffix is not specified an empty string is used.

-u userId
The user name for connecting to the data source. This parameter is optional.

-v Specifies that the output of the utility is verbose. This parameter is optional.

Example

The following example shows the command that is run against datastore, the ODBC data source. In this example, the test tables are named ABCnXYZ, where n is a number from 1 to 7.

db2fedsvrcfg -s DS1 -m "/usr/lib/odbc.a"
   -dsn datastore -dbname db1 -u authid -p password -noprep
   -prefix ABC -suffix XYZ -o "/home/user1/db2fedsvrcfg.sql"

Test table definitions for the ODBC tuning utility (db2fedsvrcfg)

In some cases, you must create the test tables for the ODBC tuning utility manually. The test table definitions are described in this topic.

You must create the test tables for the ODBC tuning utility under the following circumstances:

- If you want to use table names that are different from the default
If the data source that you are accessing through ODBC is read-only
If the ODBC tuning utility is unable to create the test tables that are required by the utility

The following table definition applies to all seven of the test tables that are needed (IITEST1 through IITEST7). The default table name prefix is IITEST and the default suffix is an empty string. If you specify a different prefix and suffix, you must specify the -prefix and -suffix options when you run the ODBC tuning utility.

Table 69. ODBC tuning utility test table definition for the table IITEST1

<table>
<thead>
<tr>
<th>Column name</th>
<th>SQL data type</th>
<th>SQL data type identifier</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT1C1</td>
<td>integer</td>
<td>SQL_INTEGER</td>
<td></td>
</tr>
<tr>
<td>IT1C2</td>
<td>integer</td>
<td>SQL_INTEGER</td>
<td></td>
</tr>
<tr>
<td>IT1C3</td>
<td>char(1)</td>
<td>SQL_CHAR</td>
<td>1</td>
</tr>
<tr>
<td>IT1C4</td>
<td>char(3)</td>
<td>SQL_CHAR</td>
<td>3</td>
</tr>
<tr>
<td>IT1C5</td>
<td>char(10)</td>
<td>SQL_CHAR</td>
<td>10</td>
</tr>
<tr>
<td>IT1C6</td>
<td>varchar(10)</td>
<td>SQL_VARCHAR</td>
<td>10</td>
</tr>
<tr>
<td>IT1C7</td>
<td>char(100)</td>
<td>SQL_CHAR</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 70. ODBC tuning utility test table definition for the table IITEST2

<table>
<thead>
<tr>
<th>Column name</th>
<th>SQL data type</th>
<th>SQL data type identifier</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT2C1</td>
<td>integer</td>
<td>SQL_INTEGER</td>
<td></td>
</tr>
<tr>
<td>IT2C2</td>
<td>integer</td>
<td>SQL_INTEGER</td>
<td></td>
</tr>
<tr>
<td>IT2C3</td>
<td>char(30)</td>
<td>SQL_CHAR</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 71. ODBC tuning utility test table definition for the tables IITEST3 through IITEST7

<table>
<thead>
<tr>
<th>Number of columns in each table</th>
<th>Column name</th>
<th>SQL data type</th>
<th>SQL data type identifier</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>66</td>
<td>ITxCn</td>
<td>char(100)</td>
<td>SQL_CHAR</td>
<td>100</td>
</tr>
</tbody>
</table>

For example, IT3C1 is the column name for the first column in table IITEST3.

Accessing Excel data using the ODBC wrapper

You can access Microsoft Excel workbooks with the ODBC wrapper by using the Excel ODBC driver.

Before you begin
- The Excel ODBC driver must be on the federated server.
- The federated server must be able to open and read the worksheets in the Excel workbook to retrieve the data. Therefore, the Excel workbooks must be on the same computer as the federated server or on a accessible mapped network drive.
- Format the columns according to the type of data the column is expected to have.
- The data inserted in the columns must comply with the format type that is specified for the column.
If the first eight rows of the spreadsheet have no data make sure that they are empty. To ensure that a cell is empty, open the spreadsheet in Microsoft Excel and select Edit → Clear All.

Ensure that the data inserted into the columns in the spreadsheet comply with the specified type.

**Restrictions**

- The ODBC wrapper cannot access a worksheet when the workbook is already opened by a user or an application in the read/write mode. However, if the ODBC wrapper opens the workbook before a user or an application opens the workbook, the user or application can open the workbook in read-only mode.
- The Excel ODBC driver expects that the first nonblank row contains the labels for the worksheet columns. You must insert a row of column labels in the worksheet if the worksheet does not have the labels.
- Because the Excel ODBC driver is only available for Windows operating systems, you can use the ODBC wrapper to access Excel data only on federated servers that run Windows.
- You can perform insert and update operations on Excel worksheets, but you cannot perform delete operations. The Excel ODBC driver does not support delete operations. To delete data from the worksheet, you must open the worksheet in Excel to make the changes.

**About this task**

The Excel application does not need to be installed on the federated server. The Excel ODBC driver is automatically installed with Windows.

With the ODBC wrapper and the Excel ODBC driver, you can access data from any of the worksheets within a workbook. The Excel ODBC driver interprets a workbook as a database and interprets each worksheet within the workbook as a table.

The Excel ODBC driver supports earlier versions of Excel workbooks even if the version of Excel application that produced the workbooks is no longer supported. For example, Microsoft no longer supports workbooks created in Excel Version 4.0, but the driver supports Excel worksheets that were created in that version.

**Procedure**

To access Excel worksheets with the ODBC wrapper:

1. Ensure that the Excel workbook that you want to access is on the federated server or on an accessible mapped network drive.
2. If necessary, change the layout of the data in the Excel worksheets to adhere to the Excel ODBC driver requirements. Repeat this step for each worksheet or named range that you want to access.
3. If necessary, create any named ranges that you want to access.
4. Create a system DSN for the workbook that you want to access. You can use the ODBC Data Source Administrator to configure the system DSN. The name that was specified when you created the system DSN is assigned as the value for the NODE option in the CREATE SERVER statement.
5. Issue the CREATE WRAPPER statement.
6. Specify the location of the workbook by registering a server object in the federated database system catalog. For the ODBC wrapper, you need a server
object for each DSN. The DSN is associated with the workbook when the Excel ODBC driver is used. The NODE compounds_workbook_dsn is the system DSN that you created. The NODE option is required for the ODBC wrapper to access Excel worksheets.

To specify the location of the workbook, issue the CREATE SERVER statement and use the DSN as the system DSN for the NODE option.

For example:

```
CREATE SERVER compounds_workbook WRAPPER odbc
   OPTIONS (NODE 'compounds_workbook_dsn', PASSWORD 'n')
```

Repeat this step for each workbook that you plan to access.

7. Issue the CREATE NICKNAME statement to create a nickname for the worksheet that you want to access. The syntax is:

```
CREATE NICKNAME nickname FOR server_name.remote_table
```

8. If you created a named range to access the data, specify the name of the range as the remote_table portion of the CREATE NICKNAME statement.

For example, if the name of the range is testing, the CREATE NICKNAME statement is:

```
CREATE NICKNAME compoundsNickname FOR compounds_workbook.testing
```

To access the data in the entire worksheet instead of a range, you specify the name of the worksheet followed by the $ symbol.

For example, if the name of the worksheet is Sheet1, the CREATE NICKNAME statement is:

```
CREATE NICKNAME compounds_nickname FOR compounds_workbook.Sheet1$
```
Chapter 11. Configuring ODBC access to WebSphere Classic Federation Server for z/OS data sources

The ODBC wrapper has been optimized to access WebSphere Classic Federation Server for z/OS data sources. The ODBC wrapper detects the ODBC driver and automatically configures the performance options.

Before you begin

The following items must be configured on your federated server:
- Federated server
- WebSphere Classic Federation Server for z/OS client
- Federated database
- Variables for the system environment db2dj.ini file, DB2 profile registry (db2set). The LIBPATH environment variable might be required. For Linux and UNIX systems, the CAC_CONFIG variable must be set to the path of the cac.ini file. For example:
  
  CAC_CONFIG=/home/db2inst1/cac.ini
- On AIX and Solaris systems, the DB2_FENCED wrapper option must be set to Y and the DB2_SOURCE_CLIENT_MODE wrapper option must be set to 32BIT.

For information about installation, configuration, and ODBC requirements, see the documentation that is provided with each of these products.

Restrictions

The ODBC wrapper does not support the following statement with WebSphere Classic Federation Server for z/OS data sources:
- CREATE TABLE

The following data types are not supported for use with WebSphere Classic Federation Server for z/OS data sources:
- BLOB
- CLOB
- DBCLOB
- CHAR FOR BIT DATA
- VARCHAR FOR BIT DATA

Procedure

To configure ODBC access to WebSphere Classic Federation Server for z/OS data sources:
1. Complete one of the following tasks depending on your operating system:
   - On Windows, ensure that the node name for the data source is defined as a system DSN. If you used Microsoft ODBC Data Source Administrator to define the DSN, you can use the Control Panel to verify that it is registered as a system DSN.
   - On UNIX, see the documentation for WebSphere Classic Federation Server for z/OS for instructions on how to configure the ODBC client.
2. You might need to add the data source client library path to the DB2LIBPATH registry variable to allow the client libraries to load properly. Issue the following command to set the registry variable:

```
   db2set DB2LIBPATH="/opt/IBM/DB2IIClassic82/cli/lib" 
   
   /opt/IBM/DB2IIClassic82/cli/lib
```

The path to the library directory of your data source client.

3. Register the ODBC wrapper

4. Register the server definitions for an ODBC data source

5. Create a user mapping for an ODBC data source

6. Test the connection to the ODBC data source server

7. Register nicknames for ODBC data source tables and views

---

### Registering the ODBC wrapper

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

**Procedure**

To register the ODBC wrapper:

Use one of the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use the Federated Objects wizard in the DB2 Control Center.</strong></td>
<td>To start the wizard, right-click the Federated Database Objects folder and click <strong>Create Federated Objects</strong>.</td>
</tr>
</tbody>
</table>
| Issue the `CREATE WRAPPER` statement and specify the default name for the ODBC wrapper. | For example:  
   ```
   CREATE WRAPPER ODBC;
   ```
   **Remember:** When you register the wrapper by using the default name, ODBC, the federated server automatically uses the appropriate ODBC wrapper library for the operating system that your federated server is running on.

   You must specify the **MODULE** wrapper option on federated servers that run Linux or UNIX. The **MODULE** wrapper option specifies the full path of the library that contains the ODBC Driver Manager.

   For AIX and Solaris, if you are configuring access to WebSphere Classic Federation Server on z/OS data sources, you must specify the **DB2_FENCED** and **DB2_SOURCE_CLIENT_MODE** options as shown in the following example.

   ```
   CREATE WRAPPER ODBC
   LIBRARY 'libdb2rcodbc.a'
   OPTIONS (DB2_FENCED 'Y',
            DB2_SOURCE_CLIENT_MODE '32BIT',
            MODULE '/opt/IBM/DB2IIClassic82/cli/lib/cacsqlcli.so')
   ```

---
To register a wrapper with the name `odbc_wrapper` on a federated server that uses AIX, issue the following statement:

```
CREATE WRAPPER odbc_wrapper
LIBRARY 'libdb2rcodbc.a';
```

The wrapper library file that you specify depends on the operating system of the federated server.

After you complete this task, you can register the server definitions.

**ODBC wrapper library files**

The ODBC wrapper library files are added to the federated server when you install the wrapper.

When you install the ODBC wrapper, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files added to the directory path are `libdb2rcodbc.a`, `libdb2rcodbcF.a`, and `libdb2rcodbcU.a`. The default wrapper library file is `libdb2rcodbc.a`. The other wrapper library files are used internally by the ODBC wrapper.

If you do not use the default wrapper name when you register a wrapper, you must include the `LIBRARY` parameter in the `CREATE WRAPPER` statement and specify the default wrapper library file name.

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

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Wrapper library file</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/install_path/lib32/</td>
<td>libdb2rcodbc.a</td>
</tr>
<tr>
<td></td>
<td>/usr/opt/install_path/lib64/</td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/IBM/db2/install_path/lib32</td>
<td>libdb2rcodbc.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/install_path/lib64</td>
<td></td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/IBM/db2/install_path/lib32</td>
<td>libdb2rcodbc.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/install_path/lib64</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>%DB2PATH%\bin</td>
<td>db2rcodbc.dll</td>
</tr>
</tbody>
</table>

`install_path` is the directory path where federation is installed on UNIX or Linux.

**CREATE WRAPPER statement - Examples for the ODBC wrapper**

Use the `CREATE WRAPPER` statement to register the ODBC wrapper. This topic provides examples for Linux, UNIX and Windows.
In the following examples, `odbc_wrapper` is the name that you assign to the wrapper that you are registering in the federated database.

**Linux and Solaris federated server**

The following example shows you how to register a wrapper with the default name on a federated server that runs Linux or Solaris:

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

You must specify the MODULE wrapper option on federated servers that run Linux or UNIX. The MODULE wrapper option specifies the full path of the library that contains the ODBC Driver Manager.

The following example shows you how to register a wrapper with an alternative name on a federated server that runs Linux or Solaris:

```
CREATE WRAPPER odbc_wrapper LIBRARY 'libdb2rcodbc.so'
   OPTIONS (MODULE '/opt/lib/odbc.so');
```

**AIX federated server**

The following example shows you how to register a wrapper with the default name on a federated server that runs AIX:

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

You must specify the MODULE wrapper option on federated servers that run UNIX. The MODULE wrapper option specifies the full path of the library that contains the ODBC Driver Manager.

The following example shows you how to register a wrapper with an alternative name on a federated server that runs AIX:

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

**WebSphere Classic Federation Server for z/OS (AIX)**

The following example shows you how to register an ODBC wrapper by issuing the CREATE WRAPPER statement on an AIX operating system. On AIX and Solaris, the `DB2_FENCED` and `DB2_SOURCE_CLIENT_MODE` options must be specified as shown in the following example.

```
CREATE WRAPPER odbc
   OPTIONS (DB2_FENCED 'Y', DB2_SOURCE_CLIENT_MODE '32BIT',
   MODULE '/opt/IBM/DB2IIClassic82/cli/lib/cacsqlcli.so');
```

You must specify the MODULE wrapper option on federated servers that run UNIX. The MODULE wrapper option specifies the full path of the library that contains the ODBC Driver Manager.

The following example shows you how to register a wrapper for accessing WebSphere Classic Federation Server for z/OS data sources with an alternative name:

```
CREATE WRAPPER odbc_wrapper LIBRARY 'libdb2rcodbc.a'
   OPTIONS (DB2_FENCED 'Y', DB2_SOURCE_CLIENT_MODE '32BIT',
   MODULE '/opt/IBM/DB2IIClassic82/cli/lib/cacsqlcli.so');
```
Windows federated server

The following example shows you how to register a wrapper with the default name on a federated server that runs Windows:

```sql
CREATE WRAPPER odbc;
```

The following example shows you how to register a wrapper with an alternative name on a federated server that runs Windows:

```sql
CREATE WRAPPER odbc_wrapper LIBRARY 'db2rcodbc.dll';
```

Registering the server definitions for an ODBC data source

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

Procedure

To register a server definition for an ODBC data source:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Issue the CREATE SERVER statement.</td>
<td>For example: CREATE SERVER server_definition_name TYPE data_source_type VERSION version_number WRAPPER wrapper_name OPTIONS (NODE 'node_name', DBNAME 'database_name'); Although the name of the node is specified as an option in the CREATE SERVER statement, it is required for ODBC data sources. After the server definition is created, use the ALTER SERVER statement to add or drop server options.</td>
</tr>
</tbody>
</table>

After you complete this task, you can create a user mapping.

CREATE SERVER statement - Examples of the ODBC wrapper

Use the CREATE SERVER statement to register server definitions for the ODBC wrapper. This topic provides a complete example with the required parameters, and an example with additional server options.

The following example shows you how to register a server definition for a MySQL data source by issuing the CREATE SERVER statement:

```sql
CREATE SERVER mysql_server TYPE mysql
  VERSION 4.0 WRAPPER wrapper_name
  OPTIONS (NODE 'odbc_node', DBNAME 'venice')

mysql_server
  A name that you assign to the ODBC data source server. Duplicate server definition names are not allowed.

TYPE mysql
  Specifies the type of data source server to which you are configuring access. This parameter is optional.
```
**VERSION 4.0**
The version of the ODBC data source that you want to access. This parameter is optional.

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

**NODE 'odbc_node'**
The name of the node (the system DSN name) that was assigned to the ODBC data source when the DSN was defined. On federated servers that run Windows, this value must be the name of a system DSN in the ODBC Data Source Administrator window. On federated servers that run UNIX, the name of the node is the DSN defined in the ODBC configuration file. The ODBC configuration file is usually called odbc.ini.

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

**DBNAME 'venice'**
Optional. The name of the ODBC data source that you want to access.

### Server options

When you create the server definition, you can specify additional server options in the CREATE SERVER statement. The server options can be general server options and ODBC-specific server options.

Some ODBC data sources (for example, MySQL) cannot process quotation marks around table names and column names in SQL statements. To access these data sources, you must include the following server options in the CREATE SERVER statement:

- `DB2_TABLE_QUOTE_CHAR ''`
- `DB2_ID_QUOTE_CHAR ''`
- `DB2_AUTHID_QUOTE_CHAR ''`

The `` character is the delimiter for identifiers such as schema names, table names, and column names.

For example:
```
CREATE SERVER mysql_server TYPE mysql
  VERSION 4.0 WRAPPER wrapper_name
  OPTIONS (NODE 'mysql_node', DB2_TABLE_QUOTE_CHAR '',
            DB2_ID_QUOTE_CHAR '',
            DB2_AUTHID_QUOTE_CHAR '')
```

### Creating a user mapping for an ODBC data source

When you attempt to access an ODBC server, the federated server establishes a connection to the ODBC server by using a user ID and password that are valid for that data source. For data sources that require a user mapping, you must define an association (a user mapping) between each federated server user ID and password and the corresponding data source user ID and password.

**About this task**

Create a user mapping for each user ID that will access the federated system to send distributed requests to the ODBC data source.
**Procedure**

To map a local user ID to the ODBC data source user ID and password:

Issue a CREATE USER MAPPING statement.

For example:

```
CREATE USER MAPPING FOR local_userID SERVER server_definition_name
   OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password');
```

Although the `remote_userID` and `remote_password` values are specified as options in the CREATE USER MAPPING statement, these options are required to access ODBC data sources.

After you complete this task, you can test the connection to the ODBC data source.

**CREATE USER MAPPING statement - Examples for the ODBC wrapper**

Use the CREATE USER MAPPING statement to map a federated server user ID to an ODBC data source user ID and password. This topic provides a complete example with the required parameters, and an example that shows you how to use the DB2 special register USER with the CREATE USER MAPPING statement.

The following example shows how to map a federated authorization ID to an ODBC data source user ID and password:

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

- `arturo` Specifies the local authorization ID that you are mapping to the remote user ID and password, which are defined at the ODBC data source.

- `mysql_server` Specifies the server definition name that you defined in the CREATE SERVER statement for the ODBC data source.

- `'art'` Specifies the remote user ID to which you are mapping `arturo`. The value is case-sensitive, unless you set the FOLD_ID server option to ‘U’ or ‘L’ in the CREATE SERVER statement.

- `'red4blue'` Specifies the remote password that is associated with `‘art’`. The value is case-sensitive, unless you set the FOLD_PW server option to ‘U’ or ‘L’ in the CREATE SERVER statement.

**DB2 special register USER**

You can use the DB2 special register USER to map the authorization ID of the person who is issuing the CREATE USER MAPPING statement to the data source authorization ID that is specified in the REMOTE_AUTHID user option.

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

```
CREATE USER MAPPING FOR USER SERVER mysql_server
   OPTIONS (REMOTE_AUTHID 'art', REMOTE_PASSWORD 'red4blue');
```
Testing the connection to the ODBC data source server

Test the connection to the ODBC data source server to determine if the federated server is properly configured to access ODBC data sources.

About this task

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

Procedure

To test the connection to the ODBC data source server:

Open a pass-through session and issue a SELECT statement on the ODBC data source system tables. If the SELECT statement returns a count, your server definition and your user mapping are set up properly.

```sql
SET PASSTHRU server_definition_name
SELECT count(*) FROM schema_name.table_name
SET PASSTHRU RESET
```

If the SELECT statement returns an error, you should troubleshoot the connection errors.

After you complete this task, you can register nicknames for the ODBC data source tables and views.

Registering nicknames for ODBC data source tables and views

For each ODBC server definition that you register, you must register a nickname for each table or view that you want to access. Use these nicknames, instead of the names of the data source objects, when you query the ODBC data sources.

Before you begin

Update the statistics at the ODBC data source before you register a nickname. The federated database relies on the data source catalog statistics to optimize query processing. Use the data source command that is equivalent to the DB2 RUNSTATS command to update the data source statistics.

Procedure

To register a nickname for an ODBC data source table or view, use one of the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Issue the CREATE NICKNAME statement. Nicknames can be up to 128 characters in length.</td>
<td>For example: CREATE NICKNAME nickname FOR server_definition_name.&quot;remote_schema&quot;.&quot;remote_table&quot; ;</td>
</tr>
</tbody>
</table>
When you create the nickname, the federated server queries the data source catalog using the nickname. This query tests the connection to the data source table or view. If the connection does not work, you receive an error message.

Repeat this step for each ODBC table or view that you want to create a nickname for.

**CREATE NICKNAME statement - Examples for the ODBC wrapper**

Use the CREATE NICKNAME statement to register a nickname for an ODBC table or view that you want to access. This topic provides a complete example with the required parameters.

The following example shows how to register a nickname for an ODBC table or view using the CREATE NICKNAME statement.

```
CREATE NICKNAME cust_europe FOR mysql_server."vinnie"."italy"
```

cust_europe
A unique nickname that is used to identify the ODBC table or view. The nickname must be unique within the schema.

**Important:** The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authorization ID of the user who registers the nickname.

If your ODBC data source does not support schemas, omit the schema from the CREATE NICKNAME statement.

`mysql_server."vinnie"."italy"`
A three-part identifier for the remote object:
- `mysql_server` is the server definition name that you assigned to the ODBC data source server in the CREATE SERVER statement.
- `vinnie` is the user ID of the owner to which the table or view belongs.
- `italy` is the name of the remote table or view that you want to access.

The federated server folds the names of the ODBC schemas and tables to uppercase unless you enclose the names in quotation marks.

### Optimizing ODBC wrapper performance with the ODBC tuning utility (db2fedsvrcfg)

You can optimize the performance of the ODBC wrapper with the ODBC tuning utility, db2fedsvrcfg. This utility runs a set of predefined queries against the data source and tests the results for accuracy. The utility creates a set of ALTER SERVER statements that you can run against the server to set the server options for optimal performance.

**Before you begin**

The following items must be configured on your federated server:
- Federation
- The ODBC wrapper
- The ODBC client software
• Variables for the system environment. The ODBCINI and LIBPATH variables might be required.

For information about installation, configuration, and ODBC requirements, see the documentation that is provided with each of these products.

Procedure

To optimize ODBC wrapper performance with the ODBC tuning utility:
1. If your ODBC server has not been defined, use the DB2 Control Center or DB2 Command Line Processor to connect to the federated server and create an ODBC wrapper and server.
2. Optional: If the test tables already exist on your ODBC data source, connect to your ODBC data source and drop them.
3. Run the ODBC tuning utility from the DB2 command line with the options that you want to use. The utility might take some time to complete.
4. Verify that the utility ran successfully. If the utility ran successfully, you see the following message in the command window or in the output file that you specified:

   ALTER SERVER "DS1" OPTIONS (ADD option1, 'value1')
   ALTER SERVER "DS1" OPTIONS (ADD option2, 'value2')
   ALTER SERVER "DS1" OPTIONS (ADD option3, 'value3')
   
   The db2fedsvrcfg command completed successfully.

   If the command ran unsuccessfully, you see an error message indicating the reason for the error. Correct the problem and run the command again.

   You must create the test tables manually under the following circumstances:
   • If you want to use table names that are different from the default
   • If the data source that you are accessing through ODBC is read-only
   • If the ODBC tuning utility is unable to create the test tables that are required by the utility

5. Connect to the federated server where the data source is defined.
6. Use the db2look utility to save your existing server settings before you run the file that is created by the ODBC tuning utility. See the documentation for the db2look utility for information about saving your existing server settings.
7. Optional: If your ODBC server is defined, you can connect to the federated server and drop the server options. The utility creates ALTER SERVER statements in the format described in step 4. If these server options have already been added, the ALTER SERVER statements will fail.
8. Use the following command to run the ALTER SERVER statements that were generated by the utility against the federated database. The ALTER SERVER statements that were created by the ODBC tuning utility are contained in the db2fedsvrcfg.sql file.
   
   db2 -tvf db2fedsvrcfg.sql
   
   9. Verify that the results of the ALTER SERVER statements. If any of the statements failed, you can modify the statements in the db2fedsvrcfg.sql file and run the statements again until they succeed.
10. After you have completed tuning the server with the ODBC tuning utility, set the PUSHDOWN server option to 'Y' to complete the optimization process.
**db2fedsvrcfg command syntax - ODBC tuning utility**

Use the db2fedsvrcfg command to improve the performance of the ODBC wrapper.

**Syntax**

```
```

**Parameters**

Use the command db2fedsvrcfg32 when you use 32-bit ODBC drivers on AIX or Solaris. Otherwise, use the command db2fedsvrcfg.

- `-dbname dsDBname`
  The database name of the data source.

- `-dscp codePage`
  The code page identifier for the data source. If this option is not specified, the utility uses the code page of the user’s environment. This parameter is optional.

- `-dsn odbcDSNname`
  The system data source name (DSN) for the data source.

- `-h`
  Causes detailed help to be displayed. This parameter is optional.

- `-m odbcDriverManagerLibrary`
  The fully qualified file name of the ODBC driver manager library. The ODBC driver manager library file name is optional for Windows.

- `-noprep`
  Prevents the testing tables from being created at the data source before testing. This parameter is optional.

- `-o outputFile`
  The fully qualified file name for the ODBC tuning utility output file. The output file contains the ALTER SERVER statements that are used to tune the ODBC wrapper performance. This parameter is optional. If this parameter is not specified, the output is displayed in the command window.

- `-p password`
  The password for connecting to the data source. This parameter is optional.

- `-prefix tableNamePrefix`
  The prefix of the ODBC data source table names that the utility uses for the analysis. If a prefix is not specified, the default prefix, IITEST, is used. This parameter is optional.

- `-s serverName`
  The name of the federated server.

- `-suffix tableNameSuffix`
  The suffix of the ODBC data source table names that the utility uses for the analysis. If a suffix is not specified an empty string is used.

- `-u userid`
  The user name for connecting to the data source. This parameter is optional.

- `-v`
  Specifies that the output of the utility is verbose. This parameter is optional.
The following example shows the command that is run against datastore, the ODBC data source. In this example, the test tables are named ABCnXYZ, where \( n \) is a number from 1 to 7.

```
db2fedsvrcfg -s DS1 -m "/usr/lib/odbc.a"
   -dsn datastore -dbname db1 -u authid -p password -noprep
   -prefix ABC -suffix XYZ -o "/home/user1/db2fedsvrcfg.sql"
```

**Test table definitions for the ODBC tuning utility (db2fedsvrcfg)**

In some cases, you must create the test tables for the ODBC tuning utility manually. The test table definitions are described in this topic.

You must create the test tables for the ODBC tuning utility under the following circumstances:

- If you want to use table names that are different from the default
- If the data source that you are accessing through ODBC is read-only
- If the ODBC tuning utility is unable to create the test tables that are required by the utility

The following table definition applies to all seven of the test tables that are needed (IITEST1 through IITEST7). The default table name prefix is IITEST and the default suffix is an empty string. If you specify a different prefix and suffix, you must specify the -prefix and -suffix options when you run the ODBC tuning utility.

**Table 73. ODBC tuning utility test table definition for the table IITEST1**

<table>
<thead>
<tr>
<th>Column name</th>
<th>SQL data type</th>
<th>SQL data type identifier</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT1C1</td>
<td>integer</td>
<td>SQL_INTEGER</td>
<td></td>
</tr>
<tr>
<td>IT1C2</td>
<td>integer</td>
<td>SQL_INTEGER</td>
<td></td>
</tr>
<tr>
<td>IT1C3</td>
<td>char(1)</td>
<td>SQL_CHAR</td>
<td>1</td>
</tr>
<tr>
<td>IT1C4</td>
<td>char(3)</td>
<td>SQL_CHAR</td>
<td>3</td>
</tr>
<tr>
<td>IT1C5</td>
<td>char(10)</td>
<td>SQL_CHAR</td>
<td>10</td>
</tr>
<tr>
<td>IT1C6</td>
<td>varchar(10)</td>
<td>SQL_VARCHAR</td>
<td>10</td>
</tr>
<tr>
<td>IT1C7</td>
<td>char(100)</td>
<td>SQL_CHAR</td>
<td>100</td>
</tr>
</tbody>
</table>

**Table 74. ODBC tuning utility test table definition for the table IITEST2**

<table>
<thead>
<tr>
<th>Column name</th>
<th>SQL data type</th>
<th>SQL data type identifier</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT2C1</td>
<td>integer</td>
<td>SQL_INTEGER</td>
<td></td>
</tr>
<tr>
<td>IT2C2</td>
<td>integer</td>
<td>SQL_INTEGER</td>
<td></td>
</tr>
<tr>
<td>IT2C3</td>
<td>char(30)</td>
<td>SQL_CHAR</td>
<td>30</td>
</tr>
</tbody>
</table>

**Table 75. ODBC tuning utility test table definition for the tables IITEST3 through IITEST7**

<table>
<thead>
<tr>
<th>Number of columns in each table</th>
<th>Column name</th>
<th>SQL data type</th>
<th>SQL data type identifier</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>66</td>
<td>ITxCn</td>
<td>char(100)</td>
<td>SQL_CHAR</td>
<td>100</td>
</tr>
</tbody>
</table>

\( x \) The number corresponding to the table that is being defined.

\( n \) The column number.
For example, IT3C1 is the column name for the first column in table IITEST3.
Chapter 12. Configuring access to OLE DB data sources

To configure the federated server to access OLE DB data sources, you must provide the federated server with information about the OLE DB providers.

Before you begin

- Federation must be installed on a server that acts as the federated server.
- Check the setup of the federated server.
- Check the federated parameter to ensure that federation is enabled.

Microsoft OLE DB is a set of component object model (COM) interfaces that enable applications to access information that is typically not stored in databases. An OLE DB consumer is any piece of system or application code that consumes an OLE DB interface. An OLE DB provider is any software component that exposes an OLE DB interface.

You can configure a federated server to access data that is stored in OLE DB data sources by issuing SQL statements on the DB2 command line.

After you configure access to the OLE DB data source, use the CREATE FUNCTION statement to register a user-defined OLE DB external table function in the federated database.

Procedure

To configure access to OLE DB data sources to a federated server:

1. **Register the wrapper.**
2. **Register the server definition.**
3. **Create the user mappings.**

Registering the OLE DB wrapper

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

About this task

The OLE DB wrapper enables you to access OLE DB providers that are compliant with Microsoft OLE DB version 2.0 (or later).

The OLE DB wrapper is used only to assist you in registering user-defined OLE DB external table functions in the federated database. Unlike other wrappers, the OLE DB wrapper does not use nicknames to access data that is stored in data sources.

Procedure

To register the OLE DB wrapper:
Use one of the following methods to register the OLE DB wrapper:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
</tbody>
</table>
| Issue the CREATE WRAPPER statement and specify the default name for the OLE DB wrapper. | For example: CREATE WRAPPER OLEDB  

**Remember:** When you register the wrapper by using the default name, OLEDB, the federated server automatically uses the appropriate OLE DB wrapper library for the operating system that your federated server is running on.  

If the default wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name that you choose. When you use a name that is different from the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement.  

For example, to register a wrapper with the name oledb_wrapper on a federated server that uses Windows, issue the following statement:  

```sql  
CREATE WRAPPER oledb_wrapper  
   LIBRARY 'db2oledb.dll'  
```

The wrapper library file that you specify depends on the operating system of the federated server. |

After you complete this task, you can register the server definition.  

**OLE DB wrapper library files**  
The OLE DB wrapper library files are added to the federated server when you install the wrapper.  

When you install the OLE DB wrapper, the library file is added to the default directory path.  

If you do not use the default wrapper name when you register the wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the library file name.  

The default directory path and default wrapper library file name is listed in the following table.  

*Table 76. OLE DB client library locations and file names*  

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Library file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>%DB2PATH%\bin</td>
<td>db2oledb.dll</td>
</tr>
</tbody>
</table>

**Registering the server definitions for an OLE DB data source**  
You must register each OLE DB data source server that you want to access in the federated database.  

**About this task**
You register a server definition for and OLE DB data source from the DB2 command line.

**Procedure**

To register a server definition for an OLE DB data source:

Issue the CREATE SERVER statement.

For example:

```
CREATE SERVER server_definition_name WRAPPER wrapper_name
  OPTIONS (CONNECTSTRING 'keyword=value;keyword=value');
```

Although the CONNECTSTRING variable is specified as an option in the CREATE SERVER statement, this option is required for OLE DB data sources.

After you complete this task, you can create a user mapping.

**CREATE SERVER statement - Examples for the OLE DB wrapper**

Use the CREATE SERVER statement to register server definitions for the OLE DB wrapper. This topic provides a complete example with the required parameters, and an example with additional server options.

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

```
CREATE SERVER Nwind WRAPPER OLEDB
  OPTIONS (CONNECTSTRING 'Provider=Microsoft.Jet.OLED.B.4.0;
                   Data Source=c:\msdasdk\bin\oledb\nwind.mdb');
```

*Nwind* A name that you assign to the OLE DB data source. Duplicate server definition names are not allowed.

**WRAPPER OLEDB**

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

**CONNECTSTRING 'Provider=Microsoft.Jet.OLED.B.4.0; Data Source=c:\msdasdk\bin\oledb\nwind.mdb'**

Provides the initialization properties that are necessary to connect to an OLE DB provider.

The value for the CONNECTSTRING option contains a series of keyword and value pairs that are separated by semicolons. The equal sign (=) separates each keyword and its value. Keywords are the descriptions of the OLE DB initialization properties (property set DBPROPSET_DBINT) or provider-specific keywords.

For the complete syntax and semantics for the CONNECTSTRING option, see the *Microsoft OLE DB 2.0 Programmer’s Reference and Data Access SDK*, Microsoft Press, 1998.

**Server options**

When you create the server definition, you can specify additional server options in the CREATE SERVER statement. The server options can be general server options and OLE DB-specific server options.
The COLLATING_SEQUENCE server option specifies whether the data source uses the same collating sequence as the federated server. The default setting for the COLLATING_SEQUENCE server option is ‘N’. When the OLE DB data source uses the same collating sequence as the federated server, set the COLLATING_SEQUENCE server option is ‘Y’.

The following example an OLE DB server definition that includes the COLLATING_SEQUENCE server option:

```
CREATE SERVER Nwind WRAPPER OLEDB
  OPTIONS (CONNECTSTRING 'Provider=Microsoft.Jet.OLEDB.4.0;
                Data Source=c:\msdasdk\bin\oledb\nwind.mdb',
             COLLATING_SEQUENCE 'Y')
```

Creating the user mappings for an OLE DB data source

When you attempt to access an OLE DB server, the federated server establishes a connection to the OLE DB server by using a user ID and password that are valid for that data source. You must define an association (a user mapping) between each federated authorization ID and password and the corresponding data source user ID and password.

Procedure

To map a local authorization ID to the remote OLE DB user ID and password:

Issue a CREATE USER MAPPING statement.

For example:

```
CREATE USER MAPPING FOR local_userid SERVER server_definition_name
  OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password');
```

If the length of either the password on the OLE DB data source or the password on the federated server is less than eight characters, SQL statements that access the OLE DB data source will fail. The following error message appears:

```
SQL30082N Attempt to establish connection failed with security reason "15"
   ("PROCESSING FAILURE"). SQLSTATE=08001
```

To avoid this problem, either change the OLE DB data source password or change the password on the federated server to eight or more characters.

CREATE USER MAPPING statement - Examples for the OLE DB wrapper

Use the CREATE USER MAPPING statement to map a federated authorization ID to a remote OLE DB user ID and password. This topic includes a complete example with the required parameters, and an example that shows you how to use the DB2 special register USER with the CREATE USER MAPPING statement.

The following example shows how to map a federated authorization ID to a remote OLE data source user ID and password:

```
CREATE USER MAPPING FOR laura SERVER Nwind
  OPTIONS (REMOTE_AUTHID 'lulu', REMOTE_PASSWORD 'raiders')
```

```
laura
```

Specifies the local authorization ID that you are mapping to a remote user ID and password, which are defined at the OLE DB data source.
SERVER Nwind
   Specifies the server definition name that you registered in the CREATE SERVER statement for the OLE DB server.

REMOTE_AUTHID 'lulu'
   Specifies the user ID at the OLE DB database server to which you are mapping lulu. The value is case-sensitive, unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

REMOTE_PASSWORD 'raiders'
   Specifies the password that is associated with 'lulu'. The value is case-sensitive, unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

DB2 special register USER

You can use the DB2 special register USER to map the authorization ID of the person who is issuing the CREATE USER MAPPING statement to the data source authorization ID that is specified in the REMOTE_AUTHID user option.

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

CREATE USER MAPPING FOR USER SERVER Nwind
   OPTIONS (REMOTE_AUTHID 'lulu', REMOTE_PASSWORD 'raiders');
Chapter 13. Configuring access to Oracle data sources

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

Before you begin
- The Oracle client software must be installed and configured on the server that acts as the federated server.
- Federation must be installed on a server that acts as the federated server.
- Check the setup of the federated server.
- Check the federated parameter to ensure that federation is enabled.

Procedure

To add Oracle data sources to a federated system:
1. Set the Oracle environment variables.
2. Set up and test the Oracle client configuration file.
3. Register the Oracle wrapper.
4. Register the server definitions for an Oracle data source.
5. Create the user mappings.
6. Test the connection to the Oracle server.
7. Register the nicknames.

Setting the Oracle environment variables

The Oracle environment variables must be set in the db2dj.ini file on the federated server.

Restrictions

Review the restrictions for the db2dj.ini file.

The db2dj.ini file contains configuration information about the Oracle client software that is installed on the federated server.

There are required and optional environment variables for Oracle data sources.

If you installed the Oracle client software before you installed the Oracle wrapper, the required Oracle environment variables are set in the db2dj.ini file.

If you did not install the Oracle client software before you installed the Oracle wrapper, or if you want to set any of the optional environment variables you must set the environment variables by using the steps in this task.

Procedure

To set the Oracle environment variables:
1. Use one of the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Automatically set the environment variables.</td>
<td>Run the WebSphere Federation Server installation wizard. Follow the instructions in the wizard. Important: Set the required environment variables by running the installation wizard. The optional environment variables must be set manually.</td>
</tr>
<tr>
<td>Manually set the environment variables.</td>
<td>Edit the db2dj.ini file:</td>
</tr>
<tr>
<td></td>
<td>The db2dj.ini file is located in the directory that the DB2 registry variable DB2_DJ_INI specifies. When the DB2_DJ_INI variable is not set, the db2dj.ini file is in one of the following default paths depending on the operating system:</td>
</tr>
<tr>
<td></td>
<td>• On Linux and UNIX: instancehome/sqlib/config/db2dj.ini. instancehome  The home directory of the instance owner.</td>
</tr>
<tr>
<td></td>
<td>• On Windows: %DB2PATH%\config\db2dj.ini %DB2PATH% The directory where the DB2 database system is installed, for example, C:\Program Files\IBM\sqlib.</td>
</tr>
<tr>
<td></td>
<td>If the file does not exist, you can create a file with the name db2dj.ini by using any text editor. In the db2dj.ini file, you must specify the fully qualified path in the value of the environment variables; otherwise you will encounter errors.</td>
</tr>
</tbody>
</table>

2. On federated servers that run UNIX, add the Oracle environment variable to the .profile file of the DB2 instance. For example:

   export ORACLE_HOME=oracle_home_directory
   export PATH=$ORACLE_HOME/bin:$PATH

   Where oracle_home_directory is the directory where the Oracle client software is installed.

3. On federated servers that run Linux or UNIX, execute the DB2 instance .profile file by entering:

   . $HOME/.profile

4. To ensure that the environment variables are set on the federated server, recycle the DB2 instance with these commands:

   db2stop
   db2start

After you complete this task, you can set up and test the Oracle client.

**Oracle environment variables**

There are required and optional environment variables for Oracle data sources. These variables are set in the db2dj.ini file.

The following environment variables are valid for Oracle:

- ORACLE_HOME
- ORACLE_BASE (optional)
- ORA_NLS (optional)
• NLS_LANG (optional)
• TNS_ADMIN (optional)

Variable descriptions

ORACLE_HOME

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

If an individual user of the federated instance sets the ORACLE_HOME environment variable locally, the federated instance does not use that setting. The federated instance uses only the value of ORACLE_HOME that is set in the db2dj.ini file.

ORACLE_BASE

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

For example:

ORACLE_BASE=oracle_root_directory

ORA_NLS

If multiple versions of Oracle are running on your system, you must ensure that:
• The appropriate ORA_NLS environment variable is set
• The corresponding NLS data files for the versions that you are using are available

The location-specific data is stored in a directory that is specified by the ORA_NLS environment variable. Each version of Oracle has a different ORA_NLS data directory.

Table 77. ORA_NLS environment variable by version

<table>
<thead>
<tr>
<th>Oracle version</th>
<th>Environment variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.x, 9.x</td>
<td>ORA_NLS33</td>
</tr>
<tr>
<td>10.x</td>
<td>ORA_NLS10</td>
</tr>
</tbody>
</table>

For example, on federated servers that run UNIX that access Oracle 8.1 data sources, the ORA_NLS33 environment variable setting is:

ORA_NLS33=oracle_home_directory/ocommon/nls/admin/<data>

NLS_LANG

The NLS_LANG environment variable is a code page environment variable. Refer to the Oracle NLS documentation for information about setting this variable.

TNS_ADMIN

On federated servers that run Windows

The Oracle client looks for the tnsnames.ora file in the
%ORACLE_HOME%\NETWORK\ADMIN directory, where
%ORACLE_HOME% is defined in the db2dj.ini file. If the tnsnames.ora
file is not in the %ORACLE_HOME%\NETWORK\ADMIN directory, you must set the TNS_ADMIN environment variable in the db2dj.ini file on the federated server. You set the environment variable in the db2dj.ini file to the path where the tnsnames.ora file is located.

On federated servers that run AIX or Linux
The Oracle client looks for the tnsnames.ora file in the /etc directory. If the tnsnames.ora file is not in the /etc directory, then the Oracle client looks for the tnsnames.ora file in the $ORACLE_HOME/network/admin directory, where $ORACLE_HOME is defined in db2dj.ini file. If the tnsnames.ora file is not in the $ORACLE_HOME/network/admin directory, you must set the TNS_ADMIN environment variable on the federated server. You set the environment variable in the db2dj.ini file to the path where the tnsnames.ora file is located.

For example, if the tnsnames.ora file is in the /home/oracle directory, you set the environment variable to:

TNS_ADMIN=/home/oracle

On federated servers that run Solaris
The Oracle client looks for the tnsnames.ora file in the /var/opt/oracle directory. If the tnsnames.ora file is not in the /var/opt/oracle directory, then the Oracle client looks for the tnsnames.ora file in the $ORACLE_HOME/network/admin directory, where $ORACLE_HOME is defined in db2dj.ini file. If the tnsnames.ora file is not in the $ORACLE_HOME/network/admin directory, you must set the TNS_ADMIN environment variable. You set the variable in the db2dj.ini file to the path where the tnsnames.ora file is located.

For example, if the tnsnames.ora file is in the /home/oracle directory, you set the environment variable to:

TNS_ADMIN=/home/oracle

Oracle code page conversion
Each time that the Oracle wrapper connects to an Oracle data source, the wrapper determines which code page value to use for that connection. You can specify that the Oracle wrapper set the code page value or you can designate a code page by setting the NLS_LANG environment variable.

The environment variables that specify Oracle code page conversion are set in the db2dj.ini file on your federated server.

If the NLS_LANG environment variable is set in the db2dj.ini file on the federated server, the wrapper uses the code page value in the db2dj.ini file.

If the NLS_LANG environment variable is not set in the db2dj.ini file on the federated server, the wrapper determines the territory and the code page of the federated database. The wrapper sets the NLS_LANG environment variable to the closest matching Oracle locale. If there is no closely matching locale, the NLS_LANG environment variable is set to American_America.US7ASCII.

See the documentation that accompanies your Oracle software for a list of valid locales.
Example of Chinese code page GB 18030:

If you access an Oracle data source that contains data that is encoded with the Chinese code page GB 18030, and your federated database uses the UTF-8 code page, the Oracle wrapper sets the Oracle NLS_LANG environment variable to:

NLS_LANG=Simplified Chinese_China.UTF8

This setting is correct if you are using an Oracle 8 client, but if you are using the Oracle Version 9i (or later) client, you must set the NLS_LANG environment variable to override the default setting of the Oracle wrapper. Set the NLS_LANG environment variable to Simplified Chinese_China.AL32UTF8, so that the Oracle 9i client translates the GB 18030 data into Unicode correctly.

For example:

NLS_LANG=Simplified Chinese_China.AL32UTF8

Setting up and testing the Oracle client configuration file

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

About this task

The client configuration file specifies the location of each Oracle database server and type of connection (protocol) for the database server.

The default name for the Oracle client configuration file is tnsnames.ora.

The default location of the client configuration file depends on the operating system that is used by the federated system:

- On Linux and UNIX systems, the default location of the file is $ORACLE_HOME/network/admin.
- On Windows systems, the default location of the file is %ORACLE_HOME%\NETWORK\ADMIN.

Procedure

To set up and test the Oracle client configuration file:

1. Create the tnsnames.ora using the Oracle NET8/NET Configuration utility that comes with the Oracle client software.

   Within the tnsnames.ora file, the SID (or SERVICE_NAME) is the name of the Oracle instance, and the HOST is the host name where the Oracle server is located.

2. If you want to place the tnsnames.ora file in a path other than the default search path, set the TNS_ADMIN environment variable to specify the file location:

   a. Edit the db2dj.ini file in the sqllib/cfg directory, and set the TNS_ADMIN environment variable. For example:

      TNS_ADMIN=x:/path/

   b. Issue the following commands to recycle the DB2 instance and ensure that the environment variable is set in the program:

      db2stop
      db2start
3. Test the connection to ensure that the client software can connect to the Oracle server. Use the Oracle sqlplus utility to test the connection.

After you complete this task, you can register the Oracle wrapper.

**Registering the Oracle wrapper**

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

**Procedure**

To register an Oracle wrapper:

Use one of the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Issue the CREATE WRAPPER statement and specify the default name for the Oracle wrapper.</td>
<td>For example: CREATE WRAPPER NET8 ; Remember: When you register the wrapper by using the default name, NET8, the federated server automatically uses the appropriate Oracle wrapper library for the operating system that your federated server is running on.</td>
</tr>
</tbody>
</table>

If you do not use the default wrapper name when you register a wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

For example, to register a wrapper with the name oracle_wrapper on the federated server that uses AIX, issue the following statement:

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

The wrapper library file that you specify depends on the operating system of the federated server.

After you complete this task, you can register the server definitions.

**Oracle wrapper library files**

The Oracle wrapper library files are added to the federated server when you install the wrapper.

When you install the Oracle wrapper, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper
library files that are added to the following directory path are libdb2net8.a, libdb2net8F.a, and libdb2net8U.a. The default wrapper library file is libdb2net8.a. The other wrapper library files are used internally by the Oracle wrapper.

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

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Library file names</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/install_path/lib32/</td>
<td>libdb2net8.a</td>
</tr>
<tr>
<td></td>
<td>/usr/opt/install_path/lib64/</td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/IBM/db2/install_path/lib32</td>
<td>libdb2net8.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/install_path/lib64</td>
<td></td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/IBM/db2/install_path/lib32</td>
<td>libdb2net8.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/install_path/lib64</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>%DB2PATH%\bin</td>
<td>db2net8.dll</td>
</tr>
</tbody>
</table>

install_path is the directory path where the federated server is installed on UNIX or Linux.

Registering the server definitions for an Oracle data source

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

Procedure

To register a server definition for an Oracle data source:
1. Locate the node name in the Oracle tnsnames.ora file. Example tnsnames.ora file:
   ```
   paris_node =
   (DESCRIPTION =
   (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = somehost)(PORT = 1521)))
   (CONNECT_DATA = (SERVICE_NAME = ora9i.seel))
   ```
   In this example, the node value to use in the CREATE SERVER statement is paris_node.
   Although the node_name is specified as an option in the CREATE SERVER SQL statement, it is required for Oracle data sources.
2. To register the server definitions for an Oracle data source, use one of the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
</tbody>
</table>
Method | Description
--- | ---
**Issue the CREATE SERVER statement.** | For example:
CREATE SERVER server_name TYPE oracle
VERSION 8.1.7 WRAPPER net8
OPTIONS (NODE 'node_name')
After the server definition is created, use the ALTER SERVER statement to add or drop server options.

The next task in this sequence of tasks is creating the user mappings for an Oracle data source.

**CREATE SERVER statement - Examples for the Oracle wrapper**

Use the CREATE SERVER statement to register server definitions for the Oracle wrapper. This topic provides a complete example with the required parameters, and an example with additional server options.

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

CREATE SERVER oraserver TYPE oracle VERSION 8.1.7 WRAPPER wrapper_name
OPTIONS (NODE 'paris_node') ;

*oraserver*

A name that you assign to the Oracle database server. Duplicate server definition names are not allowed.

*TYPE oracle*

Specifies the type of data source server to which you are configuring access. For the Oracle wrapper, the server type must be oracle.

*VERSION 8.1.7*

The version of Oracle database server that you want to access.

*WRAPPER wrapper_name*

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

*NODE 'paris_node'*

The name of the node where the Oracle database server resides. Obtain the node name from the tnsnames.ora file. This value is case sensitive.

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

**Server options**

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

The federated server assumes that all of the Oracle VARCHAR columns contain trailing blanks. If you are certain that all of the VARCHAR columns in the Oracle database do not contain trailing blanks, you can set a server option to specify that the data source use a non-blank padded VARCHAR comparison semantic.
The following example shows an Oracle server definition with the VARCHAR_NO_TRAILING_BLanks server option:

```sql
CREATE SERVER oraserver TYPE oracle VERSION 8.1.7 WRAPPER wrapper_name
  OPTIONS (NODE 'paris_node', VARCHAR_NO_TRAILING_BLANKS 'Y')
```

Use the VARCHAR_NO_TRAILING_BLANKS server option when none of the columns contains trailing blanks. If only some of the VARCHAR columns contain trailing blanks, you can set an option on those columns with the ALTER NICKNAME statement.

## Creating the user mappings for an Oracle data source

To access an Oracle server, the federated server establishes a connection to the Oracle server by using a user ID and password that are valid for that data source.

### Restrictions

The user ID at the Oracle data source must be created by using the Oracle create user command with the identified by clause, instead of the identified externally clause.

### Procedure

To create the user mappings for an Oracle data source:

Issue a CREATE USER MAPPING statement. For example:

```sql
CREATE USER MAPPING FOR local_userID SERVER server_definition_name
  OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password')
```

After you complete this task, test the connection to the Oracle server.

**CREATE USER MAPPING statement - Examples for the Oracle wrapper**

Use the CREATE USER MAPPING statement to map a federated authorization ID to a remote Oracle user ID and password. This topic includes a complete example with the required parameters, and an example that shows you how to use the DB2 special register USER with the CREATE USER MAPPING statement.

The following example shows how to map a federated authorization ID to an Oracle user ID and password:

```sql
CREATE USER MAPPING FOR robert SERVER oraserver
  OPTIONS (REMOTE_AUTHID 'rob', REMOTE_PASSWORD 'then4now')
```

*robert* specifies the authorization ID that you are mapping to a remote user ID and password, which are defined at the Oracle server.

*SERVER oraserver* specifies the server definition name that you registered in the CREATE SERVER statement for the Oracle server.

*REMOTE_AUTHID 'rob'* specifies the remote user ID to which you are mapping *robert*. The value is case-sensitive, unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.
REMOTE_PASSWORD 'then4now'
Specifies the remote password that is associated with rob. The value is case-sensitive, unless you set the FOLD_PW server option to ‘U’ or ‘L’ in the CREATE SERVER statement.

DB2 special register USER

You can use the DB2 special register USER to map the authorization ID of the person who is issuing the CREATE USER MAPPING statement to the data source authorization ID that is specified in the REMOTE_AUTHID user option.

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

```
CREATE USER MAPPING FOR USER SERVER oraserver
  OPTIONS (REMOTE_AUTHID 'rob', REMOTE_PASSWORD 'then4now') ;
```

Testing the connection to the Oracle server

Test the connection to the Oracle server to determine if the federated server is properly configured to access Oracle data sources.

About this task

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

Procedure

To test the connection to the Oracle server:

Open a pass-through session and issue a SELECT statement on the Oracle system tables. If the SELECT statement returns a count, your server definition and your user mapping are set up properly.

For example:

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

If the SELECT statement returns an error, you should troubleshoot the connection errors.

After you complete this task, you can register the nicknames for the Oracle tables and views.

Troubleshooting connectivity problems with Oracle data sources

The most common problem that you might encounter when you configure the federated server to access Oracle data sources is connectivity.

Symptom

If you are not able to connect to an Oracle data source from a federated server, you might need to update the TCP/IP hosts file.
Cause

This problem can be caused by an out of date TCP/IP hosts file.

Resolving the problem

For each host in the DESCRIPTION section of the tnsnames.ora file, you might need to update the TCP/IP hosts file. Whether you update this file depends on how TCP/IP is configured on your network. Part of the network must translate the remote host name that is specified in the DESCRIPTION section in the tnsnames.ora file to an address.

If your network has a name server that recognizes the host name, you do not need to update the TCP/IP hosts file. If your network does not have a name server that recognizes the host name, you need to add an entry in the TCP/IP hosts file for the remote host.

The location of the TCP/IP hosts file depends on the operating system that is running on the federated server:

- **On federated servers that run Linux or UNIX**
  The hosts file is in the /etc/hosts directory.

- **On federated servers that run Windows**
  The hosts file is in the x:\winnt\system32\drivers\etc\hosts directory.

Registering nicknames for Oracle tables and views

For each Oracle server definition that you register, you must register a nickname for each table or view that you want to access. Use these nicknames, instead of the names of the data source objects, when you query the Oracle servers.

Before you begin

Update the statistics at the Oracle data source before you register a nickname. The federated database relies on the data source catalog statistics to optimize query processing. Use the data source command that is equivalent to the DB2 RUNSTATS command to update the data source statistics.

For data source objects that use Oracle Label Security, the nickname data cannot be cached. You can turn caching on or off using the ALTER NICKNAME statement.

Procedure

To register a nickname for an Oracle table or view, use one of the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Issue the CREATE NICKNAME statement.</td>
<td>For example: CREATE NICKNAME nickname FOR server_definition_name.&quot;remote_schema&quot;.&quot;remote.table&quot; ;</td>
</tr>
</tbody>
</table>
When you create the nickname, the federated server queries the data source catalog by using the nickname. This query tests the connection to the data source table or view. If the connection does not work, you receive an error message.

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

**CREATE NICKNAME statement - Examples for the Oracle wrapper**

Use the CREATE NICKNAME statement to register a nickname for an Oracle table or view that you want to access. This topic provides a complete example with the required parameters.

The following example shows you how to register a nickname for an Oracle table or view using the CREATE NICKNAME statement.

```
CREATE NICKNAME PARISINV FOR oraserver."vinnie"."inventory" ;
```

**PARISINV**

A unique nickname that is used to identify the Oracle table or view. The nickname consists of the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authorization ID of the user who registers the nickname.

**oraserver."vinnie"."inventory"**

A three-part identifier for the remote object:

- `oraserver` is the server definition name that you assigned to the Oracle database server in the CREATE SERVER statement.
- `vinnie` is the user ID of the owner to which the table or view belongs.
- `inventory` is the name of the remote table or view that you want to access.

The federated server changes the names of the Oracle schemas and tables to uppercase unless you enclose the names in quotation marks.
Chapter 14. Configuring access to scripts as data sources

To configure a federated system to access scripts as data sources, you must register custom functions, a wrapper, a server definition, and nicknames for the scripts.

**Before you begin**

- Federation must be installed on a server that acts as the federated server.
- A federated database must exist on the federated server.

**About this task**

You can configure a federated server to access data through scripts by issuing SQL statements on the DB2 command line.

**Procedure**

To add scripts as data sources to a federated server:

1. Identify or write a script.
2. **Register the custom function**
3. **Configure the script daemon**
4. **Start the script daemon**
5. **Register the script wrapper**
6. **Register the server definition for a data source that is accessed by a script**
7. **Register nicknames for data sources**

**Script wrapper overview**

You can use scripts, such as Perl scripts, to access information in databases or to generate data. You can integrate that data with data from other federated data sources by using the script wrapper.

You might have existing scripts that return data from data sources such as life sciences data banks, or that generate data on their own. The script wrapper enables the use of scripts as if they are federated data sources. The script wrapper enables access to data that can then be integrated by using a federated system. The scripts must return results in XML.

Script wrapper nicknames can include input and output columns. These nicknames use function templates in predicates to pass input values to the script. Output data from the script is represented as XML in a hierarchical form, which can then map to nicknames by using primary and foreign keys.

The script wrapper is a read-only wrapper. The script wrapper cannot write data to a data source.

In the following diagram, data flows from a script through the script wrapper and script daemon to the federated database, where the data can be integrated with data from other sources and viewed with the federated client. Optionally, the federated system can access the script through a proxy server and firewall.
A script is invoked from the directory that contains the script daemon. If the script retrieves data from its own data file and cannot locate the data file, the script might include relative paths. Use absolute paths in scripts.

**Adding scripts as data sources to a federated system**

To configure a federated server to access scripts as data sources, you must configure the script daemon, a wrapper, a server definition, and nicknames for the scripts.

**Registering the custom function for the script**

You must register the script custom function WSSCRIPT.ARGS before you register the script wrapper.
About this task

You must register the custom function on each federated database instance where the script wrapper is installed.

The custom function for the script wrapper must be registered with the schema name WSSCRIPT.

You must include specific keywords when you register the custom function for the script wrapper. Include the AS TEMPLATE, DETERMINISTIC, and NO EXTERNAL ACTION keywords in the CREATE FUNCTION statement.

The federated environment uses two query engines. For the script wrapper, these query engines are the federated database query engine and the script wrapper query engine. You can specify that predicates get pushed down to the script wrapper engine by using the script wrapper custom functions in the WHERE clause of your SELECT statement.

The create_function_mappings.ddl file in the sqlib/samples/lifesci/script directory on the federated server specifies the data types for the custom function.

Procedure

To register the script custom function:

Run the create_function_mappings.ddl file on each federated database instance where the script wrapper is installed.

The following example shows the syntax for the WSSCRIPT.ARGS function:

```
CREATE FUNCTION WSSCRIPT.ARGS (input_column_data_type(), input_column_data_type())
    RETURNS INTEGER AS TEMPLATE
    DETERMINISTIC NO EXTERNAL ACTION;
```

Data types for the custom function for the script wrapper

You must specify the data type of the input column twice in each custom function.

Register a separate WSSCRIPT custom function for each valid data type for the column_name argument. The WSSCRIPT custom function has the following data types:

- VARCHAR
- INTEGER
- CLOB
- DOUBLE
- DATE

Both parameters to the custom function must be of the same data type, which is the data type of the corresponding input column. When used in a query predicate, the first parameter is the name of the switch input column. The second column is the value to be passed to the script for this switch.

Configuring the script daemon

The script wrapper requires a script daemon that listens for script job requests from the script wrapper. The script daemon must be configured before you register the script wrapper.
Before you begin

The script daemon has the following prerequisites:
- Has write access to a directory in which the daemon can write temporary files.
- Runs on a server that you can access through TCP/IP from your federated system. This server can be the same server that operates as the federated server or a separate script server.
- Requires a configuration file that must be on the same server as the script daemon.
- Runs separately from the script wrapper and the federated database.

Procedure

To configure the script daemon:

1. Ensure that the executable files for the script daemon are on the correct server. You might need to copy the script daemon executable files to another server.
   During the installation of IBM WebSphere Federation Server, the script daemon executable files are installed on the federated server. The name and location of the file is as follows:
   
   **UNIX** db2script_daemon is installed in the $DB2PATH/bin directory. $DB2PATH is the directory in which the federated server is installed.
   
   **Windows**
   
   db2script_daemon.exe is installed in the %DB2PATH%\bin directory. %DB2PATH% is the directory in which the federated server is installed, usually C\:SQLLIB\bin.
   
   If you use a separate script server, copy the script daemon executable and configuration files from the federated server to the script server. The script daemon executable files can run in any directory on the script server that does not contain spaces in the names in the directory path.

2. Ensure that the configuration file for the script daemon is on the correct server.
   
   During the installation of the federated system, a sample configuration file for the script daemon is installed on the federated server. The name of the sample configuration file is SCRIPT_DAEMON.config. The location of the file is as follows:
   
   **UNIX** The daemon configuration file is installed in the $DB2PATH/bin directory.
   
   **Windows**
   
   The daemon configuration file is installed in the %DB2PATH%\bin directory.
   
   By default, the daemon looks for the configuration file in the working directory from which the daemon is started. You can copy the configuration file to another location. If you use a script server, copy the daemon configuration file from the directory on the federated server to a directory on the script server. You can copy the daemon configuration file to any directory on the script server that the daemon can access.

3. Edit the sample configuration file for the script daemon.
   
   a. Rename the configuration file so that you can use the sample file again.
b. Ensure that the first line in the configuration file is an equal sign (=). If the
   equal sign is missing, the daemon does not start. An error message will
   indicate that the DAEMON_PORT was not specified.

c. Ensure that the last line in the configuration file ends with a new line.
The sample configuration file that is provided with the federated system
ends with a new line character. If the last line does not end with a new line
character, you receive an error message when you try to run your first script
query that uses the data source that is listed on the last line.

d. Ensure that there are no extra spaces after directory paths or at the end of
the configuration file.

e. Specify the following options in the configuration file. For options that
require paths, you can specify relative paths. Relative paths are relative to
the directory from which the daemon process is started.

```
DAEMON_PORT=port_number
   The network port on which the daemon listens for script job
   requests that are submitted by the wrapper. The default value is
   4099.

MAX_PENDING_REQUESTS=number_of_requests
   The maximum number of script job requests that can be blocking on
   the daemon at any one time. This number does not represent the
   number of script jobs that are running concurrently, only the
   number of job requests that can block at one time. Set this value to
   a number greater than five. The script daemon does not restrict the
   number of script jobs that can run concurrently.

DAEMON_LOGFILE_DIR=dir
   The directory in which the daemon creates its log file. This file
   contains status and error information that is generated by the script
   daemon.

SCRIPT_OUT_DIR_PATH=path
   The directory in which the daemon creates the temporary file to
   store the script output data. The daemon reads data from this file
   and passes the data back to the wrapper through the network
   connection. After the data is passed to the wrapper, the daemon
   cleans up the temporary file

script specification entry=entry
   A list of entries that specifies the name and location of the scripts
   that can be invoked by the script wrapper. The entry has this
   format:
   script_name=fully-qualified_script_path

   The following examples apply to the designated operating system:

   UNIX   For example, to specify a script that accesses an Oracle data
           source, add the following line to the daemon configuration
           file:
           oracle=/dsk/1/data/oracle

   Windows
           For example, to specify a script that accesses an Oracle data
           source, add the following line to the daemon configuration
           file:
           oracle=c:\data\oracle.a
```
The following example shows a SCRIPT_DAEMON.cfg file for four scripts:

```
= 
DAEMON_PORT=4099 
MAX_PENDING_REQUESTS=10 
DAEMON_LOGFILE_DIR=./ 
SCRIPT_OUT_DIR_PATH=./ 
fee=/home/user_id/fee 
fie=/home/user_id/fie 
foe=/home/user_id/foe 
fum=/home/user_id/fum
```

The sample configuration file for the script daemon provides an example of configuring the script daemon.

**Starting the script daemon**

Before you can access data sources with scripts, you must start the script daemon.

**Before you begin**

You must have write access to all the paths that are listed in the daemon configuration file for the DAEMON_LOGFILE_DIR and SCRIPT_OUT_DIR_PATH options.

**About this task**

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

**Procedure**

To start the script daemon:

1. Open the directory where the daemon executable file is located.
2. Issue the `db2script_daemon` command to run the executable files, including appropriate options.

The following example includes options:

```
db2script_daemon -a action -c config_file -d debug_level -u user_id -p password
```

**db2script_daemon command - options and examples**

The `db2script_daemon` command starts the script daemon. You can start the script daemon with several options.

The `db2script_daemon` command can be used on either UNIX or Windows servers. Some of the options listed in the syntax can be used only on Windows servers.

The options that are specified with the start action affect only the current instance of the daemon and override the values that are specified with the install action.

**Options - db2script_daemon command**

The `db2script_daemon` command has the following options:

- `a action` *(Windows only)*
  
  Performs the specified activity. The valid actions are status, install, start, stop, and remove.
-c config_file
  Instructs the daemon service to use the specified configuration file. If you do not specify the configuration file, the daemon searches for the SCRIPT_DAEMON.config file in the directory where the daemon executable files are installed. You can use this option with the install and start actions.

-d [1 | 2 | 3]
  Sets the debugging level for the daemon service to the specified value. A value of 1 turns on logging, 2 traces all commands, and 3 turns on logging, traces all commands, and saves all temporary files to capture XML output. You can use this option with the install and start actions.

-u user_id (Windows only)
  Sets the daemon service to run under the specified user ID. You can use this option with the install action.

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

Examples - db2script_daemon command

The following examples show how to use the script daemon options.

Start the daemon
  To start the daemon on UNIX, issue the following command:
  
  db2script_daemon

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

  To install and start the daemon on Windows, issue the following commands:
  
  db2script_daemon -a install
  db2script_daemon -a start

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

  In this example, the daemon configuration information is in a file called SCRIPT_D.config in the subdirectory cfg on a UNIX server. Issue the following command:
  
  db2script_daemon -c cfg/SCRIPT_D.config

Specify the debugging level
  If you want to start the daemon with debugging turned on with a debugging level of 2, issue the following commands:
  
  db2blast_daemon -a install -d 2
  db2blast_daemon -a start

Check the status of the daemon (Windows)
  To check the status of the daemon on a Windows server, issue the following command:
This command assumes that the daemon configuration file is in the same directory as the executable file.

**Stop the daemon**

To stop the daemon on UNIX, list the process ID of the daemon by issuing the following command:
```
ps -ef | grep db2script
```

Then use the process ID to stop the daemon by issuing the following command:
```
kill process_ID
```

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

To stop the daemon on Windows, issue the following command:
```
db2script_daemon -a stop
```

**Remove the daemon**

You can remove the script daemon when you no longer want to use the script wrapper.

To remove the daemon, issue the following command:
```
db2script_daemon -a remove
```

**Registering the script wrapper**

You must register the script wrapper to access data sources with scripts. The script wrapper is implemented as a library file.

**Procedure**

To register the script wrapper:

Issue the CREATE WRAPPER statement, specifying a name for the script wrapper and the name of the wrapper library file.

For example, to register a wrapper with the name script_wrapper on a federated server that uses AIX, issue the following statement:
```
CREATE WRAPPER script_wrapper LIBRARY 'libdb2lsscript.a';
```

The name of the wrapper library file that you specify depends on the operating system of the federated server.

There are no script-wrapper specific options for the script wrapper CREATE WRAPPER statement. The wrapper runs unfenced by default.

**Script wrapper library file**

To register the script wrapper, specify the script wrapper library file for the operating system of the federated server.

When you install federation, a script wrapper library file is added to the default directory path.
The default directory paths and default wrapper library file names are listed in the following table.

**Table 79. Script wrapper library locations and file names**

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Wrapper library file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/install_path/lib</td>
<td>libdb2lsscript.a</td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/IBM/db2/install_path/lib</td>
<td>libdb2lsscript.so</td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/IBM/db2/install_path/lib</td>
<td>libdb2lsscript.so</td>
</tr>
<tr>
<td>Windows</td>
<td>%DB2PATH%\bin</td>
<td>db2lsscript.dll</td>
</tr>
</tbody>
</table>

- *install_path* is the directory path where federation is installed on UNIX or Linux.
- %DB2PATH% is the environment variable that is used to specify the directory path where federation is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

**Registering the server definition for a script as a data source (DB2 command line)**

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

**Procedure**

To register a server definition for a script:

Issue the CREATE SERVER statement.

For example:

```
CREATE SERVER script_server WRAPPER script_wrapper
  OPTIONS (NODE 'myserver.example.com', DAEMON_PORT '4099');
```

The NODE and DAEMON_PORT server options are required for scripts as data sources.

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

**CREATE SERVER statement - examples for the script wrapper**

The examples show the use of required options and additional server options.

**Required options example**

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

```
CREATE SERVER server1_scriptn WRAPPER script_wrapper
  OPTIONS(NODE 'big_rs.company.com');
```

- **server1_scriptn**
  A name that you assign to the script server. Duplicate server definition names are not allowed.

- **WRAPPER script_wrapper**
  The wrapper name.

- **NODE 'big_rs.company.com'**
  Host name of the system on which the script daemon process is running. This value is case sensitive.
Although the name of the node is specified as an option in the CREATE SERVER statement, it is required for the script wrapper.

Optional option example

The following example shows an additional server option that you can specify when you register a server definition for the script wrapper:

```
CREATE SERVER server1_scriptn  WRAPPER script_wrapper
   OPTIONS(NODE 'big_rs.company.com', DAEMON_PORT '4088');
```

**DAEMON_PORT '4088'**

Specifies the number of the port that the daemon listens on for script job requests. The port number must be the same number that you specified in the DAEMON_PORT option of the daemon configuration file. The default port number is 4099.

Registering nicknames for scripts (DB2 command line)

You must register a separate nickname for each script. Use these nicknames when you query the data source that is accessed by a script.

About this task

The script wrapper associates XML data to nicknames. Parent and child nicknames correspond to the root and nested elements in an XML document. The parent and child nicknames are connected by primary and foreign keys that are specified in the CREATE NICKNAME statement. Each nickname is defined by XPath expressions that identify the XML data elements and specify how to extract column values from each element.

The data source is specified by the CREATE NICKNAME statement and associated with a script name with the DATASOURCE nickname option. You must create a column for each input argument to be passed to the script. Use input column options to control the syntax for switches on the command line. The value for each switch must be included in the query predicate by using the ARGS custom function at run time.

Simple scripts that do not require command-line arguments do not need input columns.

**Procedure**

To register a nickname for a script:

Issue the CREATE NICKNAME statement. Nicknames can be up to 128 characters long.

For example:

```
CREATE NICKNAME nickname
   
   column_name data_type OPTIONS ('nickname_column_options'),
   column_name data_type OPTIONS ('nickname_column_options'),
   column_name data_type OPTIONS ('nickname_column_options')
   
   FOR SERVER server_definition_name
   OPTIONS (nickname_options);
```
Issue the statement for each script that you want to create a nickname for.

**CREATE NICKNAME statement - examples for the script wrapper**

The examples show you how to use the CREATE NICKNAME statement to register nicknames for the script wrapper.

The following example creates a parent nickname for the XML data that is returned from a script that is named fee:

```sql
CREATE NICKNAME customers
(
    argle double OPTIONS(SWITCH '-argle', POSITION 1, DEFAULT 1.0 ),
    argfile CLOB() OPTIONS(SWITCH '-file', INPUT MODE 'FILE_INPUT', POSITION 2),
    argpos varchar() OPTIONS(SWITCH ' ', POSITION 3),
    id VARCHAR(5) OPTIONS(XPATH './@id'),
    name VARCHAR(16) OPTIONS(XPATH './name'),
    address VARCHAR(30) OPTIONS(XPATH './address/@street'),
    cid VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(PRIMARY_KEY 'YES'))
FOR SERVER script_server
OPTIONS(DATASOURCE 'fee',
       XPATH '/doc/customer', STREAMING 'YES');
```

The following example creates a nickname that is called orders. The nickname orders is a child of the nickname that is called customers, which is created in the previous example:

```sql
CREATE NICKNAME orders
(
    amount INTEGER OPTIONS(XPATH './amount'),
    date VARCHAR(10) OPTIONS(XPATH './date'),
    oid VARCHAR(16) OPTIONS(PRIMARY_KEY 'YES'),
    cid VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(FOREIGN_KEY 'CUSTOMERS'))
FOR SERVER script_server
OPTIONS(XPATH './order');
```

**Script wrapper nickname options**

You can specify options when you create a nickname for a script. Only the root nickname can include input columns.

**Nickname options**

The following list describes the nickname options:

**DATASOURCE**

The name of the script that will be invoked. The script must be listed in the script daemon configuration file. This option is required for data sources in the parent nickname. This option applies only to the root nickname.

**NAMESPACE**

A comma-separated list of name=value pairs that the wrapper uses to resolve namespace prefixes in the nickname XPath expression.

**TIMEOUT**

The maximum time, in minutes, that the script wrapper waits for results from the script daemon. The default value is 60 minutes. This option applies only to the root nickname.

**VALIDATE**

Specifies whether the XML source document must be validated before the XML data is extracted. If this option is set to YES, the DB2 database system verifies
that the structure of the source document conforms to an XML schema or to a
document type definition (DTD). This option is accepted only for columns of
the root nickname (the nickname that identifies the elements at the top level of
the XML document). The default value is NO.

The XML source document is not validated if the script wrapper cannot locate
the XML schema file or DTD file (.xsd or .dtd). The DB2 database system does
not issue an error message if the validation does not occur. Ensure that the
XML schema file or DTD file is in the location that is specified in the XML
source document. Do not set the VALIDATE parameter to YES if you set the
STREAMING parameter to YES.

STREAMING
Specifies whether the XML source document is separated into logical fragments
that correspond to the node that matches the XPath expression of the
nickname. The script wrapper then processes the XML source data fragment by
fragment, reducing total memory use. This option is accepted only for columns
of the root nickname (the nickname that identifies the elements at the top level of
the XML document). The default streaming value is NO. Do not set the
STREAMING parameter to YES if you set the VALIDATE parameter to YES.

XPATH
An XPath expression that identifies the XML element that represents individual
tuples in the data source. The wrapper evaluates the XPATH nickname option
for a child nickname in the context of the path that is specified by the XPATH
nickname option of the parent nickname. This XPath expression is used as a
context for evaluating column values that are identified by the XPATH
nickname column options.

Nickname column options

The nickname column options are described in the following:

DEFAULT
The default value for an input column. This option applies only to input
columns.

The default value is used if no value is provided by the SQL query. This option
is not required.

FOREIGN_KEY
Indicates that this nickname is a child nickname and specifies the
 corresponding parent nickname.

A nickname can have at most one FOREIGN_KEY column option. The value
for this option is case-sensitive. The column that is designated by the
FOREIGN_KEY option contains a key that is generated by the wrapper. The
value of the column cannot be retrieved in a SELECT query, and the XPATH
option must not be specified. The column can only be used to join parent
nicknames and child nicknames. A CREATE NICKNAME statement with a
FOREIGN_KEY option fails if the parent nickname has a different schema
name. Unless the nickname that is referred to in a FOREIGN_KEY clause is
explicitly defined to be lowercase or mixed case by enclosing it in quotation
marks under the corresponding CREATE NICKNAME statement, then you
must specify the nickname in uppercase when you refer to this nickname in
the FOREIGN_KEY clause.

Foreign key columns must be designated as FOR BIT DATA and NOT NULL.
INPUT_MODE
Specifies the input mode for a column. Valid values are CONFIG or FILE_INPUT. The wrapper passes the specified value to the script daemon.

CONFIG
The value is treated as a configurable parameter.

FILE_INPUT
A file is created that stores the value and the file name is passed as the command line argument.

POSITION
An integer value for positional parameters. The position sequence order starts at 1. This option applies only to input columns.

If the positional value is set to an integer, then this input must be in this position in the command line. If this option is set, the switch is inserted into the appropriate location when the query is run. If POSITION is set to -1, the option is added as the last command line option. For example, if a column value must be at the end of the command line and there is no SWITCH option, setting the value of POSITION to -1 adds the value at the end of the command line. POSITION integer values cannot be duplicated in a nickname. This option is not required.

PRIMARY_KEY
Indicates that this nickname is a parent nickname. The column data type must be VARCHAR(16). A nickname can have at most one PRIMARY_KEY column option. The only valid value is Y.

The column that is designated as PRIMARY_KEY contains a key that is generated by the wrapper. The value of the column cannot be retrieved in a SELECT query, and the XPATH option must not be specified. The column can only be used to join parent nicknames and child nicknames. Primary key columns must be designated as FOR BIT DATA and NOT NULL.

SWITCH
A character string to specify a parameter for the script on the command line. This option applies only to input columns.

On the command line, the value of this option precedes the column value that is supplied by WSCRIPT_ARGS or the default value, if any. If the value for the switch is an empty string and a default value exists for the column, the default value is added without any SWITCH information when the command line is generated. If no default value is provided and no value for the column is provided by the SQL query, then this input column is ignored when the command line is generated. This option is required for an input column.

SWITCH_ONLY
Enables the use of switches without a command line argument.

If the SWITCH_ONLY option is specified with a value of Y, then valid input values are Y or N. For an input value of Y, only the switch is added to the command line. For an input value of N, no value is added to the command line.

VALID_VALUES
A semicolon-separated set of valid values for a column.

XPATH
Specifies the XPath expression in the XML document that contains the data that corresponds to this column. The script wrapper evaluates the XPath expression after the CREATE NICKNAME statement applies this XPath
expression from this XPATH nickname option. If you run a query on a column name that has an incorrectly configured XPATH tag reference such as incorrect case, your query returns null values in this column for all returned rows.

---

### SQL queries with the script wrapper

SQL queries that are made through the script wrapper use the custom function to carry the parameter input values for the script.

Any SELECT statement that passes parameter values to a script through the script wrapper must contain at least one predicate with a custom function to take the parameter input values for the script.

#### Root nicknames

For example, the following statement creates a root nickname for a script named myscript:

```
CREATE NICKNAME customers (  
  argle double OPTIONS(SWITCH 'argle', POSITION 1, DEFAULT 1.0),  
  argfile CLOB() OPTIONS(SWITCH '-file', INPUT_MODE 'FILE_INPUT', POSITION 2),  
  argpos varchar() OPTIONS(SWITCH', POSITION 3),  
  id varchar(10) OPTIONS(XPATH './@id'),  
  name varchar OPTIONS(XPATH '/name'))  
FOR SERVER script_server  
  OPTIONS(DATASOURCE 'myscript', XPATH 'doc/customer',  
  TIMEOUT '300', VALIDATE 'YES');
```

The custom function statements are as follows:

- `CREATE FUNCTION wsscript.args (varchar(), varchar())`  
  RETURNS INTEGER AS TEMPLATE  
  DETERMINISTIC NO EXTERNAL ACTION;

- `CREATE FUNCTION wsscript.args (date(), date())`  
  RETURNS INTEGER AS TEMPLATE  
  DETERMINISTIC NO EXTERNAL ACTION;

- `CREATE FUNCTION wsscript.args (integer(), integer())`  
  RETURNS INTEGER AS TEMPLATE  
  DETERMINISTIC NO EXTERNAL ACTION;

- `CREATE FUNCTION wsscript.args (CLOB(), CLOB())`  
  RETURNS INTEGER AS TEMPLATE  
  DETERMINISTIC NO EXTERNAL ACTION;

- `CREATE FUNCTION wsscript.args (double(), double)`  
  RETURNS INTEGER AS TEMPLATE  
  DETERMINISTIC NO EXTERNAL ACTION;

The configuration file specifies the following configuration parameters:

```
SCRIPT_OUT_DIR_PATH=C:\temp  
myscript=C:\perl\bin\perl myscript.pl -model
```

To run a query and send the contents of the table t1.bigdata to the daemon and to the local file C:\temp\f12345, issue the following query and wsscript.args command:

```
SELECT id, name FROM customers, t1  
WHERE wsscript.args (customers.argfile, t1.bigdata) = 1
```

The preceding query results in the following command line:

```
C:\perl\bin\perl myscript.pl -model -argle 1.0 -file C:\temp\f12345
```
To run a query that uses the default values for all parameters, run a query on the nickname with no predicates. To avoid problems with excessive output, include the STREAMING option in the nickname.

**Child nicknames**

The script wrapper maps the XML result set from the script into nicknames that have a parent-child relationship. To retrieve data from a child nickname, join the child nickname to its parent nickname up to the root. The SELECT statements that reference a child nickname must be joined with the parent nickname of the child nickname by using primary and foreign key columns.

The following query displays the customer names and amounts for each order of each customer:

```
SELECT c.name, o.amount FROM customers c, orders o
  WHERE c.cid=o.cid
  AND wsscript.args {customers.argfile, t1.bigdata} = 1
  AND wsscript.args {customers.argpos, VARCHAR('test1')} = 1
  AND wsscript.args {customers.argle, FLOAT('3.5')} = 1
```

Specify the join `c.cid=o.cid` statement to indicate the parent-child relationship between the customers nickname and the orders nickname. If you join a child nickname to itself, an error message is returned.

**Optimizing script wrapper performance**

The location of the script daemon can affect query performance.

To improve network communication performance, use a separate script server for the script daemon. Place the federated server and the script server on separate servers. Also, place the script daemon on the script server.
Chapter 15. Configuring access to Sybase data sources

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

Before you begin
- The Sybase Client SDK software must be installed on a server that will act as the federated server. When you install the Sybase client on Windows, you must specify the Full or Custom option. If you specify the custom option, you must the XA Interface Library for ASE Distributed Transaction Manager option.
- IBM WebSphere Federation Server must be installed on a server that acts as the federated server.
- Check the setup of the federated server.
- Check the federated parameter to ensure that federation is enabled.

You can configure a federated server to access data that is stored in Sybase data sources by using the DB2 Control Center or by issuing SQL statements on the DB2 command line. The DB2 Control Center includes a wizard to guide you through the steps that are necessary to configure the required federated objects.

Procedure

To configure access to Sybase data sources:
1. Set the Sybase environment variables.
2. Set up and test the Sybase client configuration file using one of the following methods depending on your operating system:
   - Set up and test the Sybase client configuration file (Linux, UNIX).
   - Set up and test the client configuration file (Windows).
3. Register the wrapper.
4. Register the server definition.
5. Create the user mappings.
6. Test the connection to the Sybase server.
7. Register nicknames for Sybase tables and views.

Sybase wrapper support for Adaptive Server Enterprise (ASE)

The Sybase wrapper supports the Sybase Adaptive Server Enterprise (ASE) 15.0, in addition to ASE 12.5 and ASE 12.0.

Supported clients for Sybase ASE 15.0
- You can connect to the ASE 15.0 by using the Sybase wrapper with the Sybase client, version 12.5.1 or later.
- If you use Software Developer Kit (SDK), version 12.5.1 as a Sybase client, IBM recommends that you to install the Electronic Software Distribution (ESD) #12 or later for the SDK on the federated server that is associated with the Sybase wrapper. If you use SDK, version 15.0 as a Sybase client,
IBM recommends that you install the ESD #3 or later. If you do not install the ESD, you might get an unexpected error while using the Sybase wrapper.

**Updating Sybase libraries for the Sybase client, version 15.0**

UNIX: If you use the Sybase client, version 15.0, you can run the Sybase Insyblibs script to update Sybase library names to preserve consistency among supported versions of Sybase library files. The Insyblibs script creates symbolic links from the new library names to the old library names, which allows the pre-version 15.0 applications to work with renamed libraries.

Windows: If you use the Sybase client, version 15.0 you can run the copylibs.bat file to copy the required *.dll files, which allows the pre-version 15.0 applications to work with renamed libraries.

**Insyblibs script error**

The current Insyblibs script contains an problem. When you run 'Insyblibs create', the following error message occurs:

```
*libsyb*.so: No such file or directory
```

As a workaround to this problem, you can delete the | (bar character) within [] (square brackets) at line 34 of the script. Sybase is aware of this problem. For more information, go to [Sybase support](#).

**Unsupported data types**

You cannot create nicknames for data source objects that contain unsupported data types. The Sybase wrapper does not support the following data types that were introduced in ASE, version 12.5.1:

- DATE
- TIME

The Sybase wrapper does not support the following data types that were introduced in ASE, version 15.0:

- BIGINT
- LONGSYSNAME
- UNITEXT
- UNSIGNED BIGINT
- UNSIGNED INT
- UNSIGNED SMALLINT

---

**Setting the Sybase environment variables**

The Sybase environment variables must be set in the db2dj.ini file on the federated server.

**Restrictions**

Review the restrictions for the db2dj.ini file.

The db2dj.ini file contains configuration information about the Sybase Open Client software that is installed on your federated server.

There are required and optional environment variables for Sybase data sources.
If you installed the Sybase Open Client software before you installed the Sybase wrapper, the required Sybase environment variables are set in the db2dj.ini file.

You must set the environment variables by using the steps in this task if you did not install the Sybase Open Client software before you installed the Sybase wrapper or you want to set any of the optional environment variables.

**Procedure**

To set the Sybase environment variables:

1. Use one of the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Automatically set the environment variables.</td>
<td>Run the WebSphere Federation Server installation wizard. Follow the instructions in the wizard. <strong>Important:</strong> Set the required environment variables by running the installation wizard. The optional environment variables must be set manually.</td>
</tr>
</tbody>
</table>
| Manually set the environment variables. | Edit the db2dj.ini file: The db2dj.ini file is located in the directory that the DB2 registry variable DB2_DJ_INI specifies. When the DB2_DJ_INI variable is not set, the db2dj.ini file is in one of the following default paths depending on the operating system:  
   - On Linux and UNIX: instancehome/sqlib/cfg/db2dj.ini.  
     instancehome  
     The home directory of the instance owner.  
   - On Windows: %DB2PATH%\cfg\db2dj.ini  
     %DB2PATH%  
     The directory where the DB2 database system is installed, for example, C:\Program Files\IBM\sqlib.  
   
   If the file does not exist, you can create a file with the name db2dj.ini by using any text editor. In the db2dj.ini file, you must specify the fully qualified path in the value of the SYBASE environment variable; otherwise you will encounter errors. For example:  
   SYBASE=/opt/sybase  
   SYBASE_OCS=0CS-12_5 |

2. On Linux and UNIX, update the .profile file that is on the federated database instance. Specify information about the Sybase environment variables that you added to the db2dj.ini file. On Windows, the file is updated when you install the Sybase Open Client software.  
   Issue the following commands to update the .profile file:  
   ```bash  
   export SYBASE=\sybase_home_directory  
   export SYBASE_OCS=\0CS\version_release  
   export PATH=$SYBASE/bin:$PATH  
   ```  
3. From the home directory, run the .profile file that is on federated database instance.  
   Issue the following command to run the .profile file:  
   ```bash  
   .\profile  
   ```
4. To ensure that the environment variables are set on the federated server, recycle
the federated database instance with these commands:
    db2stop
    db2start

After you complete this task, you can register the wrapper.

**Sybase environment variables**

There are required and optional environment variables for Sybase data sources.
These variables are set in the db2dj.ini file.

The following environment variables are valid for Sybase:

- SYBASE
- SYBASE_OCS
- SYBASE_CHARSET (optional)

**Variable descriptions**

**SYBASE**

Specifies the directory path where the Sybase Open Client software is
installed. Specify the fully qualified path for this environment variable.

For example, if Sybase Open Client Version 12.5 is installed in the directory
path D:\djxclient\sybase\V125, specify the following SYBASE
environment variable:
    SYBASE=D:\djxclient\sybase\V125

If Sybase Open Client Version 12.0 is installed in the directory path
D:\djxclient\sybase\V12, specify the following SYBASE environment
variable:
    SYBASE=D:\djxclient\sybase\V12

**SYBASE_OCS**

Specifies the directory, version and release of the Sybase Open Client
software that is installed. Do not specify the fully qualified path when you
specify this environment variable.

SYBASE_OCS=OCS-version_release

For example, if Sybase Open Client Version 12.0 is installed in the directory
path D:\djxclient\sybase\V12\OCS-12_0, specify the following value for the SYBASE_OCS environment variable:
    SYBASE_OCS=OCS-12_0

If Sybase Open Client Version 12.5 is installed in the directory path
D:\djxclient\sybase\V125\OCS-12_5, specify the following value for the
SYBASE_OCS environment variable:
    SYBASE_OCS=OCS-12_5

**SYBASE_CHARSET**

Specifies the name of the character set that you want to use. The Sybase
wrapper uses the SYBASE_CHARSET environment variable to determine
which character set to use. You can see a list of valid character set names
in the $SYBASE\charsets directory.

The following example shows how to set the SYBASE_CHARSET when
your character set is UTF8:
If you do not set the SYBASE_CHARSET environment variable, the wrapper uses the Sybase character set that matches the one that is specified on the code page of the federated database. If there is no matching Sybase character set, the wrapper uses the iso_1 character set.

### Setting up and testing the Sybase client configuration file (Windows)

The Sybase client configuration file is used to connect to Sybase databases by using the client libraries that are installed on the federated server.

#### Before you begin

The Sybase Client SDK software must be installed on the federated server.

#### About this task

The client configuration file specifies the location of each Sybase SQL Server and Adaptive Server Enterprise instance and the type of connection (protocol) for the database server.

You must set up a client configuration file on each instance in the federated server that will be used to connect to Sybase.

#### Procedure

To set up and test the Sybase client configuration file on federated servers that run Windows:

1. Set up the client configuration file by using the utility that comes with the Sybase Open Client software. See the Sybase documentation for more information about using this utility.
   The client configuration file is created in the \%SYBASE\%ini directory. The name of the file is sql.ini.

2. Test the connection to ensure that the Sybase Open Client software is able to connect to the Sybase server.
   Use an appropriate Sybase query utility, such as isql, to test the connection.
   For example, if the Sybase Open Client software is installed in the directory path, D:\djxclient\sybase\V125, you can issue the following commands from a command prompt:
   ```
   cd D:\djxclient\sybase\V125\0CS-12_5\bin
   isql -sybnodename -Umary
   ```
   Alternatively, you can issue the following command from a command prompt:
   ```
   %SYBASE%\%SYBASE_OCS%\bin\isql -sybnodename -Umary
   ```

**Specifying the path to the interfaces file**

If you would like to use an interfaces file other than the default file, use the IFILE server option to specify the path. The Sybase wrapper searches for the interfaces file in the following places, in the order specified:

- IFILE server option
- \%DB2PATH\%\interfaces
- \%SYBASE\%ini\sql.ini
After you complete this task, you can set the environment variables.

**Setting up and testing the Sybase client configuration file (UNIX)**

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

**Before you begin**

The Sybase Client SDK software must be installed on the federated server.

The client configuration file specifies the location of each Sybase SQL Server and Adaptive Server Enterprise instance and the type of connection (protocol) for the database server.

You must set up a client configuration file on each instance in the federated server that will be used to connect to Sybase.

**Procedure**

To set up and test the Sybase client configuration file on federated servers that run UNIX:

1. Set up the client configuration file by using the utility that comes with the Sybase Open Client software.
   The client configuration file is created in the `$SYBASE` directory. The default name of the file is `interfaces`. See the Sybase documentation for more information about using this utility.

2. Test the connection to ensure that the Sybase Open Client software is able to connect to the Sybase server.
   Use an appropriate Sybase query utility, such as `isql`, to test the connection.
   For example, if the Sybase Open Client software is installed in the directory path, `/opt/djxclient/sybase/V125`, you can issue the following command from a UNIX prompt:
   ```
cd /opt/djxclient/sybase/V125/0CS-12_5
isql -Ssybnode -Umary
```
   Alternatively, you can issue the following command from a UNIX prompt:
   ```
$SYBASE/$SYBASE_OCS/bin/isql -Ssybnode -Umary
```

**Specifying the path to the interfaces file**

If you would like to use an interfaces file other than the default file, use the `IFILE` server option to specify the path. The Sybase wrapper searches for the interfaces file in the following places, in the order specified:

- `IFILE` server option
- `sqllib/interfaces`
- `$SYBASE/interfaces`

After you complete this task, you can set the environment variables.
Registering the Sybase wrapper

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

Procedure

To register the Sybase wrapper:

Use one of the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Issue the CREATE WRAPPER statement and specify the default name for the Sybase wrapper.</td>
<td>For example: CREATE WRAPPER CTLIB. <strong>Remember:</strong> When you register the wrapper by using the default name, CTLIB, the federated server automatically uses the appropriate Sybase wrapper library for the operating system that your federated server is running on. If the default wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name that you choose. When you use a name that is different from the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement. For example, to register a wrapper with the name sybase_wrapper on a federated server that uses AIX, issue the following statement: CREATE WRAPPER sybase_wrapper LIBRARY 'libdb2ctlib.a'; The wrapper library file that you specify depends on the operating system of the federated server.</td>
</tr>
</tbody>
</table>

After you complete this task, you can register the server definition.

**Sybase wrapper library files**

The Sybase wrapper library files are added to the federated server when you install the wrapper.

When you install the Sybase wrapper, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2ctlib.a, libdb2ctlibF.a, and libdb2ctlibU.a. The other wrapper library files are used internally by the Sybase wrapper.
If you do not use the default wrapper name when you register the wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default library file name.

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

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Library file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/instal_path/lib32/</td>
<td>libdb2ctlib.a</td>
</tr>
<tr>
<td></td>
<td>/usr/opt/instal_path/lib64/</td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/IBM/db2/instal_path/lib32</td>
<td>libdb2ctlib.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/instal_path/lib64</td>
<td></td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/IBM/db2/instal_path/lib32</td>
<td>libdb2ctlib.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/instal_path/lib64</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>%DB2PATH%\bin</td>
<td>db2ctlib.dll</td>
</tr>
</tbody>
</table>

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

### Registering the server definitions for a Sybase data source

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

**Procedure**

To register a server definition for a Sybase data source:

1. Locate the node name in the Sybase interfaces file. The following examples show the entries for interfaces files for federated servers that run UNIX or Windows:
   - **UNIX:**
     - `sybase125`
     - `query tcp ether anaconda 4100`
   - **Windows:**
     - `[sybase125]`
     - `query=TCP, anaconda,4100`

   - The first line in each example is the node name, such as `sybase125`.
   - The second line lists the type of connection, the host name, and the port number. In this example TCP indicates that this is a TCP/IP connection, anaconda is the host name, and 4100 is the port number.

2. To create the server definition, use one of the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Issue the CREATE SERVER statement.        | For example: CREATE SERVER server_definition_name TYPE SYBASE  
|                                            |   VERSION Sybase_client_version_number WRAPPER wrapper_name  
|                                            |   OPTIONS (NODE 'node_name', DBNAME 'database_name');  
|                                            | Although the 'node_name' and 'database_name' variables are specified as options in the CREATE SERVER statement, these options are required for Sybase data sources.  
|                                            | Important: If you did not rename the sql.ini file to interfaces when you set up the Sybase client configuration file, you must include the IFILE server option when you register the server definition.  
|                                            | After the server definition is registered, use the ALTER SERVER statement to add or drop server options.                                                                                               |

After you complete this task, you can create user mappings.

**CREATE SERVER statement - Examples for the Sybase wrapper**

Use the CREATE SERVER statement to register server definitions for the Sybase wrapper. This topic provides a complete example with the required parameters, and an example with additional server options.

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

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

**SYBSERVER**
A name that you assign to the Sybase server. Duplicate server definition names are not allowed.

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

**VERSION 12.0**
The version of the Sybase database client software that is being used for the federated connection.

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

**NODE 'sybnode'**
The name of the node where the Sybase server resides. Obtain the node name from the interfaces file. This value is case sensitive.

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

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

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

When you create a server definition, you can specify additional server options in the CREATE SERVER statement. The server options can be general server options and Sybase-specific server options.

IFILE server option

If you are not using the default interfaces file, you must create an interfaces file and include the IFILE server option in the CREATE SERVER statement. The default interfaces file is $SYBASE/interfaces for Linux and UNIX, and %SYBASE%\ini\sql.ini for Windows.

The value that you specify for the IFILE server option is the full path and name of the Sybase Open Client sql.ini file.

The following example shows how to use the IFILE server option when you register a server definition on a federated server that runs Windows:

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

TIMEOUT server option

The TIMEOUT server option sets the number of seconds that the wrapper waits for a response from the Sybase server. Use the TIMEOUT option to avoid deadlocks on transactions.

The following example shows how to specify the TIMEOUT server option when you register a server definition:

```sql
CREATE SERVER SYBSERVER TYPE SYBASE VERSION 12.0 WRAPPER CTLIB OPTIONS (NODE 'sybnode', DBNAME 'sybdb',
    TIMEOUT '60');
```

The additional Sybase-specific server options are:

- LOGIN_TIMEOUT
- PACKET_SIZE

Creating the user mappings for a Sybase data source

When you attempt to access a Sybase server, the federated server establishes a connection to the Sybase server by using a user ID and password that are valid for that data source. You must define an association (a user mapping) between each federated server user ID and password and the corresponding data source user ID and password.

Create a user mapping for each user ID that will access the federated system to send distributed requests to the Sybase data source.

Procedure

Create user mappings for a Sybase data source:

Issue a CREATE USER MAPPING statement.

For example:
CREATE USER MAPPING FOR local_userID SERVER server_definition_name OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password');

After you complete this task, test the connection to the Sybase server.

**CREATE USER MAPPING statement - Examples for the Sybase wrapper**

Use the CREATE USER MAPPING statement to map a federated server user ID to a Sybase server user ID and password. This topic provides a complete example with the required parameters, and an example that shows you how to use the DB2 special register USER with the CREATE USER MAPPING statement.

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

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

*maria* Specifies the local user ID that you are mapping to a user ID that is defined at the Sybase server.

*SERVER SYBSERVER*

Specifies the server definition name that you registered in the CREATE SERVER statement for the Sybase server.

*REMOTE_AUTHID 'mary'*

Specifies the user ID at the Sybase server to which you are mapping *maria*. The value is case-sensitive, unless you set the FOLD_ID server option to ‘U’ or ‘L’ in the CREATE SERVER statement.

Although the remote user ID is specified as an option in the CREATE SERVER statement, it is required for Sybase data sources.

*REMOTE_PASSWORD 'day2night'*

Specifies the password that is associated with ‘mary’. The value is case-sensitive, unless you set the FOLD_PW server option to ‘U’ or ‘L’ in the CREATE SERVER statement.

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

**DB2 special register USER**

You can use the DB2 special register USER to map the authorization ID of the person who is issuing the CREATE USER MAPPING statement to the data source authorization ID that is specified in the REMOTE_AUTHID user option.

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

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

**Registering nicknames for Sybase tables and views**

For each Sybase server definition that you register, you must register a nickname for each table or view that you want to access. Use these nicknames, instead of the names of the data source objects, when you query the Sybase servers.

**Before you begin**
Update the statistics at the Sybase data source before you register a nickname. The federated database relies on the data source catalog statistics to optimize query processing. Use the data source command that is equivalent to the DB2 RUNSTATS command to update the data source statistics.

**Procedure**

To register a nickname for Sybase tables or views, use one of the following methods:

<table>
<thead>
<tr>
<th>Methods</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click <strong>Create Federated Objects</strong>.</td>
</tr>
</tbody>
</table>
| Issue the CREATE NICKNAME statement. Nicknames can be up to 128 characters in length. | For example:  
CREATE NICKNAME nickname  
FOR server_definition_name."remote_schema"."remote.table" ; |

When you create the nickname, the federated server queries the data source catalog by using the nickname. This query tests the connection to the data source table or view. If the connection does not work, you receive an error message.

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

**CREATE NICKNAME statement - Examples for the Sybase wrapper**

Use the CREATE NICKNAME statement to register a nickname for a Sybase table or view that you want to access. This topic includes a complete example with the required parameters.

The following example shows you how to register a nickname for a Sybase table or view using the CREATE NICKNAME statement.

```
CREATE NICKNAME SYBSALES FOR SYBSERVER."vinnie"."europe";
```

**SYBSALES**

A unique nickname that is used to identify the Sybase table or view.

**Important:** The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authorization ID of the user who is registering the nickname.

**SYBSERVER."vinnie"."europe"**

A three-part identifier for the remote object:

- **SYBSERVER** is the server definition name that you assigned to the Sybase database server in the CREATE SERVER statement.
- **vinnie** is the user ID of the owner to which the table or view belongs.
- **europe** is the name of the remote table or view that you want to access.

The federated server folds the names of the Sybase schemas and tables to uppercase unless you enclose the names in quotation marks.
Registering nicknames for Sybase tables and views

For each Sybase server definition that you register, you must register a nickname for each table or view that you want to access. Use these nicknames, instead of the names of the data source objects, when you query the Sybase servers.

Before you begin

Update the statistics at the Sybase data source before you register a nickname. The federated database relies on the data source catalog statistics to optimize query processing. Use the data source command that is equivalent to the DB2 RUNSTATS command to update the data source statistics.

Procedure

To register a nickname for Sybase tables or views, use one of the following methods:

<table>
<thead>
<tr>
<th>Methods</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Issue the CREATE NICKNAME statement. Nicknames can be up to 128 characters in length.</td>
<td>For example: CREATE NICKNAME nickname FOR server_definition_name.&quot;remote_schema&quot;.&quot;remote.table&quot;;</td>
</tr>
</tbody>
</table>

When you create the nickname, the federated server queries the data source catalog by using the nickname. This query tests the connection to the data source table or view. If the connection does not work, you receive an error message.

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

CREATE NICKNAME statement - Examples for the Sybase wrapper

Use the CREATE NICKNAME statement to register a nickname for a Sybase table or view that you want to access. This topic includes a complete example with the required parameters.

The following example shows you how to register a nickname for a Sybase table or view using the CREATE NICKNAME statement.

```
CREATE NICKNAME SYBSALES FOR SYBSERVER."vinnie"."europe";
```

SYBSALES
A unique nickname that is used to identify the Sybase table or view.

**Important:** The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authorization ID of the user who is registering the nickname.

SYBSERVER."vinnie"."europe"
A three-part identifier for the remote object:
• SYBSERVER is the server definition name that you assigned to the Sybase database server in the CREATE SERVER statement.
• vinnie is the user ID of the owner to which the table or view belongs.
• europe is the name of the remote table or view that you want to access.

The federated server folds the names of the Sybase schemas and tables to uppercase unless you enclose the names in quotation marks.

Problems loading the Sybase wrapper library

When you create the Sybase wrapper, you might encounter errors related to the installation of the Sybase Open Client software that prevents the Sybase wrapper library from being loaded.

Symptom

When you create the Sybase wrapper, the following SQL error is issued:
SQL10013N The specified library "db2ctlibF.dll" could not be loaded.

Cause

The Sybase XA Interface Library for ASE Distributed Transaction Manager was not installed on the Windows system with the Sybase Open Client software.

Resolving the problem

Reinstall the Sybase Open Client software on Windows and select the Full or Custom installation option. If you select the Custom installation option, specify the XA Interface Library for ASE Distributed Transaction Manager option.

Missing SYBASE environment variable

If the db2dj.ini file is not in the correct directory or if it is missing, you will encounter an SQL error.

Symptom

The following SQL error is issued:
SQL1822N Unexpected error code "" received from data source
"server name". Associated text and tokens are "SYBASE variable not set".

Cause

The db2dj.ini file was not found or the file does not contain the SYBASE environment variable. The db2dj.ini file is located in the directory that the DB2 registry variable DB2_DJ_INI specifies. When the DB2_DJ_INI variable is not set, the db2dj.ini file is in one of the following default paths depending on the operating system:
• On UNIX: instancehome/sql/lib/cfg/db2dj.ini.
  instancehome
  The home directory of the instance owner.
• On Windows: %DB2PATH%\cfg\db2dj.ini
%DB2PATH%
The directory where the DB2 database system is installed, for example, C:\Program Files\IBM\sqlib.

Resolving the problem

Create a db2dj.ini file with the required Sybase environment variables and place it in the correct directory for your operating system. You can use a text editor to create the db2dj.ini file.

Missing Sybase node name

If the Sybase client configuration file is not properly configured, the federated system might be unable to locate the Sybase node name.

Symptom

The following SQL error is issued:
SQL1097N The node name was not found in the node directory.

Cause

The Sybase node name was not found.

Resolving the problem

To resolve the problem:
• If the IFILE server option was specified, verify that the node name is declared in the file that it specifies.
• If the interfaces file exists in sqlib directory, verify that the node name is declared in it.
• If the IFILE server option is not specified and the interfaces file does not exist in the sqlib directory, verify that the node name is declared in the %SYBASE%\ini\sql.ini file on Windows or the $SYBASE/interfaces file on UNIX.
Chapter 16. Configuring access to table-structured file data sources

You can integrate the data that is in table-structured file with information from other sources by using a federated system.

Procedure

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

Table-structured files - overview

A table-structured file is a file that has a regular structure that consists of a series of records or rows of information. Each record contains the same number of fields. The data in the fields are separated by a delimiter, such as a comma.

The following example shows the contents of a file called DRUGDATA1.TXT. It contains three records, each with three fields, separated by commas:

234,DrugnameA,Manufacturer1
332,DrugnameB,Manufacturer2
333,DrugnameC,Manufacturer2

The first field is the unique ID number for the drug. The second field is the name of the drug. The third field is the name of the manufacturer who produces the drug.

The field delimiter can be more than one character in length. A single quotation mark cannot be used as a delimiter. The delimiter must be consistent throughout the file. A null value is represented by two delimiters next to each other or a delimiter followed by a line terminator, if the NULL field is the last one on the line. The column delimiter cannot exist as valid data for a column.

For example:

234,DrugnameA,Manufacturer1
332,DrugnameB,Manufacturer2
333,DrugnameC,Manufacturer2
356,,Manufacturer1

Attributes of table-structured files

The records, or rows, in table-structured files can be sorted or unsorted. The table-structured files wrapper can search files that are sorted more efficiently than files that are not sorted.

If the data in a table-structured file is sorted, the sort must be in ascending order on the key column. You should set the SORTED option to Y on the key column when you create define the columns for the nickname. Otherwise the wrapper processes the data in the table-structured file as unsorted data.
Sorted files

DRUGDATA1.TXT contains sorted records. The file is sorted by the first field, the unique ID number for the drug. This field is the primary key because it is unique for each drug. Sorted files must be sorted in ascending order.

234,DrugnameA,Manufacturer1
332,DrugnameB,Manufacturer2
333,DrugnameC,Manufacturer2

Unsorted files

DRUGDATA2.TXT contains unsorted records. There is no order to the way the records are listed in the file.

556,DrugnameB,Manufacturer2
234,DrugnameA,Manufacturer1
721,DrugnameC,Manufacturer2

Table-structured files wrapper

Data in a table-structured file can be joined with data in other table-structured files, relational data, or nonrelational and unstructured data.

Using a wrapper, the federated server can process SQL statements that query data in a table-structured file as if the data is contained in an ordinary relational table or view.

How the federated server works with table-structured files is illustrated in the following figure.

![Diagram of how the table-structured file wrapper works](image)

**Figure 14. How the table-structured file wrapper works**

For example, the table-structured file DRUGDATA2.TXT is located on a computer in your laboratory. The data in the file is:

556,DrugnameB,Manufacturer2
234,DrugnameA,Manufacturer1
721,DrugnameC,Manufacturer2

It is tedious to try to query and match this data with other tables from other data sources that you use.
After you register the DRUGDATA2.TXT file with the federated server, the table-structured file wrapper can access the data in the file as if the data is in a relational table.

For example, you can run the following query:

```sql
SELECT * FROM DRUGDATA2 ORDER BY DCODE
```

This query produces the following results:

<table>
<thead>
<tr>
<th>DCODE</th>
<th>DRUG</th>
<th>MANUFACTURER</th>
</tr>
</thead>
<tbody>
<tr>
<td>234</td>
<td>DrugnameA</td>
<td>Manufacturer2</td>
</tr>
<tr>
<td>556</td>
<td>DrugnameB</td>
<td>Manufacturer1</td>
</tr>
<tr>
<td>721</td>
<td>DrugnameC</td>
<td>Manufacturer2</td>
</tr>
</tbody>
</table>

You can join the data in the DRUGDATA2.TXT file with data from other relational and nonrelational data sources and analyze all of the data together.

### Adding table-structured file data sources to a federated server

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

**Before you begin**

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

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

**Procedure**

To add the table-structured file data sources to a federated server:

1. Register the table-structured file wrapper.
2. Register the table-structured file server definition.
3. Register nicknames for the table-structured files.

### Registering the table-structured file wrapper

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

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

**Procedure**

To register the table-structured file wrapper:
Choose the method that you want to use to register the table-structured file wrapper:

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>From the command line</td>
<td>Issue the CREATE WRAPPER statement. For example to register a wrapper with the name flat_files_wrapper on the federated server that uses the AIX operating system, issue the following statement:</td>
</tr>
<tr>
<td></td>
<td>CREATE WRAPPER flat_files_wrapper LIBRARY 'libdb2lsfile.a';</td>
</tr>
</tbody>
</table>

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

**Table-structured files wrapper library files**

The table-structured files wrapper library files are added to the federated server when you install IBM WebSphere Federation Server.

When you install IBM WebSphere Federation Server, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2lsfile.a, libdb2lsfileF.a, and libdb2lsfileU.a. The default wrapper library file is libdb2lsfile.a. The other wrapper library files are used with specific wrapper options.

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

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

**Table 81. Table-structured files client library locations and file names**

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Wrapper library file names</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/&lt;install_path&gt;/lib32/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/usr/opt/&lt;install_path&gt;/lib64/</td>
<td>libdb2lsfile.a</td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib64</td>
<td>libdb2lsfile.so</td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib64</td>
<td>libdb2lsfile.so</td>
</tr>
<tr>
<td>Windows</td>
<td>%DB2PATH%\bin</td>
<td>db2lsfile.dll</td>
</tr>
</tbody>
</table>

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

Registering the server definition for table-structured files

For table-structured files, you must register a server definition because the hierarchy of federated objects requires that the table-structured files, which are identified by nicknames, are associated with a specific server definition object.

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

Procedure

To register a server definition for a table-structured file data source:

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard.</td>
</tr>
<tr>
<td></td>
<td>Right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>From the command line</td>
<td>Issue the CREATE SERVER statement.</td>
</tr>
<tr>
<td></td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td>CREATE SERVER server_definition_name WRAPPER wrapper_name;</td>
</tr>
</tbody>
</table>

CREATE SERVER statement - example for the table-structured file wrapper

Use the CREATE SERVER statement to register the server definition for the table-structured file wrapper. This example shows the required parameters.

The following example shows you how to register a server definition called biochem_lab for a text file that contains biochemical data. The CREATE SERVER statement that you issue is:

```
CREATE SERVER biochem_lab WRAPPER flat_files_wrapper;
```

- **biochem_lab**
  - A name that you assign to the table-structured file server definition.
  - Duplicate server definition names are not allowed.

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

Registering nicknames for table-structured files

You must register a nickname for each table-structured file that you want to access. Use these nicknames, instead of the names of the files, when you query table-structured file data sources.

Restrictions

- If a nonnumeric field is too long for its column type, the excess data is truncated.
- If a decimal field in the file has more digits after the radix character than are allowed, the excess data is truncated. For example, if the number is 10.123456 and the data type specifies that 3 digits are allowed after the radix character, the number is truncated to 10.123. The radix character is determined by the RADIXCHAR item of the LC_NUMERIC National Language Support category. The scale parameter for the column data type specifies the number of digits allowed after the radix character.

- If you attempt to access table-structured file data sources that are on a shared drive from federated server that runs Windows 2003, your queries might fail. This is a limitation of Windows 2003. You can avoid this problem by specifying the absolute path in the FILE_PATH option in the CREATE NICKNAME statement.

- The maximum length for a line of data is 10 megabytes (10485760 bytes).

When you create a nickname for a table-structured file, the information in the data in the file is mapped to a relational table. You create nicknames for your table-structured file in one of two ways:

- Specifying the table-structured file when you create the nickname by using the FILE_PATH nickname option.

- Specifying the table-structured file when you query the data source, by using the DOCUMENT nickname column option. When this option is used, the nickname can be used to represent data from any table-structured file whose schema matches the nickname definition.

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

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

**Procedure**

To register a nickname for a table-structured file:

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>Using the command line</td>
<td>Issue the CREATE NICKNAME statement. For example:</td>
</tr>
<tr>
<td></td>
<td>CREATE NICKNAME nickname</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>column_name data_type,</td>
</tr>
<tr>
<td></td>
<td>column_name data_type,</td>
</tr>
<tr>
<td></td>
<td>column_name data_type</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>FOR SERVER server_definition_name</td>
</tr>
<tr>
<td></td>
<td>OPTIONS (nickname_options);</td>
</tr>
</tbody>
</table>

Repeat this step for each table-structured file that you want to create a nickname for.
CREATE NICKNAME statement - examples for table-structured file wrapper

Use the CREATE NICKNAME statement to register a nickname for a table-structured file that you want to access.

You must specify either the FILE_PATH nickname option or the DOCUMENT nickname column option when you register a nickname for a table-structured file.

Creating a nickname with the FILE_PATH nickname option

The following example shows a CREATE NICKNAME statement for the table-structured file DRUGDATA1.TXT:

```
CREATE NICKNAME DRUGDATA1
  (Dcode INTEGER NOT NULL,
   Drug CHAR(20),
   Manufacturer CHAR(20))
FOR SERVER biochem_lab
OPTIONS(FILE_PATH '/usr/pat/DRUGDATA1.TXT')
```

**DRUGDATA1**

A unique nickname that is used to identify the table-structured file.

The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname is the authorization ID of the user who registers the nickname.

**Dcode INTEGER NOT NULL**

The name and data type for a table-structured file column that contains a drug code.

**Drug CHAR(20)**

The name and data type for a table-structured file column that contains the name of a drug.

**Manufacturer CHAR(20)**

The name and data type for a table-structured file column that contains the name of the drug manufacturer.

**FOR SERVER biochem_lab**

The name that you assigned to the table-structured file server definition in the CREATE SERVER statement.

**FILE_PATH '/usr/pat/DRUGDATA1.TXT'**

Specifies the fully qualified directory path and file name for the table-structured file that contains the data you want to access. The path must be enclosed in single quotation marks.

Creating a nickname with the DOCUMENT nickname column option

When you create a nickname using the DOCUMENT nickname column option, you are specifying that the name of table-structured file will be supplied when you run a query that uses the nickname. You can specify the DOCUMENT nickname column option on only one column when you register the nickname. The column that is associated with the DOCUMENT option must be either a VARCHAR or CHAR data type. You must include the full path of the file when you run a query that uses the nickname.
The following example shows a CREATE NICKNAME statement that specifies the DOCUMENT nickname column option:

```
CREATE NICKNAME customers
    
    doc VARCHAR(100) OPTIONS(DOCUMENT 'FILE'),
    name VARCHAR(16),
    address VARCHAR(30),
    id VARCHAR(16)
    FOR SERVER biochem_lab
```

`customers`
A unique name for the nickname.

The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname is the authorization ID of the user who registers the nickname.

`doc VARCHAR(100) OPTIONS(DOCUMENT 'FILE')`
The name and data type for a column that is used to specify the name of the table-structured file that you want to access. You specify the file name when you run the query.

`name VARCHAR(16)`
The name and data type for a table-structured file column that contains the name of the customer.

`address VARCHAR(30)`
The name and data type for a table-structured file column that contains the address of the customer.

`id VARCHAR(16)`

`FOR SERVER biochem_lab`
The name that you assigned to the table-structured file server definition in the CREATE SERVER statement.

`FILE_PATH '/usr/pat/DRUGDATA1.TXT'`
Specifies the fully qualified directory path and file name for the table-structured file that contains the data you want to access. You must specify either the FILE_PATH or DOCUMENT nickname option in the CREATE NICKNAME statement. The path must be enclosed in single quotation marks.

**Creating a nickname with the optional parameters**

The following example shows a CREATE NICKNAME statement for the table-structured file DRUGDATA1.TXT:

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

`DRUGDATA1`
A unique nickname that is used to identify the table-structured file.
The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname is the authorization ID of the user who registers the nickname.

**Dcode** *INTEGER NOT NULL*  
The name and data type for a table-structured file column that contains a drug code.

**Drug** *CHAR(20)*  
The name and data type for a table-structured file column that contains the name of a drug.

**Manufacturer** *CHAR(20)*  
The name and data type for a table-structured file column that contains the name of the drug manufacturer.

**FOR SERVER** *biochem.lab*  
The name that you assigned to the table-structured file server definition in the CREATE SERVER statement.

**FILE_PATH** '/usr/pat/DRUGDATA1.TXT'  
Specifies the fully qualified directory path and file name for the table-structured file that contains the data you want to access. You must specify either the FILE_PATH or DOCUMENT nickname option in the CREATE NICKNAME statement. The path must be enclosed in single quotation marks.

**COLUMN_DELIMITER** ', '  
Specifies the delimiter that is used to separate the fields in a table-structured file. The delimiter value must be enclosed in single quotation marks. The column delimiter can be more than one character in length. If you do not specify a column delimiter, the default delimiter is a comma. A single quote cannot be used as a delimiter. The column delimiter must be consistent throughout the file. A null value is represented by two delimiters next to each other or a delimiter followed by a line terminator, if the NULL field is the last one on the line. The column delimiter cannot exist as valid data for a column. For example, a column delimiter of a comma cannot be used if one of the columns contains data with embedded commas.

**SORTED** 'Y'  
Specifies that the data source file is sorted. Sorted data sources must be sorted in ascending order according to the collation sequence for the current locale, as defined by the settings in the LC_COLLATE National Language Support category. If you specify that the data source is sorted, set the VALIDATE_DATA_FILE option to 'Y'. The default value for the SORTED parameter is 'N'.

**KEY_COLUMN** 'Dcode'  
The name of the column in the file that forms the key on which the file is sorted. The key column value must be enclosed in single quotation marks. Specify this option only if you specify the SORTED nickname option. A column that is designated with the DOCUMENT nickname column option must not be specified as the key column. The value must be the name of a column that is defined in the CREATE NICKNAME statement. The key column must not contain null values. Only single column keys are supported. Multiple column keys are not allowed. The column must be sorted in ascending order. If the value is not specified for a sorted nickname, it defaults to the first column in the nicknamed file.
VALIDATE_DATA_FILE 'Y'
   Specifies if the wrapper verifies that the key column is sorted in ascending
   order and checks for null keys. The valid values for this option are Y or N.
   This option is not allowed if the DOCUMENT nickname column option is
   used for the file path.

Designating key columns when you register a nickname

You can designate a key column by specifying the NOT NULL constraint in the
nickname statement:
CREATE NICKNAME tox (tox_id INTEGER NOT NULL, toxicity VARCHAR(100))
   FOR SERVER tox_server1
      OPTIONS (FILE_PATH'/tox_data.txt', SORTED 'Y')
CREATE NICKNAME weights (mol_id INTEGER, wt VARCHAR(100) NOT NULL)
   FOR SERVER wt_server
      OPTIONS (FILE_PATH'/wt_data.txt', SORTED 'Y', KEY_COLUMN 'WT')

NOT NULL
   Specifies that the column cannot contain null, or blank, values.
   The wrapper does not enforce the NOT NULL constraint, but the federated
database does. If you create a nickname and attach a NOT NULL
constraint on a column and then select a row containing a null value for
the column, the federated database issues a SQL0407N error stating that
you cannot assign a NULL value to a NOT NULL column.

   The exception to this rule is for sorted nicknames. The key column for
   sorted nicknames cannot be NULL. If a NULL key column is found for a
   sorted nickname, the SQL1822N error is issued, stating that the key column
   is missing.

Case sensitive column names

The federated database changes the column names to uppercase unless the column
is defined with double quotation marks. The following example will not work
correctly because the value of the KEY_COLUMN option is enclosed in single
quotation marks. In the example, the column name will be converted by the
federated database to EMPNO. As a result, when you specify empno in a query, the
column is not recognized by the federated database.
CREATE NICKNAME depart (
   empno char(6) NOT NULL)
   FOR SERVER DATASTORE
      OPTIONS(FILE_PATH'data.txt', SORTED 'Y', KEY_COLUMN 'empno');

Windows 2003 federated servers

If you attempt to access table-structured file data sources that are on a shared drive
from a federated server that runs Windows 2003, your query might fail with the
following error message:
SQL1822N Unexpected error code "ERRNO = 2" received from data source
"SERVERNAME1". Associated text and tokens are "Unable to read file".
SQLSTATE=560BD

   This is a limitation of Windows 2003. You can avoid this problem by specifying the
   absolute path in the FILE_PATH option in the CREATE NICKNAME statement.

   The following example shows a CREATE NICKNAME statement with an
   abbreviated path specified in the FILE_PATH option:
CREATE NICKNAME nickname
  
  (  
    COL1 CHAR (10) NOT NULL  
  )  
  FOR SERVER servername1
  OPTIONS (FILE_PATH 'X:\textfile1.txt');

where X:\ is the drive that maps to the remote machine. Queries that use this nickname might fail because you specified the abbreviated path.

For federated server that runs Windows 2003, specify the absolute path in the FILE_PATH option in the CREATE NICKNAME statement.

For example:

CREATE NICKNAME nickname
  
  (  
    COL1 CHAR (10) NOT NULL  
  )  
  FOR SERVER servername1
  OPTIONS (FILE_PATH '\\host.svl.ibm.com\D$\textfile1.txt');

---

**File access control model for the table-structured file wrapper**

The wrapper accesses table-structured files using the authorization information of the federated database instance owner. The wrapper can only access files that can be read by this user ID or group ID. The authorization ID that establishes the connection to the federated database is not used to access table-structured files.

On a federated server, any table-structured file for which a nickname has been created must be accessible with the same path name from each node. The file does not have to be on a federated database node as long as the file can be accessed from any node with a common path.

To access a table-structured file, the wrapper needs a user identity for security purposes. The table-structured file wrapper uses the user identity that is associated with the federated database service. The name of the federated database service depends on the name of the database instance. For example, if the database instance name is DB2, then the service name is DB2 - DB2. To determine the user identity that is associated with a federated database service, use the Control Panel in Windows to display the services. Double-click the service name and display the Log On properties page.

**Table-structured files located on remote drives**

The table-structured files that you want to access must be on a local or mapped drive.

**Networks that have a Windows domain configured**

The logon account for the federated database service must be an account from the domain that has access to the shared folder on the mapped drive where the table-structured files reside.

**Networks that do not have a Windows domain configured**

The federated database service logon account should have the same user name and password as a valid user on the computer that shares that folder. That user must be on the permissions list for the shared folder with at least read access.
Guidelines for optimizing query performance for the table-structured file wrapper

You can improve the performance of table structured file queries by having files that are sorted and by creating statistics for your nicknames.

Use the following tips to improve query performance:
• Sort the data in your files. The federated server can search files that are sorted much more efficiently than files that are not sorted.
• For sorted files, specify a value or range for the key column when you submit a query.
• Statistics for nicknames of table-structured files must be updated manually by updating the SYSSTAT and SYSCAT views. Use the Nickname statistics update facility to update the statistics for the table-structured file nicknames.
Chapter 17. Configuring access to Teradata data sources

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

Before you begin

- Teradata client software must be installed and configured on the server that acts as the federated server.
- Federation must be installed on a server that acts as the federated server.
- Check the setup of the federated server.
- Check the federated parameter to ensure that federation is enabled.

About this task

You can configure a federated server to access data that is stored in Teradata data sources by using the DB2 Control Center or by issuing SQL statements on the DB2 command line. The DB2 Control Center includes a wizard to guide you through the steps that are necessary to configure the required federated objects.

Procedure

To configure access to Teradata data sources:
1. **Test the connection to the Teradata server**
2. **Verify that the Teradata library is enabled for run-time linking (AIX)**
3. **Set the environment variables for the Teradata wrapper**
4. **Register the wrapper**
5. **Register the server definition**
6. **Create the user mappings**
7. **Test the connection from the federated server to the Teradata server**
8. **Register nicknames for Teradata tables and views**

**Testing the connection to the Teradata server**

Test the connection to the Teradata server to verify that the Teradata client software is properly set up on the federated server.

Before you begin

The Basic Teradata Query (BTEQ) utility and the Teradata Data Connector Application Program Interface (PIOM) must be installed on the federated server. The BTEQ utility and the Teradata Data Connector Application Program Interface are installed on the federated server when you install the Teradata client software.

About this task

You use the BTEQ utility to submit an SQL query to verify that the federated server can connect to the Teradata server. See the Teradata documentation for more information about the BTEQ utility.
Procedure

To test the connection to the Teradata server:
1. Start a BTEQ utility session and log on to the Teradata server.
2. Issue an SQL command to verify that you can successfully connect to the Teradata server.
   For example:
   ```sql
   select count(*) from dbc.tables;
   ```
   If the connection is successful, you will see the output from the query.
   If the connection is unsuccessful, you will receive an error. Check the Teradata client software to verify that it is properly installed and configured on the federated server.
3. Log off from the Teradata server and end the BTEQ utility session.

After you complete this task, verify that the Teradata library is enabled for run-time linking.

Verifying that the Teradata library is enabled for run-time linking (AIX)

When you add a Teradata data source to your federated server on AIX, you must verify that run-time linking is enabled before you register wrappers or servers.

Procedure

To verify that the Teradata library is enabled for run-time linking:
1. Go to the directory in which the libcliv2.so file resides.
   The libcliv2.so file is installed with the Teradata client software. By default, it is installed in the /usr/lib directory.
2. From a command prompt, issue the following UNIX command to verify if run-time linking is enabled:
   ```bash
   dump -H libcliv2.so | grep libtli.a
   ```
3. Check the file names that are returned. If the libtli.a file name is returned, the Teradata library is enabled for run-time linking.
   If the libtli.a file name is not returned, open a command window and issue the following UNIX commands to enable run-time linking for the Teradata library:
   ```bash
   rtl_enable libcliv2.so -F libtli.a
   mv libcliv2.so libcliv2.so.old
   mv libcliv2.so.new libcliv2.so
   chmod a+r libcliv2.so
   ```

After you complete this task, you can set the environment variables.

Setting the Teradata environment variables

The Teradata environment variables must be set in the db2dj.ini file on the federated server.

Restrictions

Review the restrictions for the db2dj.ini file.

The db2dj.ini file contains configuration information about the Teradata client software that is installed on your federated server.
There are required and optional environment variables for Teradata data sources. If you installed the Teradata client software before you installed the Teradata wrapper, the required Teradata environment variables are set in the db2dj.ini file. You must set the environment variables by using the steps in this task if you did not install the Teradata client software before you installed the Teradata wrapper or if you want to set any of the optional environment variables.

**Procedure**

To set the Teradata environment variables:

1. Use one of the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click <strong>Create Federated Objects</strong>.</td>
</tr>
<tr>
<td>Automatically set the environment variables.</td>
<td>Run the WebSphere Federation Server installation wizard. Follow the instructions in the wizard. <strong>Important:</strong> Set the required environment variables by running the installation wizard. The optional environment variables must be set manually.</td>
</tr>
</tbody>
</table>
   | Manually set the environment variables. | Edit the db2dj.ini file: The db2dj.ini file is located in the directory that the DB2 registry variable DB2\DJ\INI specifies. When the DB2\DJ\INI variable is not set, the db2dj.ini file is in one of the following default paths depending on the operating system:  
     - On UNIX: \instancehome/sqlibcfg/db2dj.ini.  
       \instancehome  
       The home directory of the instance owner.  
     - On Windows: %DB2PATH%\cfg\db2dj.ini  
       %DB2PATH%  
       The directory where the DB2 database system is installed, for example, C:\Program FilesIBM\sqlib.  
       If the file does not exist, you can create a file with the name db2dj.ini by using any text editor. In the db2dj.ini file, you must specify the fully qualified path in the value of the environment variables; otherwise you will encounter errors. For example:  
       COPLIB=/usr/lib |

2. Set the Teradata code page conversion environment variables (as necessary).

3. To ensure that the environment variables are set on the federated server, recycle the federated database instance with these commands:

   
   ```
   db2stop  
   db2start
   ```

   After you complete this task, you can register the wrapper.

**Teradata environment variables**

There are required and optional environment variables for Teradata data sources. These variables are set in the db2dj.ini file.
The following environment variables are valid for Teradata:
- COPLIB
- COPERR
- TERADATA_CHARSET (optional)
- NETRACE (optional)
- COPANOMLOG (optional)

**Variable descriptions**

**COPLIB**
Specifies the directory path on the federated server for the libcliv2.so file.
Specify the fully qualified path for the COPLIB variable.

For example:
```bash
COPLIB=/usr/lib
```

The libcliv2.so and errmsg.cat files typically reside in the same directory.

**COPERR**
Specifies the directory path on the federated server for the errmsg.cat file.
Specify the fully qualified path for the COPERR variable.

For example:
```bash
COPERR=/usr/lib
```

**TERADATA_CHARSET**
Specifies the code page character set to use with Teradata data sources.

Each time that the federated server connects to a Teradata data source, the Teradata wrapper determines which code page character set to use for that connection. You can have the Teradata wrapper set the code page character set or you can designate a code page by setting the TERADATA_CHARSET environment variable.

If the TERADARA_CHARSET environment variable is set in the db2dj.ini file on the federated server, the wrapper uses the code page character set in the db2dj.ini file. The value in the TERADATA_CHARSET environment variable is not validated, but if the environment variable is not set to a valid value, the Teradata data source returns an error.

If the TERADARA_CHARSET environment variable is not set in the db2dj.ini file on the federated server, the wrapper detects the client character set based on the code page of the database.

On federated servers that run UNIX, the following values are valid for the TERADATA_CHARSET environment variable:
- HANGULKSC5601_2R4
- KanjiEUC_0U
- LATIN1_0A
- LATIN9_0A
- LATIN1252_0A
- SCHGB2312_1T0
- TCHBIG5_1R0
- UTF8
- ASCII
On federated servers that run Windows, the following values are valid for the TERADATA_CHARSET environment variable:

- HANGULSC5601_2R4
- KanjiSJIS_0S
- LATIN1_0A
- LATIN1252_0A
- SCHGB2312_1T0
- TCHBIG5_1R0
- UTF8
- ASCII

**NETRACE**
Optional. Enables the tracing feature of the Teradata client software. This variable is needed only for debugging.

**COPANOMLOG**
Optional. Enables the logging feature of the Teradata client software. This variable is needed only for debugging.

**Verifying the character set on the Teradata server**

If the correct character set is not specified on the Teradata server, you can receive connection errors. Verify that the character set that you want to use is installed on the Teradata server.

**Procedure**

To verify that the character set that you want to use is installed on the Teradata server:

1. Log on to the Teradata server by using the BTEQ utility or any other valid logon utility.
2. Issue the following statement to display the dbc.chartranslations table:
   ```sql
   select *
   from dbc.chartranslations;
   ```
3. Check the value in the third column, InstallFlag, of the table that is returned. The value ‘Y’ in the third column indicates that the character set is installed and in use on the Teradata server.

Use the following table to determine if you have the correct character set installed:

**Table 82. Character sets for Teradata**

<table>
<thead>
<tr>
<th>Double-byte character set</th>
<th>Single-byte character set</th>
<th>Teradata character set</th>
<th>Language</th>
<th>IBM DB2 code set</th>
</tr>
</thead>
<tbody>
<tr>
<td>941</td>
<td>897</td>
<td>&quot;KanjiSJIS_0S&quot;</td>
<td>Japanese</td>
<td>IBM-943</td>
</tr>
<tr>
<td>1362</td>
<td>1126</td>
<td>&quot;HANGULSC5601_2R4&quot;</td>
<td>Korean</td>
<td>1363</td>
</tr>
<tr>
<td>1385</td>
<td>1114</td>
<td>&quot;SCHGB2312_1T0&quot;</td>
<td>Simplified Chinese</td>
<td>GBk</td>
</tr>
<tr>
<td>380</td>
<td>1115</td>
<td>&quot;SCHGB2312_1T0&quot;</td>
<td>Simplified Chinese</td>
<td>IBM-1381</td>
</tr>
<tr>
<td>947</td>
<td>1114</td>
<td>&quot;TCHBIG5_1R0&quot;</td>
<td>Traditional Chinese</td>
<td>big5</td>
</tr>
<tr>
<td>1200</td>
<td>1208</td>
<td>&quot;UTF8&quot;</td>
<td>Unicode</td>
<td>UTF-8</td>
</tr>
<tr>
<td>0</td>
<td>819</td>
<td>&quot;Latin1_0A&quot;</td>
<td>English (Latin 1)</td>
<td>ISO8859-1</td>
</tr>
<tr>
<td>0</td>
<td>1252</td>
<td>&quot;Latin1252_0A&quot;</td>
<td>English (Win Latin)</td>
<td>ISO8859-1/15</td>
</tr>
<tr>
<td>0</td>
<td>819</td>
<td>&quot;ASCII&quot;</td>
<td>English (ASCII)</td>
<td>ISO8859-1</td>
</tr>
</tbody>
</table>
4. If you do not have the required character set installed, install the character set to use the Teradata wrapper.
   - ASCII is enabled on the Teradata server, but it is not cataloged in the dbc.chartranslations table. If all of the values that are returned for InstallFlag are 'N', ASCII is the only valid character set on the Teradata server and the TERADATA_CHARSET environment variable must be set to ASCII in the db2dj.ini file.
   - If the character set that you want to use is listed in the dbc.chartranslations table, but the InstallFlag value is set to 'N', issue the following statement to change the InstallFlag to 'Y':
     ```sql
     update dbc.chartranslations
     set installflag='Y' where CharSetName= 'character_set_name';
     ```
   - If the character set that you want to use is not listed in the dbc.chartranslations table, contact Teradata customer support.

5. Restart the Teradata server to update the list of character sets. In a Teradata command window, enter: `tpareset -f reason_for_restart`

### Troubleshooting character sets for Teradata data sources

When you set the TERADATA_CHARSET environment variable for a Teradata data source, you might encounter errors if the correct character set is not specified.

**Symptom**

If the correct character set is not specified for the Teradata data source, you will encounter the following error:

```
SQL 1822N Unexpected error code "227"
received from data source "<string>". Associated text and tokens are "MTDP: EM_CHARNAME(227): invalid character set name speci". SQLSTATE=560BD
```

**Cause**

The character set that is specified in the TERADATA_CHARSET environment variable is incorrect.

**Resolving the problem**

Verify that the correct character set is installed and specified on the Teradata server and in the db2dj.ini file.

### Registering the Teradata wrapper

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

**Procedure**

To register the Teradata wrapper:
Use one of the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
</tbody>
</table>
| Issue the CREATE WRAPPER statement and specify the default name for the Teradata wrapper. | For example: CREATE WRAPPER TERADATA;  

Remember: When you register the wrapper by using the default name, TERADATA, the federated server automatically uses the appropriate Teradata wrapper library for the operating system that your federated server is running on.  
If the default wrapper name conflicts with an existing wrapper name in the federated database, you can substitute the default wrapper name with a name that you choose. When you use a name that is different from the default name, you must include the LIBRARY parameter in the CREATE WRAPPER statement.  
For example, to register a wrapper with the name tera_wrapper on a federated server that uses AIX, issue the following statement: CREATE WRAPPER tera_wrapper  
LIBRARY 'libdb2teradata.a'  
OPTIONS (DB2_FENCED 'Y',  
DB2_SOURCE_CLIENT_MODE '32BIT');  

The wrapper library file that you specify depends on the operating system of the federated server.  
Mandatory options for AIX and Solaris  
Note: The DB2_FENCED and DB2_SOURCE_CLIENT_MODE wrapper options in the CREATE WRAPPER statement are mandatory for AIX and Solaris because the Teradata wrapper supports only 32-bit clients.  
For example: CREATE WRAPPER TERADATA  
OPTIONS (DB2_FENCED 'Y',  
DB2_SOURCE_CLIENT_MODE '32BIT') |

After you complete this task, you can register the server definition.

**Teradata wrapper library files**

The Teradata wrapper library files are added to the federated server when you install the wrapper.

When you install the Teradata wrapper, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2teradata.a, libdb2teradataF.a, and libdb2teradataU.a. The default wrapper library file is libdb2teradata.a. The other wrapper library files are used internally by the Teradata wrapper.

If you do not use the default wrapper name when you register the wrapper, you must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.
The default directory paths and default wrapper library file names are listed in the following table.

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Library file names</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td><code>/usr/opt/install_path/lib32/</code></td>
<td><code>libdb2teradata.a</code></td>
</tr>
<tr>
<td></td>
<td><code>/usr/opt/install_path/lib64/</code></td>
<td></td>
</tr>
<tr>
<td>Solaris</td>
<td><code>/opt/IBM/db2/install_path/lib32</code></td>
<td><code>libdb2teradata.so</code></td>
</tr>
<tr>
<td></td>
<td><code>/opt/IBM/db2/install_path/lib64</code></td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td><code>%DB2PATH%\bin</code></td>
<td><code>db2teradata.dll</code></td>
</tr>
</tbody>
</table>

`install_path` is the directory path where the federated server is installed on UNIX.

Registering the server definitions for a Teradata data source

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

Procedure

To register a server definition for a Teradata data source:

1. Locate the hosts file.
   - On federated servers that run AIX, the hosts file is located in the `/etc/hosts` directory.
   - On federated servers that run Windows, the hosts file is located in the `%WINDIR%\system32\drivers\etc\hosts` directory.
2. Search the hosts file for the alias of the remote server.
   This alias begins with an alphabetic string and ends with the suffix COP\textit{n}. The value \textit{n} is the number of the application processor that is associated with the Teradata communications processor.
3. Find the first non-numeric field on the line in the hosts that contains the alias.
   Sample hosts file:
   
   ```
   127.0.0.1    localhost
   9.22.5.77    nodexyz    nodexyzCOP1    # teradata server
   9.66.111.133 rtplib05.data.xxx.com aap
   9.66.111.161 rtpscm11.data.xxx.com aapwrt
   9.66.111.161 rtpscm11.data.xxx.com accessm
   ```
   In this example, nodexyz is the node name.
4. To create the server, use one of the following methods:

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
</tbody>
</table>
CREATE SERVER statement - Examples for the Teradata wrapper

Use the CREATE SERVER statement to register server definitions for the Teradata wrapper. This topic includes a complete example with the required parameters, and an example with additional server options.

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

```
CREATE SERVER tera_server TYPE TERADATA
    VERSION 2.5 WRAPPER my_wrapper
    OPTIONS (NODE 'tera_node');
```

**tera_server**
A name that you assign to the Teradata database server. Duplicate server definition names are not allowed.

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

**VERSION 2.5**
The version of the Teradata database server that you want to access.

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

**NODE ‘tera_node’**
The name of the node where the Teradata database server resides. Obtain the node name from the hosts file. This value is case sensitive.

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

**Server options**

When you create a server definition, you can specify additional server options in the CREATE SERVER statement. The server options can be general server options and Teradata-specific server options.

The CPU_RATIO and IO_RATIO server options provide the statistical information about the Teradata server to the query optimizer. To specify that the CPU resources of the federated server are twice as powerful as the CPU resources of the Teradata server, set the value of the CPU_RATIO server option to 2.0. To specify that the I/O devices of the federated server process data three times faster than the I/O devices of the Teradata server, set the IO_RATIO server option to 3.0.
The following example shows a Teradata server definition with these options:

```sql
CREATE SERVER tera_server TYPE TERADATA
  VERSION 2.5 WRAPPER my_wrapper
  OPTIONS (NODE 'tera_node', CPU_RATIO '2.0', IO_RATIO '3.0');
```

### Creating the user mapping for a Teradata data source

When you attempt to access an Teradata server, the federated server establishes a connection to the Teradata server by using a user ID and password that are valid for that data source.

**About this task**

Create a user mapping for each user ID that will access the federated system to send distributed requests to the Teradata data source.

**Procedure**

To create the user mappings for a Teradata data source:

Issue a CREATE USER MAPPING statement.

For example:

```sql
CREATE USER MAPPING FOR local_userID SERVER server_definition_name
  OPTIONS (REMOTE_AUTHID 'remote_userID', REMOTE_PASSWORD 'remote_password')
```

Although the REMOTE_AUTHID and REMOTE_PASSWORD variables are specified as options in the CREATE USER MAPPING statement, these options are required to access Teradata data sources.

After you complete this task, test the connection from the federated server to the Teradata server.

**CREATE USER MAPPING statement - Examples for the Teradata wrapper**

Use the CREATE USER MAPPING statement to map a federated server authorization ID to a remote Teradata user ID and password. This topic includes a complete example with the required parameters, and an example that shows you how to use the DB2 special register USER with the CREATE USER MAPPING statement.

The following example shows how to map a local authorization ID to a remote Teradata user ID and password:

```sql
CREATE USER MAPPING FOR MICHAEL SERVER tera_server
  OPTIONS (REMOTE_AUTHID 'mike', REMOTE_PASSWORD 'passxyz123');
```

**MICHAEL**

Specifies the local authorization ID that you are mapping to the remote user ID and password, which are defined at the Teradata server.

**SERVER tera_server**

Specifies the server definition name that you registered in the CREATE SERVER statement for the Teradata server.

**REMOTE_AUTHID 'mike'**

Specifies the remote Teradata user ID to which you are mapping.
MICHAEL. The value is case-sensitive, unless you set the FOLD_ID server option to 'U' or 'L' in the CREATE SERVER statement.

Although the remote user ID is specified as an option in the CREATE SERVER statement, it is required for Teradata data sources.

REMOTE_PASSWORD 'passxyz123'

Specifies the remote Teradata password that is associated with 'mike'. The value is case-sensitive, unless you set the FOLD_PW server option to 'U' or 'L' in the CREATE SERVER statement.

Although the remote password is specified as an option in the CREATE SERVER statement, it is required for Teradata data sources.

DB2 special register USER

You can use the DB2 special register USER to map the authorization ID of the person who is issuing the CREATE USER MAPPING statement to the data source authorization ID that is specified in the REMOTE_AUTHID user option.

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

```
CREATE USER MAPPING FOR USER SERVER tera_server
    OPTIONS (REMOTE_AUTHID 'mike', REMOTE_PASSWORD 'passxyz123');
```

Testing the connection from the federated server to the Teradata server

Test the connection to the Teradata data source server to determine if the federated server is properly configured to access Teradata data sources.

About this task

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

Procedure

To test the connection to the Teradata server:

Open a pass-through session and issue a SELECT statement on the Teradata system tables. If the SELECT statement returns a count, your server definition and your user mapping are set up properly.

For example:

```
SET PASSTHRU server_definition_name
SELECT count(*) FROM dbc.tables
SET PASSTHRU RESET
```

If the SELECT statement returns an error, you should troubleshoot the connection errors.

After you complete this task, you can register the nicknames for Teradata tables and views.
Troubleshooting data source connection errors

A test connection to the data source server might return an error for several reasons. There are actions that you can take to determine why the error occurred.

**Symptom**

An error is returned when you attempt to connect to the data source.

**Cause**

There are several possible causes for a connection problem.

**Resolving the problem**

To troubleshoot data source connection errors, check the following items for problems:

- Verify that the data source is available.
- If applicable, ensure that the data source server is configured for incoming connections.
- Ensure that your user mapping settings for the REMOTE_AUTHID and REMOTE_PASSWORD options are valid for the connections to the data source. Alter the user mapping, or create another user mapping as necessary.
- If applicable, ensure that the data source client software on the federated server is installed and configured correctly to connect to the data source.
- For ODBC data sources, ensure that the ODBC driver on the federated server is installed and configured correctly to connect to the ODBC data source server. On federated servers that run Windows, use the ODBC Data Source Administrator tool to check the driver. On federated servers that run UNIX, consult the ODBC client vendor’s documentation.
- Verify that the settings for the variables set on the federated server are correct for the data source. These variables include the system environment variables, the variables in the db2dj.ini file, and the DB2 Profile Registry (db2set) variables.
- Check your server definition. If necessary, drop the server definition and create it again.

### Registering nicknames for Teradata tables and views

For each Teradata server definition that you register, you must register a nickname for each table or view that you want to access. Use these nicknames, instead of the names of the data source objects, when you query the Teradata servers.

#### Before you begin

The federated database relies on the data source catalog statistics to optimize query processing. To ensure that the federated database has complete statistics on Teradata tables, use the COLLECT STATISTICS Teradata command before you register a nickname.

From the Teradata server, use the COLLECT STATISTICS Teradata command to collect statistics on one or more columns or indexes in a table.

When you register the nickname with the CREATE NICKNAME statement, the federated database reads the statistics from the Teradata system catalog and updates the local statistics for the nickname.
About this task

When you register a nickname on a Teradata view, the federated database recognizes all columns of the view as nullable, even if the columns in the Teradata view do not allow null values. There is no workaround for this limitation.

Procedure

To register a nickname for a Teradata table or view:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Federated Objects wizard in the DB2 Control Center.</td>
<td>To start the wizard, right-click the Federated Database Objects folder and click <em>Create Federated Objects</em>.</td>
</tr>
</tbody>
</table>
| Issue the CREATE NICKNAME statement. Nicknames can be up to 128 characters in length. | For example:  
CREATE NICKNAME nickname  
FOR server_definition_name.remote_schema.remote_table; |

When you create the nickname, the federated server queries the data source catalog by using the nickname. This query tests the connection to the data source table or view. If the connection does not work, you receive an error message.

Repeat this step for each Teradata table or view that you want to create a nickname for.

**Teradata nicknames on federated servers**

When you query a Teradata data source from a federated server, you use a nickname in the query to identify the Teradata table and view that you want to access.

When you create a nickname for a Teradata table or view, the federated server connects to the Teradata server associated with the table or view. The federated server uses the nickname to verify the connection to the Teradata server. The federated database verifies the presence of the table or view at the data source and then attempts to gather statistical data about the Teradata table or view from the catalog on the Teradata server. The statistics that are gathered about the nicknamed object are stored in the global catalog on the federated server.

The federated server relies on the statistics that it collects for the nicknamed objects to optimize query processing. Because some or all of the Teradata catalog information might be used by the query optimizer, you should update the statistics at the Teradata server before you create a nickname. Update the statistics at the Teradata server by using a command or utility that is equivalent to the DB2 RUNSTATS command.

You cannot submit an INSERT, UPDATE, or DELETE statement to a nickname that references an updatable Teradata view unless the SQL statement can be completely pushed down to the Teradata data source.
CREATE NICKNAME statement - Examples for the Teradata wrapper

Use the CREATE NICKNAME statement to register a nickname for an Teradata table or view that you want to access. This topic includes a complete example with the required parameters.

This example shows how to create a nickname for a Teradata table or view on the Teradata server:

```
CREATE NICKNAME TERASALES FOR tera_server.vinnie.europe ;
```

**TERASALES**

A unique nickname that is used to identify the Teradata table or view.

**Important**: The nickname is a two-part name—the schema and the nickname. If you omit the schema when you register the nickname, the schema of the nickname will be the authorization ID of the user who registers the nickname. The authorization ID is for the federated server, not the remote Teradata data source.

**tera_server.vinnie.europe**

A three-part identifier for the remote object:

- **tera_server** is the server definition name that you assigned to the Teradata database server in the CREATE SERVER statement.
- **vinnie** is the user ID of the owner to which the table or view belongs. This value is case sensitive.
- **europe** is the name of the remote table or view that you want to access.

---

**Enabling run-time linking for libcliv2.so (AIX)**

If you run the djxlinkTeradata.sh file to link to the Teradata shared library called libcliv2.so, you might receive an error message when you issue a CREATE NICKNAME statement.

**Symptom**

An example of an error message that you might receive is:

```
DB21034E The command was processed as an SQL statement because it was not a valid Command Line Processor command. During SQL processing it returned:
SQL30081N A communication error has been detected. Communication protocol being used: "TCP/IP". Communication API being used: "SOCKETS". Location where the error was detected: "9.112.26.28". Communication function detecting the error: "recv". Protocol specific error code(s): "*", "*", "0".
SQLSTATE=08001
```

**Cause**

The OsCall function caused the federated server to stop.

**Resolving the problem**

If you receive an error message, check the /sqllib/db2dump directory for any trap files. Trap file names begin with the letter t and end with a suffix of 000. For example:

```
t123456.000
```
Check the trace information in the trap file for any OsCall function references that indicate that the OsCall function caused the federated server to stop.

The following example shows trace information with an OsCall function reference that you might find in a trap file:

```
*** Start stack traceback ***
0x239690E0 OsCall + 0x28C
0x23973FB0 mtdpassn + 0x8A4
0x239795A4 mtddp + 0x208
0x2395A928 MTDPIO + 0x28C
0x239609C4 CLICON + 0xD50
0x23962350 DBCHCL + 0xC4
```

If you find an OsCall function reference in one of the trap files, you must enable run-time linking for the libcliv2.so shared library.

Issue the following UNIX commands to enable run-time linking:
```
cd /usr/lib
rtl_enable libcliv2.so -F libtli.a
mv libcliv2.so libcliv2.so.old
mv libcliv2.so.new libcliv2.so
chmod a+r libcliv2.so
```

---

**Troubleshooting character sets for Teradata data sources**

When you set the TERADATA_CHARSET environment variable for a Teradata data source, you might encounter errors if the correct character set is not specified.

**Symptom**

If the correct character set is not specified for the Teradata data source, you will encounter the following error:

```
SQL 1822N Unexpected error code "227"
received from data source "<string>". Associated text and tokens are "MTDP: EM_CHARNAME(227): invalid character set name specif". SQLSTATE=560BD
```

**Cause**

The character set that is specified in the TERADATA_CHARSET environment variable is incorrect.

**Resolving the problem**

Verify that the correct character set is installed and specified on the Teradata server and in the db2dj.ini file.

---

**Troubleshooting UPDATE or DELETE operation errors on nicknames**

By default, rows are not uniquely identified on Teradata data source tables. You might receive an SQL error message when you try to update or delete a nickname that is associated with a Teradata table or view.

**Symptom**

The SQL30090N, RC="21" error is returned when you try to update or delete a nickname that is associated with a Teradata table or view.
**Cause**

This problem is caused because rows are not uniquely defined on the Teradata source table.

**Resolving the problem**

To resolve the problem:
1. Drop and recreate the nickname.
2. Create at least one unique index on the Teradata table that is being updated or deleted, and try the operation again.

**Working with Teradata access logging**

The Teradata product provides an access logging feature that generates log entries when Teradata checks the specific security privileges of various users on one or more databases. Although access logging provides considerable and meaningful security information, this feature significantly increases processor usage and can degrade system performance.

If you need to improve system performance, evaluate the checking privilege rules that you defined for access logging. Then, terminate any unnecessary rules by defining END LOGGING statements.

For the best performance, turn off all access logging. Drop the Teradata DBC.AccLogRules macro and then force a trusted parallel application (TPA) reset to stop access logging completely.

See the Teradata documentation for more information.
Chapter 18. Configuring access to Web services data sources

To configure the federated system to access Web services data sources, you must provide the federated server with information about the data sources and objects that you want to access, such as a valid Web services description language (WSDL) document.

Before you begin

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

You can configure the federated server to access Web services data sources by using the DB2 Control Center or the DB2 command line. The DB2 Control Center includes a wizard to guide you through the steps that are required to configure the federated server.

Procedure

To configure access to Web services data sources:
1. Register the Web services wrapper
2. Register the server definition for Web services data sources
3. Register user mappings to enable security for HTTP authentication (optional)
4. Register nicknames for Web services data sources:
   - Register nicknames for Web services data sources by using the DB2 command line
   - Register nicknames for Web services data sources by using the DB2 Control Center
5. Create federated views for Web services nicknames

Web services and the Web services wrapper

Web service providers are described by Web Services Description Language (WSDL) documents. You can use the Web services wrapper to access Web service providers.

The Figure 15 on page 304 diagram shows the architecture of Web services.
1. A Web service provider implements a service and publishes the WSDL information to a service broker, such as UDDI.
2. The service consumer can then use the service broker to find a Web service provider.
3. When the service consumer finds a Web service provider, the service consumer binds to the service provider so that the consumer can use the Web service.
4. The consumer invokes the service by exchanging SOAP (simple object access protocol) messages between the requester and provider.
The SOAP specification defines the layout of an XML-based message. A SOAP message is contained in a SOAP envelope. The envelope consists of an optional SOAP header and a mandatory SOAP body. The SOAP header can contain information about the message, such as encryption information or authentication information. The SOAP body contains the message. The SOAP specification also defines a default encoding for programming language bindings, which is called the SOAP encoding.

**The WSDL document and the Web service**

The key to the Web service is the WSDL document. The WSDL document is an XML document that describes Web services in terms of the messages that it sends and receives. Messages are described by using a type system, which is typically the XML schema. A Web service operation associates a message exchange pattern with one or more messages. A message exchange pattern identifies the sequence and cardinality of messages that are sent or received, as well as who the messages are logically sent to or received from. An interface groups together operations without any commitment to the transport or wire format. A WSDL binding specifies transport and wire format details for one or more interfaces. An endpoint associates a network address with a binding. A service groups together endpoints that implement a common interface. The messages can contain document-oriented information or process-oriented information, which is also known as remote procedure calls (RPC). A WSDL document can contain one or more Web services.

The example in [Figure 16 on page 305](#) shows the WSDL definition of a simple service that provides stock quotes. The Web service supports a single operation that is named GetLastTradePrice. The service can be accessed with the SOAP 1.1 protocol over HTTP. The request reads a ticker symbol as input, which is a string data type, and returns the price, which is a float data type. The string and float data types are predefined types in the XML schema standards. A Web service can also define data types and use those user-defined data types in messages. Predefined and user-defined XML data types map to columns of the nicknames. The complete example and the WSDL specification is at the W3C Web site.
<?xml version="1.0"?>
<definitions name="StockQuote"
...

<types>
  <schema targetNamespace="http://example.com/stockquote.xsd"
    xmlns="http://www.w3.org/2000/10/XMLSchema">
    <element name="TradePriceRequest">
      <complexType>
        <all>
          <element name="tickerSymbol" type="string"/>
        </all>
      </complexType>
    </element>
  </schema>
</types>

<message name="GetLastTradePriceInput">
...
</message>

<portType name="StockQuotePortType">
  <operation name="GetLastTradePrice">
    <input message="tns:GetLastTradePriceInput"/>
    <output message="tns:GetLastTradePriceOutput"/>
  </operation>
</portType>

<binding name="StockQuoteSoapBinding"
  type="tns:StockQuotePortType">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetLastTradePrice">
    <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>

<service name="StockQuoteService">
  <documentation>My first service</documentation>
  <port name="StockQuotePort" binding="tns:StockQuoteBinding">
    <soap:address location="http://example.com/stockquote"/>
  </port>
</service>
</definitions>

Figure 16. Example of a WSDL document

Chapter 18. Configuring access to Web services data sources 305
The WSDL document, Web services wrapper, and nicknames

The Web services wrapper uses the operations in a port type that have a SOAP binding with an HTTP transport. The input messages in the operation, and the associated types or elements become columns in the nickname. The output messages in the operation are extracted into the nickname hierarchy. You can create a separate hierarchy of nicknames for each operation in the WSDL document.

By using the Web services wrapper, you can use the federated systems functions to join data from Web services with data on other federated data sources.

The example [Figure 17 on page 307] uses a WSDL document that contains a portType with an operation name of GETTEMP. With this Web service, you enter a zip code as input and receive a temperature for that zip code.
The input value is described by the zipcode column of the nickname. The output value is described by the return column of the nickname. In the WSDL document, those columns are identified in the messages element. The messages element represents the logical definition of the data that is sent between the Web service provider and the Web service consumer. If more explanation of the information in the message element is needed, then the WSDL document can also contain a type element. The type element can refer to predefined types that are based on the XML schema specifications or types that are defined by a user.

The example Figure 18 on page 308 shows the nickname that the DB2® Control Center Discovery tool produces from the GETTEMP Web service WSDL document. The zipcode column is a required input column because of the nickname TEMPLATE syntax:

```xml
<?xml version="1.0"?>
<definitions name="TemperatureService" targetNamespace=http://www.xmethods.net/sd/TemperatureService.wsdl
xmlns:tns="http://www.xmethods.net/sd/TemperatureService.wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
<message name="getTempRequest">
  <part name="zipcode" type="xsd:string"/>
</message>
<message name="getTempResponse">
  <part name="return" type="xsd:float"/>
</message>
<portType name="TemperaturePortType">
  <operation name="getTemp">
    <input message="tns:getTempRequest"/>
    <output message="tns:getTempResponse"/>
  </operation>
</portType>
<binding name="TemperatureBinding" type="tns:TemperaturePortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="getTemp">
    <input>
      <soap:body use="encoded" namespace="urn:xmethods-Temperature" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:xmethods-Temperature" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
  </operation>
</binding>
<service name="TemperatureService">
  <documentation>
    Returns current temperature in a given U.S. zipcode
  </documentation>
  <port name="TemperaturePort" binding="tns:TemperatureBinding">
    <soap:address>
      <location>http://services.xmethods.net:80/soap/servlet/rpcrouter</location>
    </soap:address>
  </port>
</service>
</definitions>

Figure 17. GETTEMP Web service

The input value is described by the zipcode column of the nickname. The output value is described by the return column of the nickname. In the WSDL document, those columns are identified in the messages element. The messages element represents the logical definition of the data that is sent between the Web service provider and the Web service consumer. If more explanation of the information in the message element is needed, then the WSDL document can also contain a type element. The type element can refer to predefined types that are based on the XML schema specifications or types that are defined by a user.

The example Figure 18 on page 308 shows the nickname that the DB2® Control Center Discovery tool produces from the GETTEMP Web service WSDL document. The zipcode column is a required input column because of the nickname TEMPLATE syntax:
CREATE NICKNAME GETTEMP ( 
    ZIPCODE VARCHAR (48) OPTIONS(TEMPLATE 'column'),
    RETURN VARCHAR (48) OPTIONS(XPATH './return/text()')
 )
FOR SERVER "EHPWSSERV"
    OPTIONS(URL 'http://services.xmethods.net:80/soap/servlet/rpcrouter',
            SOAPACTION ' ',
            TEMPLATE '<soapenv:Envelope>
                <soapenv:Body>
                    <ns2:getTemp>
                        <zipcode>&zipcode[1,1]</zipcode>
                    </ns2:getTemp>
                </soapenv:Body>
            </soapenv:Envelope>',
            XPATH '/soapenv:Envelope/soapenv:Body/*'
    )
NAMESPACES ' ns1="http://www.xmethods.net/sd/TemperatureService.wsdl",
            ns2="urn:xmethods-Temperature",
            soapenv="http://schemas.xmlsoap.org/soap/envelope";

Figure 18. GETTEMP nickname

The nickname options in the Web services wrapper, URL and SOAPACTION, provide the ability to override the endpoint, or the address that you specified when you created the nickname. When you use the URLCOLUMN or SOAPACTIONCOLUMN enabled columns in a query, you can use dynamic addresses with the same nicknames. If you define the nickname options URL and SOAPACTION when you create a nickname and enable the URLCOLUMN and SOAPACTIONCOLUMN on the column option, then you are using the late binding functions of Web services wrappers. The value for the SOAPACTION nickname option becomes an attribute in the HTTP header. The value for the URL nickname option is the HTTP URL to which the request is sent.

The URL and SOAPACTION nickname options provide dynamic nickname associations. These dynamic addresses are useful if several companies implement a Web service portType. The Web services wrapper requires that the only differences between the WSDL documents are different URLs and SOAPACTIONS. You can use the late binding function to create and use the same nickname for different service endpoints that different companies might want to use. The URL and SOAPACTION values are derived from the WSDL document.

The following example shows how you can use the URLCOLUMN and SOAPACTIONCOLUMN column options:

CREATE NICKNAME GetPartQuote( 
    partnumber INTEGER OPTIONS (TEMPLATE 'column'),
    price FLOAT OPTIONS (XPATH './price'),
    uricol VARCHAR(100) OPTIONS (URLCOLUMN 'Y'),
    soapactioncol VARCHAR(100) OPTIONS (SOAPACTIONCOLUMN 'Y'),
) 
FOR SERVER myServer
    OPTIONS ( 
        SOAPACTION 'http://example.com/GetPartPrice',
        ... )

Figure 19. GetPartQuote nickname
The following example uses the columns URLCOL and SOAPACTIONCOL that were defined with the URLCOLUMN column option enabled and the SOAPACTIONCOLUMN column option enabled:

```sql
SELECT * FROM supplier_endpoints p,
      GetPartQuote q
WHERE partnumber=1234 AND
  p.url=q.urlcol AND
  p.soapaction=q.soapactioncol;
```

The SQL application can defer choosing which endpoints to use until the time that a query is run, instead of defining a specific endpoint at the time that the nickname is created.

The Web services wrapper can separate a large amount of WSDL document data into fragments to decrease the total memory that is used. Specify the STREAMING option in the DB2 Control Center in the Settings page of the Properties window, when you create a Web services nickname. The Web services wrapper processes the resulting stream of XML data and then extracts the information that is requested by a query fragment. The Web services wrapper parses one fragment at a time. Use the STREAMING option to parse large XML documents only.

### Registering the Web services wrapper

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

#### Before you begin

See the list of Web services wrapper library files for the correct name to specify in the CREATE WRAPPER statement.

The name of the wrapper library file that you specify depends on the operating system of the federated server.

#### Procedure

To register a wrapper:

Issue a CREATE WRAPPER statement with the name of the wrapper and the name of the wrapper library file. For example, to register a wrapper with the name websr_wrapper on a federated server that uses Windows, issue the following statement:

```
CREATE WRAPPER websr_wrapper LIBRARY 'db2ws.d11';
```

### Web services wrapper library files

The Web services wrapper library files are added to the federated server when you install the federated server.

When you install IBM WebSphere Federation Server, library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files added to the directory path are libdb2ws.a, libdb2wsF.a, and libdb2wsU.a. The default wrapper library file is libdb2ws.a. The other wrapper library files are used with specific wrapper options.
You must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.

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

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Wrapper library file</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/&lt;install_path&gt;/lib32</td>
<td>libdb2ws.a</td>
</tr>
<tr>
<td></td>
<td>/usr/opt/&lt;install_path&gt;/lib64</td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib32</td>
<td>lsb.db2ws.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib64</td>
<td></td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib32</td>
<td>lsb.db2ws.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib64</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>%DB2PATH%\bin</td>
<td>db2ws.dll</td>
</tr>
</tbody>
</table>

- `<install_path>` is the directory path where the federated server is installed on UNIX or Linux.
- `%DB2PATH%` is the environment variable that is used to specify the directory path where the federated server is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

### Registering the server definition for Web services data sources

You can register a server definition by using the DB2 Control Center or by using the DB2 command line. This task describes how to register a Web services server definition from the DB2 command line.

A server definition must be registered for each Web service that you want to access.

**Procedure**

To register a server definition to the federated system for the Web services wrapper, issue the CREATE SERVER statement.

For example, to register a Web services server definition named `ws_server` on Windows, issue the following statement:

```sql
CREATE SERVER ws_server WRAPPER websr_wrapper;
```

You can set optional timeout and proxy server parameters for the CREATE SERVER statement.

**CREATE SERVER statement - Examples for Web services wrapper**

Use the CREATE SERVER statement to register server definitions for the Web services wrapper with time out and proxy server settings.

Even if you are not using a proxy server to access Web services documents, you must still register a server definition. The hierarchy of federated objects requires that the Web services files are associated with a specific server definition object. The statement that you issue to register a server definition is:
CREATE SERVER my_server WRAPPER my_wrapper
   OPTIONS (TIMEOUT '60');

my_server
   A name that you assign to the Web services server definition. Duplicate
   server definition names are not allowed.

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

TIMEOUT '60'
   Specifies the time, in minutes, that the federated server should wait for a
   network transfer and the computation of a result.

Server definitions when a proxy server is used

You must use the proxy server options in the CREATE SERVER statement if all of
the following conditions are true:
   • You want to retrieve data by using a URI.
   • The URI retrieves data from behind a firewall, through a proxy.
   • The firewall or proxy is HTTP or SOCKS.

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

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

Registering a server definition for an HTTP proxy server

To register a server definition and specify an HTTP proxy server, use the following
statement:
CREATE SERVER ws_server_http
   WRAPPER ws_wrapper
   OPTIONS (PROXY_TYPE 'HTTP', PROXY_SERVER_NAME 'proxy_http',
            PROXY_SERVER_PORT '8080');

ws_server_http
   A name that you assign to the Web services server definition. Duplicate
   server definition names are not allowed.

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

PROXY_TYPE 'HTTP'
   Specifies the proxy type that is used to access the Internet when behind a
   firewall.

PROXY_SERVER_NAME 'proxy_http'
   Specifies the proxy server name or IP address.

PROXY_SERVER_PORT '8080'
   Specifies the proxy server port number.
Registering a server definition for a SOCKS proxy server with authentication information

To register a server definition and specify a SOCKS proxy server with authentication information, issue this statement:

```
CREATE SERVER ws_server_socks
  WRAPPER ws_wrapper
  OPTIONS (PROXY_TYPE 'SOCKS', PROXY_SERVER_NAME 'proxy_socks',
           PROXY_SERVER_PORT '1081', PROXY_AUTHID 'Sampson',
           PROXY_PASSWORD 'sailing4me');
```

**PROXY_AUTHID 'Sampson'**

Specifies the user name on the proxy server.

**PROXY_PASSWORD 'sailing4me'**

Specifies the password on the proxy server that is associated with the user name Sampson.

Enabling security through the Web services wrapper

The Web services wrapper supports HTTP authentication by using the CREATE USER MAPPING statement.

The Web services wrapper supports HTTPS as a transport protocol for SOAP messages. The WSDL document that is generated by the Web services provider contains https:// in the URL. Thus, the SOAP messages in the HTTP request or HTTP response are encrypted.

If the Web service uses HTTPS as a transport protocol, you can configure the wrapper to validate the SSL certificates that the server sends for identification by using the SSL_VERIFY_SERVER_CERTIFICATE wrapper or server option. The Web service wrapper can call Web services with self-signed certificates.

The user mapping is optional. If you do not specify a user mapping, you might see an error if the Web service provider expects authentication information. Some servers might use authentication to restrict access to a service. The need for authentication is not apparent from the information in the WSDL document.

**Procedure**

To map a federated server user ID to a Web services user ID and password:

Issue the CREATE USER MAPPING statement.

For example, with the following CREATE USER MAPPING statement, when the Web services nickname on the S1 server is accessed, the HTTP request is sent with SYSTEM as the user ID and MANAGER as the password.

```
CREATE USER MAPPING
  FOR RSPALTEN SERVER S1
  OPTIONS (REMOTE_AUTHID 'SYSTEM', REMOTE_PASSWORD 'MANAGER',
           PROXY_AUTHID 'ID', PROXY_PASSWORD 'PWD',
           SSL_CLIENT_CERTIFICATE_LABEL 'LABEL');
```
Registering nicknames for Web services data sources

For each Web services server definition that you register, you must register a nickname for each data source that you want to access. You can register a nickname by using the command line or the DB2 Control Center.

To register nicknames for Web services data sources:

Select one of the following methods:

- "Registering nicknames for Web services data sources (DB2 command line)"
- "Registering nicknames for Web services data sources (DB2 Control Center)" on page 314

Registering nicknames for Web services data sources (DB2 command line)

For each Web services server definition that you register, you must register a nickname for each data source that you want to access. Use these nicknames instead of the names of the data sources when you query the Web services data sources.

Before you begin

You must have access to a valid WSDL document that describes the Web service with which you want to communicate.

Restrictions

- Only request-response operations are supported.
- A SOAP binding with an HTTP transport is the only binding that is supported.
- You must use either the TEMPLATE option or the XPATH option on each column, except for the special columns with the SOAPACTIONCOLUMN, URLCOLUMN, PRIMARY_KEY, or FOREIGN_KEY options.

You create one nickname hierarchy for each Web service operation that is defined in the Web services description language (WSDL) document. The parent nickname identifies the SOAP envelope; child nicknames identify the SOAP body elements and the optional SOAP header.

Procedure

To register nicknames for Web services data sources from the DB2 command line:

Issue a CREATE NICKNAME statement. For example, to register the nicknames on Windows for a Web service named GETTEMP, issue the following statement:

```
CREATE NICKNAME GETTEMP
  ZIPCODE VARCHAR (48) OPTIONS (TEMPLATE ' &column'),
  RETURN VARCHAR (48) OPTIONS (XPATH './return/text()')
) FOR SERVER "EHPWSSERV"
  OPTIONS (URL 'http://services.xmethods.net:80/soap/servlet/rpcrouter',
    SOAPACTION ' ',
    TEMPLATE '<soapenv:Envelope>
      <soapenv:Body>
        <ns2:getTemp>
          <zipcodeaddGap[1,1]
```
Registering nicknames for Web services data sources (DB2 Control Center)

For each Web services server definition that you register, you must register a nickname for each data source that you want to access. Use these nicknames instead of the names of the data sources when you query the Web services data sources.

Before you begin

You must have access to a valid WSDL document that describes the Web service with which you want to communicate.

Restrictions

- Only request-response operations are supported.
- A SOAP binding with an HTTP transport is the only binding that is supported.
- You must use either the TEMPLATE option or the XPATH option on each column, except for the special columns with the SOAPACTIONCOLUMN, URLCOLUMN, PRIMARY_KEY or FOREIGN_KEY options.

You create one nickname hierarchy for each Web service operation that is defined in the Web services description language (WSDL) document. The parent nickname identifies the SOAP envelope; child nicknames identify the SOAP body elements and the optional SOAP header.

In the DB2 Control Center, you can use the Discovery tool to create the nicknames. The input to the Discovery tool is a URL of the location of a WSDL document. The Discovery tool creates nicknames as a result of processing the WSDL document. The DB2 Control Center generates unique nicknames by grouping part names or element names with the column name from the WSDL document. The WSDL document can contain schema definitions that are embedded in the WSDL file or in an external XML schema file that is imported in the WSDL file. These schema definitions are imported by using a URL address.

When the DB2 Control Center generates a child nickname that is only used for input, the XPATH statement contains a period, such as in the following example: XPATH ".".

Procedure

To register nicknames for Web services data sources from the DB2 Control Center:
1. Expand the Federated Database Objects folder.
2. Expand the wrapper folder for which you want to register nicknames.
3. Expand the Server Definitions folder.
4. Expand the server folder for which you want to register nicknames.
5. Right click the Nicknames folder and select Create.
6. In the Create Nicknames window, click Discover to define search criteria to help you select objects at the data source.

7. Specify the WSDL document that contains the definition of the Web service that you want federated system users to access. The WSDL document can be a local document or you can specify its location by using a URL.

8. Click OK to create the nickname according to the selected WSDL document.

The DB2 Control Center extracts the WSDL document into multiple create-nickname DDL statements, with the appropriate parent-child relationship definitions. The data definition language (DDL) statement that the DB2 Control Center generates maps all input elements to columns of the root nickname in the nickname hierarchy.

**CREATE NICKNAME statement – examples for the Web services wrapper**

When you create a nickname to access a Web service, you create an input column for each value in the input message of a Web service operation and an output column for each value in the output message of a Web service operation. You control the input and output column definitions with the nickname column option definitions.

The TEMPLATE column option specifies that a column is an input column. The XPATH column option specifies that a column is an output column. When the TEMPLATE nickname option contains a bracketed notation ([1,1]), the column is a required input column.

The NAMESPACES nickname option is a comma-separated list of name-value pairs that a federated system uses to resolve the namespaces for elements in input and output XML documents. The namespaces are used in the message request so that the prefixes in the TEMPLATE nickname option are defined. The NAMESPACES nickname option resolves the prefixes in XPath expressions with the namespace URLs that are defined in the WSDL or XML schemas. The XPath expressions are applied on the XML document that is returned from the Web service.

**Example 1: Required input columns**

The following example shows a nickname for a Web service named getQuote. The Web service reads a stock ticker symbol and returns a trading price. The following DDL statement is created by the Discovery tool in the DB2 Control Center.

```sql
CREATE NICKNAME "stockquote.stockquoteport_getquote_nn" ( symbol VARCHAR (48) OPTIONS(TEMPLATE '&column'), result VARCHAR (48) OPTIONS(XPATH './Result/text()')) FOR SERVER "xmethods_server" OPTIONS(
    URL 'http://66.28.98.121:9090/soap',
    SOAPACTION 'urn:xmethods-delayed-quotes#getQuote',
    TEMPLATE '<soapenv:Envelope>
        <ns2:getQuote>
            <symbol>&symbol[1,1]</symbol>
        </ns2:getQuote>
    </soapenv:Envelope>',
    XPATH '/soapenv:Envelope/ns2:getQuote/*',
    NAMESPACES 'ns2="urn:xmethods-delayed-quotes",'
                     ns1="http://www.example.com/wsd1/net.xmethods.services.stockquote.StockQuote/*",'
    soapenv="http://schemas.xmlsoap.org/soap/envelope/"');
```
The nickname TEMPLATE option specifies the column SYMBOL as a required input column, because the column contains the [1,1] designation. In the nickname TEMPLATE option, the complete SOAP envelope is specified for the Web service. The getQuote input value is contained in the SOAP envelope and body elements. The XPATH nickname option contains the information to find the trading price value through the SOAP envelope and body tags.

Use the stockquote.stockquoteport_getquote_nn nickname to access the Web service, such as in the following query:

```sql
SELECT * FROM "stockquote.stockquoteport_getquote_nn"
WHERE symbol='IBM';
```

You must use the predicate, `symbol='IBM'`, in this statement because `symbol` is a required input column. The equality predicate is the only valid predicate on input columns. Each of the equality predicates sets a value in the input message. If the input column is optional, an equality predicate on that column is not necessary. If the input column is required, then you must issue the query with an equality predicate. You can use a literal value such as IBM in an equality expression or a value from a joined table or nickname.

**Example 2: Repeating elements and child nicknames**

The following example uses a Web service named getZooReport that produces a report for zoos. The input value is a zoo identifier. The output value is a report that is described by the following schema:
The following DDL statement is generated by the DB2 Control Center Discovery tool that is based on the WSDL that contains the schema:

```xml
<wSDL:definitions name="Name" targetNamespace="http://myzoo.com"
...>
<wSDL:types>
<xSD:schema elementFormDefault="qualified" targetNamespace="http://myzoo.com"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xSD:element name="Animal">
  <xSD:complexType>
    <xSD:sequence>
      <xSD:element ref="tns:Name"/>
      <xSD:element ref="tns:Species"/>
      <xSD:element ref="tns:Lot"/>
    </xSD:sequence>
  </xSD:complexType>
</xSD:element>
<xSD:element name="AnimalCareList">
  <xSD:complexType>
    <xSD:sequence>
      <xSD:element maxOccurs="unbounded" minOccurs="1" ref="tns:Animal"/>
    </xSD:sequence>
  </xSD:complexType>
</xSD:element>
<xSD:element name="Count" type="xsd:string"/>
<xSD:element name="LastModified" type="xsd:string"/>
<xSD:element name="Name" type="xsd:string"/>
<xSD:element name="NumberOfCages" type="xsd:string"/>
<xSD:element name="Species" type="xsd:string"/>
<xSD:element name="Zoo">
  <xSD:complexType>
    <xSD:sequence>
      <xSD:element ref="tns:ZooName"/>
      <xSD:element ref="tns:Count"/>
      <xSD:element ref="tns:LastModified"/>
      <xSD:element maxOccurs="unbounded" minOccurs="0" ref="tns:Zookeeper"/>
    </xSD:sequence>
    <xSD:attribute name="id" type="xsd:string" use="optional"/>
  </xSD:complexType>
</xSD:element>
<xSD:element name="ZooName" type="xsd:string"/>
<xSD:element name="Zookeeper">
  <xSD:complexType>
    <xSD:sequence>
      <xSD:element ref="tns:Name"/>
      <xSD:element ref="tns:NumberOfCages"/>
      <xSD:element ref="tns:AnimalCareList"/>
    </xSD:sequence>
    <xSD:attribute name="id" type="xsd:string" use="optional"/>
  </xSD:complexType>
</xSD:element>
</xsd:schema>
</wSDL:types>
...>

Figure 20. getZooReport Web service

The following DDL statement is generated by the DB2 Control Center Discovery tool that is based on the WSDL that contains the schema:
The schema includes some elements that are repeated, known as sequence elements. These repeated elements become child nicknames of the parent nickname, as shown in Figure 21, Figure 22, and Figure 23. For example, zooname,
count, lastmodified, and zookeeper are all elements of zoo. The element zoo contains 0 or more zookeeper elements. The root nickname, zoo, contains the columns zooname, count, and lastmodified. A child nickname, zookeeper, is created by the DB2 Control Center Discovery tool to describe the repeating elements of zookeeper. The third element in the zookeeper column, animalcarelist, also contains 0 or more elements and so it becomes a child nickname, zooport_getzooreport_report_report_nn. The following figure shows the nickname hierarchy:

---

**Root nickname: zooport_getzooreport_nn**
- Zoo (parent):
  - Zooname
  - Count
  - LastModified
  - ZooKeeper (there are 0 or more ZooKeeper elements)

**Child nickname: zooport_getzooreport_report_nn**
- ZooKeeper elements
  - Name
  - NumberOfCages
  - AnimalCareList (there are 0 or more Animal elements)

**Child nickname: zooport_getzooreport_report_report_nn**
- Animal
  - Name
  - Species
  - Lot

---

**Figure 24. Parent —> Child —> nickname hierarchies**

The following statement is a typical query that you might issue on the nicknames to access the zoo report Web service. When you issue this statement, you retrieve the information from the zoo report based on a specific identifier and on where the primary and foreign keys of the child nickname zoo reports match.

```sql
SELECT * FROM zooport_getzooreport_nn ,
   zooport_getzooreport_report_nn zk ,
   zooport_getzooreport_report_report_nn a
WHERE zooid='1' AND zooport_getzooreport_pkey=zk.zooport_getzooreport_fkey
   and zk.zooport_getzooreport_pkey=a.zooport_getzooreport_fkey;
```

---

**Example 3: Late binding**

The following example shows how you can use the late binding option. You can use this option from the DB2 Control Center or from a DB2 command line. If you define the nickname options URL and SOAPACTION and if you enable the column options URLCOLUMN and SOAPACTIONCOLUMN when you create a nickname, you are using the late binding functions. The DB2 Control Center creates the column options, URLCOLUMN and SOAPACTIONCOLUMN, and sets the values of the columns to yes.

The following example is for a Web service that provides price quotes for parts that is implemented by all suppliers for a company. Here is the CREATE NICKNAME statement that includes the URLCOLUMN and SOAPACTIONCOLUMN definitions:
CREATE NICKNAME GetPartQuote(
    partnumber INTEGER OPTIONS (TEMPLATE('&column'),
    price FLOAT OPTIONS (XPATH './price')),
    urlcol VARCHAR(100) OPTIONS (URLCOLUMN 'Y'),
    soapactioncol VARCHAR(100) OPTIONS (SOAPACTIONCOLUMN 'Y'),
    FOR SERVER myServer
    OPTIONS (...
        SOAPACTION 'http://example.com/GetPartPrice',
        ...
    )
)

To get price quotes from all of the suppliers with a single query, the values that each supplier uses for the SOAPACTION and URL column options are needed. The query looks like this:

SELECT * FROM supplier_endpoints p,
    GetPartQuote q
WHERE partnumber=1234 AND
    p.url=q.urlcol AND
    p.soapaction=q.soapactioncol;

Local table supplier_endpoints contains all of the URLs and SOAP addresses with which you can call the Web service. You can include an ORDER BY price clause to determine the least expensive supplier for this part.

**Example 4: ESCAPE_INPUT column option**

You can include XML fragments as input values in your query. When you register a nickname, include the column option ESCAPE_INPUT=N. This option maintains the special characters, such as angle brackets (< and >) in XML fragments in the input values.

When a schema contains repeating input values that requires you to send XML as part of the SOAP message, you can use the ESCAPE_INPUT column option to build the output message with the correct XML.

For example, the zoo Web service includes an operation to add a new zoo keeper and the animals that are associated with that zoo keeper. In the schema for this example, an AnimalCareList can have multiple animals.

CREATE NICKNAME add_zookeeper(
    zookeeper_id VARCHAR(48) OPTIONS(TEMPLATE '...'),
    name VARCHAR(48) OPTIONS(TEMPLATE '...'),
    numberofcages VARCHAR(48) OPTIONS(TEMPLATE '...'),
    animals VARCHAR(3000) OPTIONS( TEMPLATE '...', ESCAPE_INPUT 'N')
    ...)

To add a new zoo keeper with two animals, issue a query such as the following example:

SELECT * FROM add_zookeeper
WHERE zookeeper_id='37' AND
    name='Amit Kapoor' AND
    numberofcages='3' AND
    animals='&lt;AnimalCareList xmlns="http://myzoo.com">  
    &lt;Animal&gt;
    &lt;Name&gt;Larry&lt;/Name&gt;
    &lt;Species&gt;Gorilla&lt;/Species&gt;
    &lt;Lot&gt;7&lt;/Lot&gt;
    &lt;/Animal&gt;
    &lt;Animal&gt;
    &lt;Name&gt;Bill&lt;/Name&gt;
The add_zookeeper nickname is a Web service operation that can change the state of the Web service, or update information. Although nonrelational wrappers cannot be updated, the SELECT statement in this example updates the zoo information to add a new zoo keeper.

You can also use the ESCAPE_INPUT column option for a schema with an element such as xsd:anyType. In this case, the type of the element is unknown. You can use the ESCAPE_INPUT column option on the input column for that element so that you can specify arbitrary XML fragments for your input.

**SOAP header access**

Use the Web Service wrapper to access Simple Object Access Protocol (SOAP) header values that are embedded in a SOAP message.

A SOAP message, either a request or response, consists of an envelope that contains the namespaces that are used by the rest of the SOAP message, an optional header, and a body. A SOAP header typically contains metadata about the SOAP message or other optional information. The SOAP message travels from a source to a destination by passing through a series of intermediary services that process the message or its side-effects. For example, a message may pass through a transaction service that provides the client with guaranteed invocation if the network fails, or a security service that is located at an enterprise portal may provide authentication information. These intermediaries often work on the metadata contained in the SOAP header.

SOAP headers often contain optional information or support evolving user interfaces. For example, a bank allows customers to use ATM cards to withdraw money from their checking accounts. However, some customers have more than one account. If a customer has multiple accounts, the bank must be able to identify the account from which the withdrawal is made.

Using this example, the following is the header element that specifies a bank account:

<xsd: element name="AccountID" type="xsd:int"/>

The following is a SOAP message that uses this header element. In this case, the customer selected to withdraw $1,000 from account 1 and incurred a transaction fee of 50 cents.

<SOAP-ENV:Envelope>
  <SOAP-ENV:Header>
    <tns:AccountID>1</tns:AccountID>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tns:Action>withdrawal</tns:Action>
    <tns:Amount>1000</tns:Amount>
    <tns:Fee>0.50</tns:Fee>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

SOAP headers have these requirements:
- soapenv:Header is an optional element under soapenv:Envelope.
- If it is used, the SOAP header must be the first element.
• All children under soapenv:Header must be namespace-qualified.
• A soapenv:Header can have these attributes: actor, mustUnderstand, and encoding style.

The nickname hierarchy for accessing SOAP headers requires that the root nickname point to the SOAP envelope. The header and the body elements are child nicknames of the root nickname. You can create the nicknames yourself or use the Control Center to generate the nicknames automatically. Then you can create queries that access the information that is stored in the SOAP header.

Example

These CREATE NICKNAME statements illustrate how to create root, body, and header nicknames that use this WSDL file.

```
CREATE NICKNAME STOCKSERVICE_GETLASTSELLPRICE_ROOT_NN
(NN_PKEY VARCHAR(16) NOT NULL OPTIONS(PRIMARY_KEY 'YES'))
FOR SERVER "WSSERVER_HDR"
OPTIONS
SOAPACTION ' ', TEMPLATE
<soapenv:Envelope>&STOCKSERVICE_GETLASTSELLPRICE_HEADER_NN[1,1]
&STOCKSERVICE_GETLASTSELLPRICE_BODY_NN[1,1]</soapenv:Envelope>', XPATH
'/soapenv:Envelope',
NAMESPACES 'soapenv="http://schemas.xmlsoap.org/soap/envelope/",
nsl="http://soapheader.ibm.com"');

CREATE NICKNAME STOCKSERVICE_GETLASTSELLPRICE_HEADER_NN
(NN_PKEY VARCHAR(16) NOT NULL OPTIONS(PRIMARY_KEY 'YES'),
NN_FKEY VARCHAR(16) NOT NULL OPTIONS(FOREIGN_KEY
'STOCKSERVICE_GETLASTSELLPRICE_ROOT_NN'),
QUOTE_TIMESTAMP VARCHAR(30) OPTIONS(TEMPLATE '&column'))
FOR SERVER "WSSERVER_HDR"
OPTIONS
(XPATH './soapenv:Header', TEMPLATE '<soapenv:Header>
<quote_timestamp>&QUOTE_TIMESTAMP[1,1]
</quote_timestamp></soapenv:Header>',
nsl="http://soapheader.ibm.com"');

CREATE NICKNAME STOCKSERVICE_GETLASTSELLPRICE_BODY_NN
(NN_PKEY VARCHAR(16) NOT NULL OPTIONS(PRIMARY_KEY 'YES'),
NN_FKEY VARCHAR(16) NOT NULL OPTIONS(FOREIGN_KEY
'STOCKSERVICE_GETLASTSELLPRICE_ROOT_NN'),
GETLASTSELLPRICE_TICKER VARCHAR(48)
OPTIONS(TEMPLATE '&column'), GETLASTSELLPRICERETURN VARCHAR(48)
OPTIONS(XPATH
','/ns1:getLastSellPriceResponse/ns1:getLastSellPriceReturn/text()')
FOR SERVER "WSSERVER_HDR"
OPTIONS
(TEMPLATE '<soapenv:Body><ns1:getLastSellPrice>
<ns1:ticker>&GETLASTSELLPRICE_TICKER[1,1]
</ns1:ticker></ns1:getLastSellPrice></soapenv:Body>',
XPATH './soapenv:Body',
NAMESPACES 'soapenv="http://schemas.xmlsoap.org/soap/envelope/" ,
nsl="http://soapheader.ibm.com" ');
```

The following query returns the price of IBM stock for a specific date and time:

```
SELECT * from
STOCKSERVICE_GETLASTSELLPRICE_ROOT_NN as a,
STOCKSERVICE_GETLASTSELLPRICE_HEADER_NN as b,
STOCKSERVICE_GETLASTSELLPRICE_BODY_NN as c
where
```
Nicknames and XPath expressions

The WebSphere Business Integration wrapper and the Web services wrapper use nicknames that correspond to the tree structure of XML data. Each nickname is defined by XPath expressions that represent output values.

Nicknames correspond to the tree structure of your XML document data. Parent nicknames and child nicknames correspond to the root structure and nested elements of the data tree structure. These parent and child nicknames are connected by primary and foreign keys that are specified with the CREATE NICKNAME statement.

Each nickname is defined by XPath expressions that represent output values. The WebSphere Business Integration wrapper and the Web services wrapper use XPath expressions to establish a correspondence between the data in an XML document and the rows in a relational table. These XPath expressions identify the values in the XML document and determine how these values correspond to the columns of each row. The WebSphere Business Integration wrapper and the Web services wrapper read the XML document data only. The wrappers do not update the data. For the Web services wrapper, the XPath option contains the information to find the SOAP messages through the SOAP envelope and SOAP body tags. The getQuote message is contained in the SOAP envelope and body elements.

The XPath expression for the NICKNAME option points to repeating tags that are in the output element. The XPath expression determines how many or which rows will be in the nickname. The column option XPath expression is relative to the NICKNAME XPath expression. The column option XPath identifies the values in a row. A NICKNAME option XPath in a child nickname is relative to a NICKNAME option XPath expression in a parent nickname.

When you create a nickname, you choose options that specify the association between the nickname and the XML document. Nicknames that are created for WebSphere Business Integration wrappers are associated with an XML schema definition (XSD) document. Nicknames that are created for Web services wrappers are associated with a Web services description language (WSDL) document.

TEMPLATE option at the nickname and column levels

The WebSphere Business Integration wrapper and the Web services wrapper need the nickname-level and the column-level template fragments, which is the TEMPLATE option on the CREATE NICKNAME statement, at the time that the nickname is created.

The WebSphere Business Integration wrapper and the Web services wrapper build XML documents that are required by the WebSphere Business Integration Adapter and the Web services environment. The wrappers use this information during the query planning and the query execution phases.

The TEMPLATE option for the Web services wrapper

The Web services wrapper needs the TEMPLATE option on the CREATE NICKNAME statement at the time that the nickname is created.
Web services wrapper

For the Web services wrapper, the required and optional attributes vary according to the definitions in the WSDL document and how a column is derived. A column can be derived from either an element or an attribute of an element.

- If the column is derived from an element, then the minOccurs value determines if a column is optional.
- If the value of minOccurs equals 0, then the column is optional.
- If the value of minOccurs equals 1, then the column is required.
- If the column is derived from an attribute of an element, then the value of use on the attribute determines if a column is optional.
- If an attribute contains the value use=optional, then the column is optional.
- If an attribute contains the value use=required, then the column is required.

The following example is an attribute in a schema definition that is associated with a column:

```xml
<xsd:complexType>
  <xsd:sequence>
    <xsd:element ref="tns:ZooName"/>
    <xsd:element ref="tns:Count"/>
    <xsd:element ref="tns:LastModified"/>
  </xsd:sequence>
  <xsd:attribute maxOccurs="unbounded" minOccurs="0" ref="tns:Zookeeper"/>
</xsd:complexType>
```

Creating federated views for Web services nicknames

Create federated views for Web services nicknames to ensure that queries that join pieces of a Web services nickname hierarchy run properly.

You can define federated views for the hierarchy of nicknames that describe a Web services document. A federated view ensures that the queries that join pieces of a Web services nickname hierarchy can run properly.

The Web services wrapper requires joins from any child nickname to include all nicknames back to the parent nickname. A SELECT statement on only a child nickname fails if the parent nickname is not included in the statement. The preferred method of including all required nicknames in a query is to define a federated view.

Procedure

To create federated views for Web services nicknames:

1. Define a view that includes all of the nicknames that are related to an operation in the Web service if you want to join those nicknames.
2. In the WHERE clause of the view, use join predicates for all columns that are related by the PRIMARY_KEY and FOREIGN_KEY column options.

In the following example, the primary key is on column ooport_getzooreport_pk in nickname zooport_getzooreport_report_nn. The foreign key is on column ooport_getzooreport_fkey in nickname zooport_getzooreport_report_report_nn.

```sql
CREATE VIEW zooport
  (zooname, number_of_zookeeper, lastmodified,zookeeper_id, zookeeper_name, cages, animal_name, animal_species, animal_lot)
```
AS ( SELECT zooid, report_zooname, 
    report_count, report_lastmodified, 
    zookeeper_id, zk.report_name, report_numbercages, 
    a.report_name, report_species, report_lot 
FROM zooport_getzooreport_nn , 
    zooport_getzooreport_report_nn as zk, 
    zooport_getzooreport_report_report_nn as a 
WHERE zk.ooport_getzooreport_pkey=a.ooport_getzooreport_fkey 
AND zooport_getzooreport_pkey=zk.ooport_getzooreport_fkey); 

You can get information from all of the nicknames with the following SELECT statement:
SELECT * FROM zooreport WHERE zooid='1';

---

### Query restrictions for wrappers for business applications and Web services

The use of query predicates and joins are restricted for business applications and Web services as shown in examples provided.

#### Equal predicates

The only valid predicates on input columns are equal predicates. For output columns, any predicate is valid.

The following example returns an error with a message that indicates that the predicate is not supported on that column. In this example, the column zipcode is an input column:
SELECT return FROM gettemp WHERE zipcode='95141'

The following example shows a valid query using an equal predicate on the input columns. The customers nickname is joined with a local table that contains customer IDs. The query contains an additional predicate on the Sales column, which is an output-only column.
SELECT a.name, a.address 
FROM customers a, local_table b 
WHERE 
a.customer_id=b.custid AND 
a.Sales > 300000;

#### Predicates for required input columns

You must provide equality predicate values for all required input columns in your SQL queries for the nickname hierarchy that you reference. The wrapper returns an SQLCODE -901 for all queries that violate this restriction.

#### IN or OR predicates

For WebSphere Business Integration wrappers and Web services wrappers, no IN or OR predicates are allowed for input columns.

The following examples show nonvalid queries. The customers nickname has one required input column, customer_id:
SELECT * FROM customers 
WHERE customer_id IN (12345, 67890, 11223); 
SELECT * FROM customers 
WHERE customer_id IN (SELECT custid FROM local_table); 

---

Chapter 18. Configuring access to Web services data sources
However, for the WebSphere Business Integration wrappers, you can use IN list predicates with required input columns if you define a unique index with the SPECIFICATION ONLY parameter for the required input columns:

```
CREATE UNIQUE INDEX myindex ON customers(customer_id) SPECIFICATION ONLY;
```

### Joins on optional input columns

The following examples demonstrate a restriction on joining optional input columns and on using host variables. You cannot join optional input columns from a local table or nickname. If the WSDL generates an input nickname column as optional and you need to use that column in a join, then you must edit the DDL statement to change the column to a required input column.

In this example, a Web service wrapper nickname named order is created with shipping_method as an optional input column. The following statement is a valid query because it uses a literal in the predicate:

```
SELECT * FROM order
  WHERE part="hammer" AND shipping_method="FEDEX";
```

However, if you include a local table named orderparts, which defines parts and shipping methods, in the query, and the table contains a column named shipping_method that is optional, the statement is nonvalid:

```
SELECT * FROM
  order o, orderparts op
WHERE
  o.part="hammer" AND
  o.shipping_method=op.shipping_method
```

If you use host variables in queries with predicates on optional input columns, the statement is also invalid:

```
SELECT * FROM
  order o
WHERE
  o.part="hammer" AND
  o.shipping_method=:ashipping_method
```

For a WebSphere Business Integration wrapper, predicates on optional input columns of a nickname might be pushed down to the WebSphere Business Integration Adapter. The federated server can decide to apply those predicates locally on the rows that are fetched from the application data source. To ensure that predicates for a given input column always get pushed down to the adapter, declare the input column as a required input column. Every query on the nickname hierarchy must include predicate values for the required input columns.

To ensure valid results, joined input columns must be required columns for Web Services wrappers.

### Outer joins

Outer joins between nicknames using the primary key from a parent nickname and the foreign key from child nickname columns are not supported.

When a parent element in an XML document contains no child elements, and if you use an inner join between the parent nickname and the child nickname, then no rows are returned for that element. For example, for a given customer, if there
is no bankdetail information in the SAP system, then no rows are returned for the sap_bapi_customer_getdetail2_sap_customerbankdetail_NN nickname for the particular customer.

The following CREATE NICKNAME statements define the columns that are used in the example query:

```
CREATE NICKNAME sap_bapi_customer_getdetail2_NN(
  ...
  NAME VARCHAR(35)
    OPTIONS(XPATH './ns3:sap_customeraddress/
      ns1:sap_customeraddress/ns1:NAME/text()'),
  ...
  NN__PKEY VARCHAR(16) OPTIONS(PRIMARY_KEY 'YES'),
  COMPANYCODE VARCHAR(4) OPTIONS(XPATH './ns3:COMPANYCODE/text()',
    TEMPLATE '<ns3:COMPANYCODE>&column</ns3:COMPANYCODE>'),
  CUSTOMERNO VARCHAR(10) OPTIONS(XPATH './ns3:CUSTOMERNO/text()',
    TEMPLATE '<ns3:CUSTOMERNO>&column</ns3:CUSTOMERNO>'),
  ...
  FOR SERVER sap_server
OPTIONS(XPATH '/ns3:sap_bapi_customer_getdetail2',
  TEMPLATE '<ns3:sap_bapi_customer_getdetail2>
    &sap_bapi_customer_getdetail2_sap_customerbankdetail_NN[0,1]
    &COMPANYCODE[0,1]
    &CUSTOMERNO[1,1]
    </ns3:sap_bapi_customer_getdetail2>',
  ...
)
```

*Figure 25. Excerpt from getdetail2 nickname*

```
CREATE NICKNAME sap_bapi_customer_getdetail2_sap_customerbankdetail_NN(
  CUSTOMER VARCHAR(10) OPTIONS(XPATH './ns2:CUSTOMER/text()',
    TEMPLATE '<ns2:CUSTOMER>&column</ns2:CUSTOMER>'),
  BANK_KEY VARCHAR(15) OPTIONS(XPATH './ns2:BANK_KEY/text()',
    TEMPLATE '<ns2:BANK_KEY>&column</ns2:BANK_KEY>'),
  BANK_ACCT VARCHAR(18) OPTIONS(XPATH './ns2:BANK_ACCT/text()',
    TEMPLATE '<ns2:BANK_ACCT>&column</ns2:BANK_ACCT>'),
  CTRL_KEY VARCHAR(2) OPTIONS(XPATH './ns2:CTRL_KEY/text()',
    TEMPLATE '<ns2:CTRL_KEY>&column</ns2:CTRL_KEY>'),
  BANK_REF VARCHAR(20) OPTIONS(XPATH './ns2:BANK_REF/text()',
    TEMPLATE '<ns2:BANK_REF>&column</ns2:BANK_REF>'),
  NN__FKEY VARCHAR(16) OPTIONS(FOREIGN_KEY 'SAP_BAPI_CUSTOMER_GETDETAIL2_NN'))
FOR SERVER sap_server
OPTIONS(XPATH '/ns3:sap_customerbankdetail/ns2:sap_customerbankdetail',
  TEMPLATE '<ns2:sap_customerbankdetail>
    &CUSTOMER[0,1]
    &BANK_KEY[0,1]
    &BANK_ACCT[0,1]
    &CTRL_KEY[0,1]
    &BANK_REF[0,1]
    </ns2:sap_customerbankdetail>',
  ...
)
```

*Figure 26. Excerpt from customer bank detail nickname*

In the following example, the query returns no rows because there is an inner join condition between the two nicknames:
SELECT a.name, b.bank_key
FROM sap_bapi_customer_getdetail2_NN a,
sap_bapi_customer_getdetail2_sap_customerbankdetail_NN b
WHERE a.customerno='1234567890'
AND a.NN__PKEY=b.NN__FKEY;

If a WebSphere Business Integration wrapper or a Web services wrapper nickname definition contains required input columns, then a left outer join between this nickname and any other local table or other nicknames is not supported.

---

**Web services data sources – example queries**

Examples of SQL queries for the Web services wrapper are shown.

**Example 1: Using materialized query tables**

You use materialized query tables to locally cache the results of a query and to improve the performance of queries. You can use nicknames from Web services data sources to create materialized query tables. For some queries, the database can automatically determine whether the materialized query table can answer a query without accessing the base tables. The following procedure shows how to create and populate a materialized query table:

1. Create a local or base table:
   ```sql
   CREATE TABLE mystocks(ticker VARCHAR(10));
   ```
   You can use the local table to maintain all the values that you want to cache.

2. Insert all of the values that you want to cache into the table:
   ```sql
   INSERT INTO mystocks VALUES('IBM');
   INSERT INTO mystocks VALUES('MSFT');
   ...
   ```

3. Create a Web services nickname:
   ```sql
   CREATE NICKNAME stockquote_nn (
     ticker VARCHAR(40) OPTIONS (TEMPLATE '&column'),
     price VARCHAR(16) OPTIONS (XPATH './Result/text()')
   )
   FOR SERVER stock_server
   OPTIONS (TEMPLATE '<ticker>&column</ticker>'
    XPATH './Result/text()');
   ```

4. Create a view that consists of the nickname and the local table:
   ```sql
   CREATE VIEW stock_quote_view (ticker, price)
   AS (
     SELECT nn.ticker, nn.price
     FROM stockquote_nn nn, mystocks s
     WHERE nn.ticker=s.ticker
   );
   ```

5. Create a materialized query table:
   ```sql
   CREATE TABLE stockquote_MQT (ticker, ticker2, price)
   as (SELECT nn.ticker, s.ticker as ticker2, nn.price
     FROM stockquote_nn nn, mystocks s
     WHERE nn.ticker=s.ticker)
   DATA INITIALLY DEFERRED REFRESH DEFERRED;
   ```
   Include all of the VARCHAR columns in the join predicate (nn.ticker and s.ticker) in the materialized query table output list to maximize the opportunities that the materialized query table is used by the federated database.
To defer the refresh of the materialized query table, specify the REFRESH DEFERRED keyword. Materialized query tables that are specified with the REFRESH DEFERRED keyword do not reflect changes to the underlying base table. Use the clause DATA INITIALLY DEFERRED so that your data is not inserted into the table as part of the CREATE TABLE statement.

6. Issue a REFRESH TABLE statement to populate the table. The data in the table reflects the result of the query as a snapshot at the time that you issue the REFRESH TABLE statement. The following example populates the stockquote_MQT table and sets a value for the special register with the current refresh age.

   REFRESH TABLE stockquote_MQT;
   SET CURRENT REFRESH AGE any;

   The queries that run on the data in the materialized query table are faster than the queries that run on the data in a base table. When you want to use the materialized query table, you refer to the view and not the nickname:

   SELECT * FROM stock_quote_view
   WHERE ticker='IBM';

   If you issue a query to select a value that has not been cached, 0 rows are returned.

**Example 2: Issuing joins using the primary and foreign keys**

The PRIMARY_KEY and FOREIGN_KEY columns define relationships between the parent and child nicknames. Each parent nickname must have a primary key column option. You define the children of a parent nickname with the foreign key column option that references the primary key column of a parent nickname. A nickname can have multiple children, but a nickname can have only one parent.

Because these columns contain only binary data, the columns are defined with the FOR BIT DATA NOT NULL keywords. The DB2 Control Center generates this definition when you create the nickname. You can explicitly define the PRIMARY_KEY and FOREIGN_KEY columns as FOR BIT DATA NOT NULL when you create the nickname.

The following example shows how the Web services wrapper uses the PRIMARY_KEY and FOREIGN_KEY columns to associate parent and child nicknames.

   CREATE NICKNAME zooprot_getzooreport_nn (
   zooId VARCHAR (48) OPTIONS(TEMPLATE '{@column'},
   zoo_id VARCHAR (48) OPTIONS(XPATH './ns1:Zoo/@id'),
   report_zoo_zooname VARCHAR (48)
   OPTIONS(XPATH './ns1:Zoo/ns1:ZooName/text()'),
   report_zoo_count VARCHAR (48)
   OPTIONS(XPATH './ns1:Zoo/ns1:Count/text()'),
   report_zoo_lastmodified VARCHAR (48)
   OPTIONS(XPATH './ns1:Zoo/ns1:LastModified/text()'),
   nn_pk VARCHAR (16) NOT NULL OPTIONS(PRIMARY_KEY 'YES'),
   url VARCHAR (256) OPTIONS(URLCOLUMN 'Y'),
   soapaction VARCHAR (256) OPTIONS(SOAPACTIONCOLUMN 'Y')
   ) FOR SERVER "mytestsrvr"
   OPTIONS(
   URL 'http://localhost:9080/MaelstromTest/services/ZooPort',
   SOAPACTION 'http://myzoo.com/getZooReport',
   TEMPLATE '<soapenv:Envelope>
<soapenv:Body>
<zooId>&zooId[1,1]/zooId>'

   Chapter 18. Configuring access to Web services data sources  329
The foreign key, nn_fk, in nickname zooport_getzooreport_report_zookeeper_nn, refers to the parent nickname, zooport_getzooreport_nn in the foreign key option. The designated primary and foreign key nickname columns do not correspond to data in your WSDL document because these nickname columns contain keys that are generated by the wrapper. These keys identify a relationship between the parent and child nicknames that is unique only within a query. If the child nickname contains an input column, the parent nickname option template refers to that child nickname in the template structure with the nickname option.

The following SQL statement joins the parent and child nicknames:

```sql
SELECT *
FROM zooport_getzooreport_nn a,
     zooport_getzooreport_report_zookeeper_nn z,
WHERE a.nn_pk = z.nn_fk
AND a.zooid = 100
;```

The following description explains how the Web services wrapper uses the TEMPLATE and XPATH nickname and column options during query execution. It is not intended as an example of a specific implementation.

When you join the primary and foreign key columns, the Web services wrapper sends a message to the Web services provider, and a set of rows is returned from the Web services provider. The wrapper generates a message for the parent nickname by substituting the values of the input column (a.zooid = 100) from the query for the reference in the column option template (ZOOID VARCHAR (48) OPTIONS(TEMPLATE 'columname'), and then all of the column references in the nickname template option (<zooid>&zoooid[1,1]/zooi</zooi>). The nickname template option can include column references or child nickname references. The message is then sent to the Web service.

The wrapper generates the rows for a nickname by applying the nickname option XPATH on the document that the Web service returns. If the nickname option XPATH returns multiple XML fragments, then the nickname contains multiple rows. The column XPATH option is applied on the resulting XML fragments that represent the rows to get the column values. If a nickname has one or more indirect parents, all of the parent nickname XPATH expressions are applied in the order from the top of the hierarchy down before the nickname option XPATH and the column option XPATH are applied for this nickname.
Chapter 19. Configuring access to business application data sources

You can integrate the data that is in the business application databases with information from other sources by using a federated system. To configure the federated server, you must register a wrapper, a server definition, and nicknames for the business application databases.

Before you begin

The federated server must be installed and a federated database that uses a 32-bit DB2 database instance must exist on the federated server.

Procedure

To configure access to business application data sources, complete these tasks:

WebSphere Business Integration wrapper

The WebSphere Business Integration wrapper provides an SQL interface to business applications, such as those produced by SAP, Siebel, and PeopleSoft.

By using the WebSphere Business Integration wrapper, you can use the federated systems functions to join business data from business applications with data on other federated data sources. The WebSphere Business Integration wrapper extracts business object definitions into a hierarchy of nicknames. The wrapper supports result sets that return multiple business objects.

The WebSphere Business Integration wrapper is a read-only wrapper and uses the WebSphere Business Integration Adapters to access business applications. To return result sets that include multiple business objects, the WebSphere Business Integration wrapper requires the WebSphere Business Integration Adapters and Framework, version 2.6.

The following figure shows the relationship between the WebSphere Business Integration wrapper and the adapters in the DB2 database environment, and to other federated data sources.
The following steps describe the process for accessing business application data in a federated system:

1. A user sends a query to the federated server to access a nickname that maps to a data source such as a Siebel application.
2. The wrapper transforms the query into a business object.
3. The wrapper places the business object on a WebSphere MQ message request queue.
4. The WebSphere Business Integration adapter for the particular application reads the business object, which is the request, from the message request queue.
5. The WebSphere Business Integration adapter works with the business application to prepare a response business object.
6. The WebSphere Business Integration adapter puts the response business object on the message response queue.
7. The wrapper reads the response business object from the response queue.
8. The wrapper extracts the response business object into a result set based on the relational schema that is defined with the nickname definition.

Figure 27. **WebSphere Business Integration wrapper in the DB2® database environment**
Configuring the WebSphere Business Integration Adapters

For each business application that you want to access with SQL statements by using the federated wrapper functions, you must install and configure a WebSphere Business Integration Adapter. Each adapter maps to a federated server definition.

Before you begin

Supported adapters are as follows:
- WebSphere Business Integration Adapter for SAP Exchange Infrastructure
- WebSphere Business Integration Adapter for Siebel Business Applications
- WebSphere Business Integration Adapter for PeopleSoft Enterprise

Procedure

To configure a business integration adapter:
1. Configure the Object Discovery Agent tool and the Business Object Designer tool, and build the business object definitions.
2. Define a configuration file by using the Connector Configurator tool from the WebSphere Business Integration Adapter interface. The configuration file must contain the following information:
   a. The business objects that the adapter supports.
   b. The configuration properties for the adapter.
3. Define the WebSphere MQ message queues that the wrapper requires: request_queue, response_queue, and fault_queue.
4. Define the additional message queues that are required by the adapter:
5. Define the WebSphere MQ user authorization by using either of the following methods:
   • Define the DB2 instance owner ID as part of the MQManager group.
   • Ensure that the MQManager administrator sets the MCAUSER value while creating the ServerConnection channel. The value of MCAUSER must be a user ID that is part of the MQManager group or the Administrator group.

The WebSphere Business Integration Adapters, version 2.6, support result sets for business applications. The adapter configuration file provides two properties to enable result sets for business applications:

ResultsSetEnabled
   Set this property to true to enable result sets. The default value is false.

ResultsSetSize
   The WebSphere Business Integration wrapper sets the maximum size of the result set at run time as a property in the JMS message header. Do not change this value. The default value is 0.

Configure these properties in the adapter configuration file before you start the adapter.

Business object definitions

A business object definition is a template from which the business integration adapter creates an instance of a business object. A business object definition represents a business application data entity.
A business object is an instance of a business object definition and can be either a flat or hierarchical structure. A flat business object contains only simple attributes. A hierarchical business object contains one or more complex attributes. A repository of business object definitions exists within each business integration adapter for the particular application that is supported.

The following example shows a flat business object:

**Customer**

- Gomez
- Juanita
- Apt 2C
- 123 Main Street
- Big City
- California
- 91234
- 888
- 1111111

The following example shows a hierarchical business object with some complex attributes:

**Contact** (Parent)

- ID
- Customer ID
- Date
- Text
- Authorization

  Line items (there are 0 or more line item elements)

  (Child elements)

  - Business object 1
  - Business object 2
  - Business object 3

Business object definitions must be generated by using the Object Discovery Agent tool that is packaged with each business integration adapter. The Object Discovery Agent tool generates an XML schema definition file for a business object definition. The Object Discovery Agent tool might generate multiple schema files if the business object that is being defined has a hierarchical structure.

The XML schema definition is a file with file type XSD in a directory that is specified in the business integration configuration. You must generate the business object definition before you create the nicknames for the business integration wrapper.

To create nicknames, use the XSD file that the Object Discovery Agent tool creates. Nicknames provide a relational schema representation of the business object definition. The business integration wrapper maps a hierarchical business object into a hierarchy of relational nicknames. For example, each child business object of cardinality 'n' is mapped to a separate nickname that is linked to the nickname of the parent business object with a foreign key constraint.

The business objects that are accessible to the federated server map to the specific application entities in the following table:

<table>
<thead>
<tr>
<th>Business objects</th>
<th>Application entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siebel</td>
<td>Business Component</td>
</tr>
<tr>
<td>PeopleSoft</td>
<td>Component Interface</td>
</tr>
</tbody>
</table>
Table 85. Business objects and the related application entities (continued)

<table>
<thead>
<tr>
<th>Business objects</th>
<th>Application entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP</td>
<td>BAPI</td>
</tr>
</tbody>
</table>

**Configuration properties for business object definitions**

There are standard configuration properties and application-specific configuration properties for business object definitions.

The following configuration properties must be in the configuration file when you define business object definitions:

**Standard configuration properties**

You must customize some property values to use the adapter with a federated system. Some specific properties to configure are included in the following list:

- Specify the value of the integration broker as WMQI.
- Specify the location of the metadata repository that is owned by the adapter. The XML schema definition files, which contain the business object definitions, are saved in this location.
- Specify the type of delivery transport as WMQI-MQ.
- Specify the name of the queue manager that manages the queues that are used by the adapter.
- Specify the names of the queues that are required to run the adapter.

**Application-specific configuration properties**

These properties specify values for a particular application-specific component. The values that you provide help to establish a session with the application. The properties also direct the processing behavior for the application-specific components.

**WebSphere MQ message queues for the WebSphere Business Integration wrapper and the adapter**

WebSphere MQ is the messaging and transport layer between the adapter and the wrappers. You must define the queues that are required for the WebSphere Business Integration wrapper and adapter.

You must define eight queues to use the WebSphere Business Integration wrapper. The wrapper uses three of these static queues to exchange messages, including data objects and error messages, between the adapter and the wrapper:

- **request_queue**
  Delivers request messages from the federated server to the adapter.
- **response_queue**
  Delivers response messages from the adapter to the federated server.
- **fault_queue**
  Delivers fault messages from the adapter to the federated server. The adapter places a message on this queue when it is unable to place the message on the reply-to queue.

The following diagram shows the flow of messages on message queues between Siebel applications and the federated server.
The WebSphere Business Integration adapters require five additional queues that are used when the adapter is used with a WebSphere MQ Integrator broker. You must create and configure these additional message queues so that the adapter can be started:

- AdminInQueue
- AdminOutQueue
- SynchronousRequestQueue
- SynchronousResponseQueue
- DeliveryQueue

If result sets are enabled for the WebSphere Business Integration wrapper, WebSphere MQ log files can be very large. Configure more log files or larger log files if necessary.

**Registering the WebSphere Business Integration wrapper (DB2 command line)**

You must register a wrapper to access WebSphere Business Integration data sources. Wrappers are used by federated servers to communicate with and retrieve data from data sources.

**Before you begin**

1. Install and configure the appropriate adapter.
2. Install and configure WebSphere MQ, version 5.3 (CSD level 5) or later.
3. Create the WebSphere MQ message queues.
4. If the WebSphere MQ manager is not installed on the same system as the federated server, install the WebSphere MQ, version 5.3 (CSD level 5) client on the same system on which you installed a federated server instance.

**About this task**

You can register the wrapper by using the DB2 Control Center or the DB2 command line. The DB2 Control Center includes a wizard to guide you through the steps that are required to register the wrapper.

**Procedure**
To register a wrapper:

Issue the CREATE WRAPPER statement with the name of the wrapper and the name of the wrapper library file. For example, to register a wrapper with the name wbi_adapter on the federated server that uses Windows, issue the following statement:

```
CREATE WRAPPER wbi_adapter LIBRARY 'db2wbi.dll';
```

The name of the wrapper library file that you specify depends on the operating system of the federated server.

**Business integration wrapper library files**

The business integration wrapper library files are added to the federated server when you install the federated server.

When you install the federated server, library files are added to the directory path listed in the table. For example, if the federated server is running on AIX, the wrapper library files added to the directory path are libdb2wbi.a, libdb2wbiF.a, and libdb2wbiU.a.

You must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name. When you register a wrapper, specify only the library file name that is listed in the table.

**Table 86. Business integration wrapper library locations and file names**

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Default directory path</th>
<th>Default wrapper library file</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/install_path/lib/</td>
<td>libdb2wbi.a</td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/IBM/db2/install_path/lib</td>
<td>libdb2wbi.so</td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/IBM/db2/install_path/lib</td>
<td>libdb2wbi.so</td>
</tr>
<tr>
<td>Windows</td>
<td>%DB2PATH%\bin</td>
<td>db2wbi.dll</td>
</tr>
</tbody>
</table>

- **install_path** is the directory path where the federated server is installed on UNIX or Linux.
- %DB2PATH% is the is the environment variable that is used to specify the directory path where the federated server is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

**Registering the server definition for business application data sources (DB2 command line)**

You can register a server definition from the DB2 command line.

**Restrictions**

The server options must include the WebSphere MQ Series queue definitions:

- Specify the MQ_SVRCONN_CHANNELNAME option only if you specify the MQ_CONN_NAME option.
- Do not drop the MQ_CONN_NAME option until you drop the MQ_SVRCONN_CHANNELNAME option.
- If MQ_CONN_NAME is not specified, the federated system uses the value of the MQSERVER environment variable.
The WebSphere Business Integration wrapper requires the WebSphere Business Integration Adapters and Framework, version 2.6 for result set support. The wrapper supports the value 2.6 for the VERSION parameter that is specified in the CREATE SERVER statement. To enable result sets, you must specify version 2.6 for the VERSION parameter in the CREATE SERVER statement.

About this task

After you register the wrapper, you must register a corresponding server.

A server option enables result sets. Set the RESULTSET_ENABLED option to Yes to enable result sets. Set the RESULTSET_ENABLED option to No to disable result sets. The default value for this server option is No. The VERSION parameter value must be set to 2.6 or later to enable result sets. Before you set the RESULTSET_ENABLED option to Yes, ensure that the ResultSetEnabled property in the adapter configuration file is set to true. If the values do not match, an error results.

Procedure

To register the server definition for a business application:

Issue the CREATE SERVER statement. For example, to register a server definition for the Siebel business applications:

```sql
CREATE SERVER siebel_server
VERSION 2.6
WRAPPER wbi_wrapper
OPTIONS ( App_Type 'siebel',
          Request_Queue 'myqueue3',
          Response_Queue 'myqueue4',
          Fault_Queue 'myqueue5',
          MQ_Manager 'mymq',
          MQ_REPONSE_TIMEOUT '55000',
          MQ_CONN_NAME '9.30.76.151(1420)',
          MQ_SRVCONN_CHANNELNAME 'SYSTEM.DEF.SVRCONN',
          RESULTSET_ENABLED 'Y'
    )
```

In the example, the business application is a Siebel application, which is identified with the APP_TYPE option. The valid values are SIEBEL, PSOFT, and SAP. The VERSION option represents the version of the business integration adapters that you are using. Valid values are 2.3, 2.4, and 2.6, but you must specify version 2.6 to enable result sets. The default value for MQ_RESPONSE_TIMEOUT is set to 50000 milliseconds. A value of -1 specifies that there is no timeout limit.

Registering nicknames for business application data sources

You can register the nickname by using the DB2 Control Center or the DB2 command line. The DB2 Control Center includes a wizard to guide you through the steps that are required to register the nickname.

You must create nicknames that correspond to the structural hierarchy of your business object definition. Parent nicknames contain at least one child nickname. Child nicknames correspond to the elements that contain a cardinality greater than 1 that are nested within the element for the parent nickname. The DB2 Control Center generates unique nicknames by grouping part names or element names with the column name from the XML schema document.
Procedure

To register nicknames for business application data sources:

Use one of the following methods:

Registering nicknames for business application data sources (DB2 Control Center)

For each business integration server definition that you register, you must register a nickname for each data source that you want to access. Use these nicknames instead of the names of the data sources when you query the Web services data sources.

Before you begin

If result sets are not enabled, identify the required input columns before you generate nickname definitions by using the DB2 Control Center.

Procedure

To register nicknames for business application data sources from the DB2 control center:

1. Expand the Federated Database Objects folder.
2. Expand the wrapper folder for which you want to register nicknames.
3. Expand the Server Definitions folder.
4. Expand the server folder for which you want to register nicknames.
5. Right-click the Nicknames folder and select Create.
6. In the Create Nicknames window, click Discover to define search criteria to help you select objects from the data source.
7. Specify the XML schema definition file that contains the definition of the business objects that you want federated system users to access.
8. Click OK to create the nickname according to the selected XML schema definition file.

The DB2 Control Center extracts the schema file into multiple CREATE NICKNAME statements, with the appropriate parent-child relationship definitions. The nicknames that are created represent the business object hierarchy that is defined in the XML schema definition file.

Identifying required input columns of a business object

If you want to retrieve only one business object from a query, you need to ensure that the wrapper sends all the required predicates. If result sets are enabled, you can retrieve result sets of multiple business objects.

Before you begin

If you are using a WebSphere Business Integration Adapter, version 2.4 or earlier, you must identify the required input columns in the business object.

If you are using a WebSphere Business Integration Adapter, version 2.6 or later and result sets are not enabled, you can identify the required input columns.
Because the business integration wrapper supports result sets of multiple business objects, you do not need to identify the required input columns in the business objects for PeopleSoft and Siebel applications. However, the business integration adapters support only equality predicates in all queries on input columns. You can deselect any column that should not be an input column so that you can query by using a nonequality predicate.

Optionally, you can identify required input columns to limit the response to a query to only a single business object.

Procedure

To identify the required input columns in the business object:

1. Identify the columns in the SAP, Siebel, or PeopleSoft application repository that represent a unique key for the application entity being mapped.

<table>
<thead>
<tr>
<th>Application</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SAP</strong></td>
<td>Use the SAP Business Object Repository to identify the required input parameters for the BAPI that is being mapped to a WebSphere Business Integration business object definition by the WebSphere Business Integration Object Discovery Agent tool.</td>
</tr>
<tr>
<td><strong>Siebel</strong></td>
<td>Use one of the following methods:</td>
</tr>
<tr>
<td></td>
<td>- The Siebel application has a unique identifier column that is associated with each business component and generates hexadecimal values for this column for each instance of the business component. This identifier column exists at the highest level of the business component hierarchy and is already flagged by the isKey=&quot;true&quot; specification in the appSpecificInfo section of the XML annotation for the element in the generated XSD file. If result sets are enabled, every column in the root level can be marked as an input column.</td>
</tr>
<tr>
<td></td>
<td>- You can use Siebel tools to identify the database columns that represent a composite unique key for the business component that is being mapped. These columns must all be at the highest or root level of the business object hierarchy.</td>
</tr>
<tr>
<td><strong>PeopleSoft</strong></td>
<td>Use the Application Designer tool to identify the getKey columns in the component interface for the highest level of the hierarchy that is being mapped to a WebSphere Business Integration business object definition. If result sets are enabled, findKey columns can be used as input columns.</td>
</tr>
</tbody>
</table>

2. Edit the XSD files that are generated for the business object definition by the WebSphere Business Integration Object Discovery Agent tool to flag the required input columns.

3. Generate the nickname DDL statement for the business object definition from the DB2 Control Center.

**Registering nicknames for business application data sources (DB2 command line)**

For each business integration server definition that you register, you must register a nickname for each data source that you want to access. Use these nicknames instead of the names of the data sources when you query the Web services data sources.
Procedure

To register nicknames for business application data sources from the DB2 command line, issue a CREATE NICKNAME statement:

For example, to register a nickname for a Siebel business object definition that is called sieb_ssa_Contact_Contract, issue the following statement:

```sql
CREATE NICKNAME sieb_ssa_Contact_Contract_NN(
  Id VARCHAR(15) OPTIONS(XPATH './ns1:Id/text()',
    TEMPLATE '<ns1:Id>&column</ns1:Id>'),
 FirstName VARCHAR(50) OPTIONS(XPATH './ns1:FirstName/text()',
    TEMPLATE '<ns1:FirstName>&column</ns1:FirstName>'),
 LastName VARCHAR(50) OPTIONS(XPATH './ns1:LastName/text()',
    TEMPLATE '<ns1:LastName>&column</ns1:LastName>'),
 AccountId VARCHAR(255) OPTIONS(XPATH './ns1:AccountId/text()'),
 PrimaryAccountName VARCHAR(100) OPTIONS(XPATH './ns1:PrimaryAccountName/text()'),
 PrimaryPostalCode VARCHAR(30) OPTIONS(XPATH './ns1:PrimaryPostalCode/text()'),
 PrimaryStreetAddress VARCHAR(200) OPTIONS(XPATH './ns1:PrimaryStreetAddress/text()'),
 SalesRep VARCHAR(255) OPTIONS(XPATH './ns1:SalesRep/text()'),
 State VARCHAR(255) OPTIONS(XPATH './ns1:State/text()')
) FOR SERVER siebel_server
  OPTIONS(XPATH '//ns1:sieb_ssa_Contact_Contract',
    TEMPLATE '<ns1:sieb_ssa_Contact_Contract>
      &Id[1,1] &FirstName[0,1] &LastName[0,1] &AccountId[0,1] &PrimaryAccountName[0,1] &PrimaryPostalCode[0,1] &PrimaryStreetAddress[0,1] &SalesRep[0,1] &State[0,1]'
  )
BUSOBJ_NAME 'sieb_ssa_Contact_Contract',
  sieb_ssa_Contact_Contract"'
REQUIRE_PREDICATE 'N',
RESULTSET_SIZE '50');
```

The BUSOBJ_NAME nickname option is the name of the XML schema definition (XSD) file that represents the business object definition.

CREATE NICKNAME statement – Examples for the WebSphere Business Integration wrapper

The examples in this section show how to use the CREATE NICKNAME statement.

Example 1: Flat business object

Figure 1 shows a portion of an XSD file that represents a WebSphere Business Integration business object definition for a Siebel Business Component. The business object definition hierarchy consists of a single level, which contains only the root business object. The DB2 Control Center creates a single relational nickname to represent this business object definition.

In the XSD file, the ID element is flagged as a required input column by adding the isRequired="true" flag in the annotation section for the element. The FirstName and LastName columns are designated as optional input columns by adding the isRequired="false" flag.

---

Figure 29. XML schema file for a flat business object
The DB2 Control Center generates the following CREATE NICKNAME statement from the XSD file that is shown in Figure 1. A TEMPLATE option value is specified for each input column. The column option templates are associated with the nickname option template. The input columns are also referenced in the option value of the nickname level TEMPLATE. The nickname option template provides the structure for the input business object. The value of the minOccurs attribute for each of the input column references in the nickname template value determines whether the input column is a required or optional column. The reference for the ID column is specified as &Id[1,1]. The references for the FirstName and LastName columns are specified as &FirstName [0,1] and &LastName [0,1]. All output columns include an XPATH column option value. The nickname is for a flat business object that does not contain any children (elements of cardinality ‘n’):

```
CREATE NICKNAME sieb_ssa_Contract_Contract_NN(  
  Id VARCHAR(15) OPTIONS(XPATH './ns1:Id/text()',  
    TEMPLATE '<ns1:Id>&column</ns1:Id>'),  
  FirstName VARCHAR(50) OPTIONS(XPATH './ns1:FirstName/text()'),
```

Chapter 19. Configuring access to business application data sources  343
Example 2: Hierarchical business object

Figure 2 shows a business object definition that consists of a two-level hierarchy comprising the root business object and two child business objects, or three XSD files. Only two nicknames are generated to represent the business object definition hierarchy. The sap_customeraddress child business object has a cardinality of 1, which is indicated by the absence of a maxOccurs attribute specification in the element definition. All of the columns of sap_customeraddress are included in the root nickname, sap_bapi_customer_getdetail2_NN. The sap_customerbankdetail child business object has a cardinality of n, which is indicated by the maxOccurs="unbounded" specification in the element definition. The sap_customerbankdetail child business object is mapped to a separate child nickname, sap_bapi_customer_getdetail2_sap_customerbankdetail_NN. The child nickname is associated with the root nickname by a special primary key-foreign key relationship.

Figure 30. SAP hierarchical business object: customer_getdetail2

```xml
<xsd:element name="sap_bapi_customer_getdetail2">
  <xsd:annotation>
    <xsd:appinfo>
      <bx:boDefinition version="3.0.0" />
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>

<xsd:element name="COMPANYCODE" minOccurs="0">
  <xsd:annotation>
    <xsd:appinfo>
      <bx:boAttribute>
        <bx:appSpecificInfo>ICOMPANYCODE</bx:appSpecificInfo>
        <bx:attributeInfo isForeignKey="false" isKey="true" />
      </bx:boAttribute>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>
```
Chapter 19. Configuring access to business application data sources
Figure 31. SAP hierarchical business object: customer_address

```xml
<xsd:element name="sap_customeraddress">
  <xsd:annotation>
    <xsd:appinfo>
      <bx:boDefinition version="3.0.0">
        <bx:appSpecificInfo>:CUSTOMERADDRESS</bx:appSpecificInfo>
      </bx:boDefinition>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="CUSTOMER" minOccurs="0">
        <xsd:annotation>
          <xsd:appinfo>
            <bx:boAttribute>
              <bx:appSpecificInfo>:CUSTOMER</bx:appSpecificInfo>
              <bx:attributeInfo isForeignKey="false" isKey="true" />
            </bx:boAttribute>
          </xsd:appinfo>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="NAME" minOccurs="0">
        <xsd:annotation>
          <xsd:appinfo>
            <bx:boAttribute>
              <bx:appSpecificInfo>:NAME</bx:appSpecificInfo>
              <bx:attributeInfo isForeignKey="false" isKey="false" />
            </bx:boAttribute>
          </xsd:appinfo>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="CITY" minOccurs="0">
        <xsd:annotation>
          <xsd:appinfo>
            <bx:boAttribute>
              <bx:appSpecificInfo>:CITY</bx:appSpecificInfo>
              <bx:attributeInfo isForeignKey="false" isKey="false" />
            </bx:boAttribute>
          </xsd:appinfo>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="POSTL_CODE" minOccurs="0">
        <xsd:annotation>
          <xsd:appinfo>
            <bx:boAttribute>
              <bx:appSpecificInfo>:POSTL_CODE</bx:appSpecificInfo>
              <bx:attributeInfo isForeignKey="false" isKey="false" />
            </bx:boAttribute>
          </xsd:appinfo>
        </xsd:annotation>
      </xsd:element>
      ...
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure 31. SAP hierarchical business object: customer_address

346  Configuration Guide for Federated Data Sources
Figure 32. SAP hierarchical business object: bank_detail
The DB2 Control Center generates two nicknames from the three SAP XSD files: sap_bapi_customer_getdetail2_NN, which is shown in figure 5, and sap_bapi_customer_getdetail2_sap_customerbankdetail_NN, which is shown in figure 6. Column customerno is marked as a required input column in the nickname level template of the sap_bapi_customer_getdetail2_NN nickname because of the XSD file specifications for the customerno element. Customerno is flagged with an "I" prefix and a minOccurs=1 attribute value.

Figure 33. getdetail2 nickname

CREATE NICKNAME sap_bapi_customer_getdetail2_NN(
    CUSTOMER VARCHAR(10)
    OPTIONS(XPATH './ns3:sap_customeraddress/ns1:sap_customeraddress/ns1:CUSTOMER/text()'),
    NAME VARCHAR(35)
    OPTIONS(XPATH './ns3:sap_customeraddress/ns1:sap_customeraddress/ns1:NAME/text()'),
    CITY VARCHAR(35)
    OPTIONS(XPATH './ns3:sap_customeraddress/ns1:sap_customeraddress/ns1:CITY/text()'),
    POSTL_CODE VARCHAR(10)
    OPTIONS(XPATH './ns3:sap_customeraddress/ns1:sap_customeraddress/ns1:POSTL_CODE/text()'),
    STREET VARCHAR(35)
    OPTIONS(XPATH './ns3:sap_customeraddress/ns1:sap_customeraddress/ns1:STREET/text()'),
    REGION VARCHAR(3))
OPTIONS(XPATH './ns3:sap_customeraddress/
   ns1:sap_customeraddress/ns1:REGION/text()'),
NN__PKEY VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(PRIMARY_KEY 'YES'),
COMPANYCODE VARCHAR(4) OPTIONS(XPATH './ns3:COMPANYCODE/text()',
   TEMPLATE '<ns3:COMPANYCODE>&column</ns3:COMPANYCODE>'),
CUSTOMERNO VARCHAR(10) OPTIONS(XPATH './ns3:CUSTOMERNO/text()',
   TEMPLATE '<ns3:CUSTOMERNO>&column</ns3:CUSTOMERNO>'),
ObjectEventId VARCHAR(48) OPTIONS(XPATH './ns3:ObjectEventId/text()')
FOR SERVER sap_server
OPTIONS(XPATH './ns3:sap_bapi_customer_getdetail2',
   TEMPLATE '<ns3:sap_bapi_customer_getdetail2>
      &sap_bapi_customer_getdetail2_sap_customerbankdetail_NN[0,1]
      &COMPANYCODE[0,1]
      &CUSTOMERNO[0,1]
   </ns3:sap_bapi_customer_getdetail2>',
BUSOBJ_NAME 'sap_bapi_customer_getdetail2',
NAMESPACES 'ns1="http://www.ibm.com/websphere/
crossworlds/2002/BOSchema/sap_customeraddress",
ns2="http://www.ibm.com/websphere/
crossworlds/2002/BOSchema/sap_customerbankdetail",
ns3="http://www.ibm.com/websphere/
crossworlds/2002/BOSchema/sap_bapi_customer_getdetail2"');

CREATE NICKNAME sap_bapi_customer_getdetail2_sap_customerbankdetail_NN(,
   CUSTOMER VARCHAR(10) OPTIONS(XPATH './ns2:CUSTOMER/text()',
      TEMPLATE '<ns2:CUSTOMER>&column</ns2:CUSTOMER>'),
   BANK_KEY VARCHAR(16) OPTIONS(XPATH './ns2:BANK_KEY/text()',
      TEMPLATE '<ns2:BANK_KEY>&column</ns2:BANK_KEY>'),
   BANK_ACCT VARCHAR(18) OPTIONS(XPATH './ns2:BANK_ACCT/text()',
      TEMPLATE '<ns2:BANK_ACCT>&column</ns2:BANK_ACCT>'),
   CTRL_KEY VARCHAR(2) OPTIONS(XPATH './ns2:CTRL_KEY/text()',
      TEMPLATE '<ns2:CTRL_KEY>&column</ns2:CTRL_KEY>'),
   BANK_REF VARCHAR(20) OPTIONS(XPATH './ns2:BANK_REF/text()',
      TEMPLATE '<ns2:BANK_REF>&column</ns2:BANK_REF>'),
   NN__FKEY VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(FOREIGN_KEY 'SAP_BAPI_CUSTOMER_GETDETAIL2_NN'))
FOR SERVER sap_server
OPTIONS(XPATH './ns3:sap_customerbankdetail/ns2:sap_customerbankdetail',
   TEMPLATE '<ns3:sap_customerbankdetail>
      &CUSTOMER[0,1]
      &BANK_KEY[0,1]
      &BANK_ACCT[0,1]
      &CTRL_KEY[0,1]
      &BANK_REF[0,1]
   </ns3:sap_customerbankdetail>',
NAMESPACES 'ns1="http://www.ibm.com/websphere/
crossworlds/2002/BOSchema/sap_customeraddress",
ns2="http://www.ibm.com/websphere/
crossworlds/2002/BOSchema/sap_customerbankdetail",
ns3="http://www.ibm.com/websphere/
crossworlds/2002/BOSchema/sap_bapi_customer_getdetail2"');

Figure 34. Customer bank detail nickname

Example 3: Primary and foreign keys

The PRIMARY_KEY and FOREIGN_KEY columns are used to define relationships between the parent and child nicknames. Each parent nickname must have a primary key column option. You define the children of a parent nickname with the
foreign key column option that references the primary key column of a parent nickname. A nickname can have multiple children, but a nickname can have only one parent.

Primary and foreign key values for the WebSphere Business Integration wrapper are only valid and unique within a single query. A primary and foreign key cannot be used to retrieve a row with a second query. The values cannot persist into another table, because the uniqueness of the value is not guaranteed if that table is populated with multiple queries.

The following CREATE NICKNAME statements are derived from the XML schema definition files that are shown in figures 2, 3, and 4. The foreign key, nn_fkey, uniquely associates the child nickname, sap_bapi_customer_getdetail2_sap_customerbankdetail_nn, with the parent nickname sap_bapi_customer_getdetail2_nn. The parent nickname also uses a reference to the child nickname in the nickname options template structure.

CREATE NICKNAME sap_bapi_customer_getdetail2_NN(
   CUSTOMER VARCHAR(10)
   OPTIONS(XPATH './ns3:sap_customeraddress/ns1:sap_customeraddress/ns1:CUSTOMER/text()'),
   NAME VARCHAR(35)
   OPTIONS(XPATH './ns3:sap_customeraddress/ns1:sap_customeraddress/ns1:NAME/text()'),
   ...
   NN__PKEY VARCHAR(16) FOR BIT DATA NOT NULL
   OPTIONS(PRIMARY_KEY 'YES'),
   ...
   TEMPLATE '<ns3:sap_bapi_customer_getdetail2>
      &sap_bapi_customer_getdetail2_sap_customerbankdetail_NN[0,1]
      &COMPANYCODE[0,1]
      &CUSTOMERNO[1,1]
   </ns3:sap_bapi_customer_getdetail2>',

CREATE NICKNAME sap_bapi_customer_getdetail2_sap_customerbankdetail_NN(
   CUSTOMER VARCHAR(10) OPTIONS(XPATH './ns2:CUSTOMER/text()',
   BANK_KEY VARCHAR(15) OPTIONS(XPATH './ns2:BANK_KEY/text()',
   BANK_ACCT VARCHAR(18) OPTIONS(XPATH './ns2:BANK_ACCT/text()',
   CTRL_KEY VARCHAR(2) OPTIONS(XPATH './ns2:CTRL_KEY/text()',
   BANK_REF VARCHAR(20) OPTIONS(XPATH './ns2:BANK_REF/text()',
   NN__FKEY VARCHAR(16) FOR BIT DATA NOT NULL
   OPTIONS(FOREIGN_KEY 'sap_bapi_customer_getdetail2_NN'))
   FOR SERVER sap_server
   OPTIONS(XPATH 'ns3:sap_customerbankdetail/ns2:sap_customerbankdetail',
   TEMPLATE '<ns2:sap_customerbankdetail/ns2:sap_customerbankdetail'>
      &CUSTOMER[0,1]
      &BANK_KEY[0,1]
      &BANK_ACCT[0,1]
      &CTRL_KEY[0,1]
      &BANK_REF[0,1]
   </ns2:sap_customerbankdetail>',

...
Example 4: Namespaces to resolve XPath expression prefixes

The NAMESPACES option is a comma separated list of name-value pairs. It resolves the prefixes that are used in XPath expressions with the namespace URIs that are defined in the XML schemas. These XPath expressions are applied on the business objects (the XML document) that are returned from WebSphere Business Integration Adapter. The following example includes namespace prefixes and the definitions of those prefixes:

```sql
CREATE NICKNAME sap_customer
(
  sap_customeraddress_CUSTOMER VARCHAR(10)
  OPTIONS(XPATH './ns5:sap_customeraddress/ns2:CUSTOMER/text()'),
  sap_customeraddress_NAME VARCHAR(35)
  OPTIONS(XPATH './ns5:sap_customeraddress/ns2:NAME/text()'),
  ...

  sap_bapi_customer_getdet1_PKEY VARCHAR(16) FOR BIT DATA NOT NULL
  OPTIONS(PRIMARY_KEY 'YES'),
  COMPANYCODE VARCHAR(4)
  OPTIONS(XPATH './ns5:COMPANYCODE/text()',
           TEMPLATE '<ns5:COMPANYCODE>&column</ns5:COMPANYCODE>')
)
FOR SERVER SAP_SOURCE
OPTIONS (XPATH './ns5:sap_bapi_customer_getdetail2',
          TEMPLATE '<ns5:sap_bapi_customer_getdetail2>
                     &customerbankdetail_NN[0,1] &COMPANYCODE[0,1] &CUSTOMERNO[1,1]
                    </ns5:sap_bapi_customer_getdetail2>',
          BUSOBJ_NAME 'sap_bapi_customer_getdetail2',
          NAMESPACES

  ...

);
```

Nicknames and XPath expressions

The WebSphere Business Integration wrapper and the Web services wrapper use nicknames that correspond to the tree structure of XML data. Each nickname is defined by XPath expressions that represent output values.

Nicknames correspond to the tree structure of your XML document data. Parent nicknames and child nicknames correspond to the root structure and nested elements of the data tree structure. These parent and child nicknames are connected by primary and foreign keys that are specified with the CREATE NICKNAME statement.

Each nickname is defined by XPath expressions that represent output values. The WebSphere Business Integration wrapper and the Web services wrapper use XPath...
expressions to establish a correspondence between the data in an XML document and the rows in a relational table. These XPath expressions identify the values in the XML document and determine how these values correspond to the columns of each row. The WebSphere Business Integration wrapper and the Web services wrapper read the XML document data only. The wrappers do not update the data. For the Web services wrapper, the XPath option contains the information to find the SOAP messages through the SOAP envelope and SOAP body tags. The getQuote message is contained in the SOAP envelope and body elements.

The XPath expression for the NICKNAME option points to repeating tags that are in the output element. The XPath expression determines how many or which rows will be in the nickname. The column option XPath expression is relative to the NICKNAME XPath expression. The column option XPath identifies the values in a row. A NICKNAME option XPath in a child nickname is relative to a NICKNAME option XPath expression in a parent nickname.

When you create a nickname, you choose options that specify the association between the nickname and the XML document. Nicknames that are created for WebSphere Business Integration wrappers are associated with an XML schema definition (XSD) document. Nicknames that are created for Web services wrappers are associated with a Web services description language (WSDL) document.

**TEMPLATE option at the nickname and column levels**

The WebSphere Business Integration wrapper and the Web services wrapper need the nickname-level and the column-level template fragments, which is the TEMPLATE option on the CREATE NICKNAME statement, at the time that the nickname is created.

The WebSphere Business Integration wrapper and the Web services wrapper build XML documents that are required by the WebSphere Business Integration Adapter and the Web services environment. The wrappers use this information during the query planning and the query execution phases.

---

**Query restrictions for wrappers for business applications and Web services**

The use of query predicates and joins are restricted for business applications and Web services as shown in examples provided.

**Equal predicates**

The only valid predicates on input columns are equal predicates. For output columns, any predicate is valid.

The following example returns an error with a message that indicates that the predicate is not supported on that column. In this example, the column zipcode is an input column:

```sql
SELECT return FROM gettemp WHERE zipcode='95141'
```

The following example shows a valid query using an equal predicate on the input columns. The customers nickname is joined with a local table that contains customer IDs. The query contains an additional predicate on the Sales column, which is an output-only column.
SELECT a.name, a.address
FROM customers a, local_table b
WHERE
  a.customer_id=b.custid AND
  a.Sales > 300000;

**Predicates for required input columns**

You must provide equality predicate values for all required input columns in your SQL queries for the nickname hierarchy that you reference. The wrapper returns an SQLCODE -901 for all queries that violate this restriction.

**IN or OR predicates**

For WebSphere Business Integration wrappers and Web services wrappers, no IN or OR predicates are allowed for input columns.

The following examples show nonvalid queries. The customers nickname has one required input column, customer_id:

```sql
SELECT * FROM customers
  WHERE customer_id IN (12345, 67890, 11223);
SELECT * FROM customers
  WHERE customer_id IN (SELECT custid FROM local_table);
```

However, for the WebSphere Business Integration wrappers, you can use IN list predicates with required input columns if you define a unique index with the SPECIFICATION ONLY parameter for the required input columns:

```sql
CREATE UNIQUE INDEX myuindex ON customers(customer_id) SPECIFICATION ONLY;
```

**Joins on optional input columns**

The following examples demonstrate a restriction on joining optional input columns and on using host variables. You cannot join optional input columns from a local table or nickname. If the WSDL generates an input nickname column as optional and you need to use that column in a join, then you must edit the DDL statement to change the column to a required input column.

In this example, a Web service wrapper nickname named order is created with shipping_method as an optional input column. The following statement is a valid query because it uses a literal in the predicate:

```sql
SELECT * FROM order
  WHERE part="hammer" AND shipping_method="FEDEX"
```

However, if you include a local table named orderparts, which defines parts and shipping methods, in the query, and the table contains a column named shipping_method that is optional, the statement is nonvalid:

```sql
SELECT * FROM order o, orderparts op
  WHERE
    o.part="hammer" AND
    o.shipping_method=op.shipping_method
```

If you use host variables in queries with predicates on optional input columns, the statement is also nonvalid:
SELECT * FROM order o
    WHERE o.part = 'hammer' AND o.shipping_method = :ashipping_method

For a WebSphere Business Integration wrapper, predicates on optional input columns of a nickname might be pushed down to the WebSphere Business Integration Adapter. The federated server can decide to apply those predicates locally on the rows that are fetched from the application data source. To ensure that predicates for a given input column always get pushed down to the adapter, declare the input column as a required input column. Every query on the nickname hierarchy must include predicate values for the required input columns.

To ensure valid results, joined input columns must be required columns for Web Services wrappers.

**Outer joins**

Outer joins between nicknames using the primary key from a parent nickname and the foreign key from child nickname columns are not supported.

When a parent element in an XML document contains no child elements, and if you use an inner join between the parent nickname and the child nickname, then no rows are returned for that element. For example, for a given customer, if there is no bankdetail information in the SAP system, then no rows are returned for the sap_bapi_customer_getdetail2_sap_customernumberdetail_NN nickname for the particular customer.

The following CREATE NICKNAME statements define the columns that are used in the example query:

```sql
CREATE NICKNAME sap_bapi_customer_getdetail2_NN(
    ...
    NAME VARCHAR(35)
        OPTIONS(XPATH './/ns3:sap_customeraddress/ns1:NAME/text()'),
    ...
    NN__PKEY VARCHAR(16) OPTIONS(PRIMARY_KEY 'YES'),
    COMPANYCODE VARCHAR(4) OPTIONS(XPATH './/ns3:COMPANYCODE/text()',
        TEMPLATE '<ns3:COMPANYCODE>&column</ns3:COMPANYCODE>'),
    CUSTOMERNO VARCHAR(10) OPTIONS(XPATH './/ns3:CUSTOMERNO/text()',
        TEMPLATE '<ns3:CUSTOMERNO>&column</ns3:CUSTOMERNO>'),
    ...
    FOR SERVER sap_server
        OPTIONS(XPATH './/ns3:sap_bapi_customer_getdetail2',
            TEMPLATE '<ns3:sap_bapi_customer_getdetail2>
                &sap_bapi_customer_getdetail2_sap_customernumberdetail_NN[0,1]
                &COMPANYCODE[0,1]
                &CUSTOMERNO[1,1]
            </ns3:sap_bapi_customer_getdetail2>',
    ...
)
```

_Figure 35. Excerpt from getdetail2 nickname_
In the following example, the query returns no rows because there is an inner join condition between the two nicknames:

```sql
SELECT a.name, b.bank_key
FROM sap_bapi_customer_getdetail2_NN a,
     sap_bapi_customer_getdetail2_sap_customerbankdetail_NN b
WHERE a.customerno='1234567890'
AND a.NN__PKEY=b.NN__FKEY;
```

If a WebSphere Business Integration wrapper or a Web services wrapper nickname definition contains required input columns, then a left outer join between this nickname and any other local table or other nicknames is not supported.

### Business application data sources – Example queries

You can join parent and child nicknames, create views, and query using required input columns.

**Example 1: Joining parent and child nicknames**

If a select statement contains child nickname columns, then you must specify a join predicate with the parent nickname. The join conditions are necessary to maintain the parent-child relationships along the nickname hierarchy. Specify the primary and foreign key join conditions for each parent-child nickname pair in the hierarchy by including the hierarchy association from the child nickname that is referenced to the parent nickname for the hierarchy.

The following queries are nonvalid because they do not contain all of the elements that are necessary to maintain the nickname hierarchy:
select * from <childNickname>;
select b.col1
from <parentNickname> a,<childNickname> b
where a.required_column=<value>;

The following is an example of a valid query that maintains the nickname hierarchy:
select b.col1, a.cola
from <parentNickname> a,<childNickname> b
where a.primary_key_column=b.foreign_key_column
   and a.required_column=<value>;

In the following example, all of the required parent and child input columns are included in the predicates in the WHERE clause. The WHERE clause includes join predicates that specify an equality between the parent primary key column and a child foreign key column:
select a.customer, a.name, b.bank_key, b.bank_acct
from sap_bapi_customer_getdetail2_NN a,
sap_bapi_customer_getdetail2_sap_customerbankdetail_NN b
where a.nn__pkey=b.nn__fkey
   and a.customer = 'ABC'

Example 2: Views

You can create these types of views:

- A view that is derived from columns in a child nickname so that you can issue queries directly on the child nickname without including the parent-child join conditions in the queries.

create view view1 (customer, bankkey, bankact, customerno) as
(select b.customer, b.bank_key, b.bank_acct,
   a.customerno
from sap_bapi_customer_getdetail2_NN a,
sap_bapi_customer_getdetail2_sap_customerbankdetail_NN b
where a.nn__pkey=b.nn__fkey
   and a.customer = 'ABC');

The view definition includes the required input column, customerno, for the nickname hierarchy. Queries that use view view1 must include predicate values for the required column, such as in the following example:
select * from view1
where customerno='1234567890';

- A global view that includes all columns of the child and parent nicknames except for the primary and foreign key columns.

Example 3: Required input columns

Queries must include predicate values for all required input columns. Required input columns are those columns with TEMPLATE column options definitions and a reference value of [1,1] in the nickname TEMPLATE option value. The wrapper returns an error for any query that does not include the required input columns in the predicates.

The following example shows a nonvalid query. The customers nickname contains a required input column customer_id.
select * from customers;

The following example shows a valid query.
SELECT * FROM customers WHERE customer_id = 123;

The following example shows a local table in a database that contains customer IDs in the custid column of table local_table. This example is an inner join between the business integration nickname and the local table.

SELECT a.name, a.address
FROM customers a, local_table b
WHERE a.customer_id=b.custid;

Chapter 19. Configuring access to business application data sources  357
Chapter 20. Configuring access to XML data sources

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

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

XML wrapper

The Extensible Markup Language (XML) is a universal format for structured documents and data. XML uses tags for structuring the data in documents.

XML files use the .xml file extension. Like HTML, XML uses words that are enclosed in angle brackets (< >) as tags. The tags structure the data that is in the document.

The following document is a sample XML document.

```xml
<?xml version="1.0" encoding=UTF-8"?>
<doc>
  <customer id='123'>
    <name>...</name>
    <address>...</address>
    ...
    <order>
      <amount>...</amount>
      <date>...</date>
      <item quant='12'>
        <name>...</name>
      </item>
      <item quant='4'>...</item>
      ...
    </order>
  </customer>
  ...
  <payment>
    <number>...</number>
    <date>...</date>
  </payment>
  ...
  <customer id='124'>...</customer>
</doc>
```

Figure 37. Sample XML document

How the XML wrapper works

The XML wrapper enables you to use SQL to query the following types of data:

- External XML documents that are stored in a single file
- Multiple XML files in a directory path
Remote XML files that are referenced with a Uniform Reference Identifier (URI)  
XML documents stored in relational columns

The following figure shows how the XML wrapper works with your federated system.

With the XML wrapper, you can map XML data from an external data source into a relational schema that is composed of a set of nicknames. The structure of an XML document is logically equivalent to a relational schema in which the nested and repeating elements are modeled as separate tables with foreign keys.

The nicknames that correspond to an XML document are organized into a tree structure in which the child nicknames map to elements that are nested within the element that corresponds to the parent nickname.

A root nickname is a nickname at the top-level of a nickname hierarchy. A nonroot nickname is a nickname that has a parent nickname in a nickname hierarchy. You can have root nicknames that are not the top level element in an XML document.

When nested elements are repeated or have distinct identities with complex structures, you can provide separate nicknames for each nested element.

Child and parent nicknames are connected by primary and foreign keys that are generated by the wrapper.

XPath expressions are used to map an XML document into a relational schema that is composed of a set of nicknames. XPath is an addressing mechanism for identifying the parts of an XML file (for example, the groups of nodes and attributes within an XML document tree). The basic XPath syntax is similar to file system addressing.

Each nickname is defined by an XPath expression that identifies the XML elements representing individual tuples, and a set of XPath expressions that specifies how to extract the column values from each element.
An example of XML document mapping

The following example illustrates how:

- The sample XML document is mapped into a set of nicknames
- Parent and child relationships are established by using primary and foreign keys
- XPath expressions are used to define individual tuples and columns within each element of the document
- A query can run on the XML document after the document is registered to your federated system

The sample XML document contains a set of customer elements. Each element encloses several order and payment elements.

The order elements enclose several item elements.

The relationship among the elements is shown in the following figure.

![Tree structure of the sample XML document](image)

*Figure 39. Tree structure of the sample XML document*

From this structure, you can use the CREATE NICKNAME statement to map the XML document into a relational schema that includes four nicknames:

- customers
- orders
- payments
- items

You define relationships between the nicknames by specifying each nickname as a parent nickname or a child nickname by using special nickname column options that specify primary and foreign keys. Each parent nickname must have a special column that is designated with a primary key column option. You define the children of a parent nickname with the foreign key column option that references the primary key column of a parent nickname. The designated primary and foreign nickname columns do not correspond to data in your XML document because these nickname columns will contain keys that are generated by the wrapper. A nickname can have multiple children, but a nickname can have only one parent. The root nickname has no parent.

For the sample XML document, the customers nickname has a defined primary key, and the orders, payments, and items nicknames have defined foreign keys that point to the parent nickname. The foreign keys of the orders and payments nicknames point to the customers nickname, and the foreign key of the items nickname points to the orders nickname.
To identify the XML elements that represent individual tuples, you create one XPath expression. In this example, all the customer elements are referenced by using the '/doc/customer' XPath expression, and all the order elements are referenced by using the './order' XPath expression. The period in the './order' XPath expression indicates that the tuples of each order element are nested within the tuples of the corresponding customer element.

You create a set of XPath expressions to specify how to extract the column values from each element. In this example, the id attribute of the customer elements, now a column defined in the nickname, is referenced by using the './@id' XPath expression. The name element of the customer elements is referenced by using the './name' XPath expression. The address element of the customer elements is referenced by using the './address/@street' XPath expression.

After you map the XML document into a set of nicknames by using the CREATE NICKNAME statement, you define each nickname as a parent or child by using primary and foreign keys. You specify XPath expressions on these primary and foreign keys that define individual tuples and columns within each element of the document. You can then run SQL queries on the XML document.

### Adding XML to a federated system

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

**Before you begin**

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

**About this task**

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

**Procedure**

To add XML data sources to a federated server:

1. **Register the XML wrapper.**
2. **Register the XML server definition.**
3. **Register nicknames for the XML data sources.**
4. **Create federated views for non-root nicknames.**

### Registering the XML wrapper

You must register a wrapper to access XML data sources.

**About this task**

Wrappers are used by federated servers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.
You can register a wrapper by using the Control Center or from the command line. The Control Center includes a wizard to guide you through the steps that are necessary to register the wrapper.

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

**Procedure**

To register the XML wrapper:

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Start the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects.</td>
</tr>
<tr>
<td>From the command line</td>
<td>Issue the CREATE WRAPPER statement.</td>
</tr>
<tr>
<td></td>
<td>CREATE WRAPPER wrapper_name</td>
</tr>
<tr>
<td></td>
<td>LIBRARY library_name</td>
</tr>
<tr>
<td></td>
<td>If you use a proxy server or a keystore to access XML files, you must specify several options when you register either the XML wrapper or the server definition. To specify the proxy server information when you register the XML wrapper, the statement that you issue is:</td>
</tr>
<tr>
<td></td>
<td>CREATE WRAPPER wrapper_name</td>
</tr>
<tr>
<td></td>
<td>LIBRARY library_name</td>
</tr>
<tr>
<td></td>
<td>OPTIONS (PROXY_TYPE 'type',</td>
</tr>
<tr>
<td></td>
<td>PROXY_SERVER_NAME 'server_name',</td>
</tr>
<tr>
<td></td>
<td>PROXY_SERVER_PORT 'port_number');</td>
</tr>
</tbody>
</table>

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

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

**XML wrapper library files**

The XML wrapper library files are added to the federated server when you install the federated server.

When you install IBM WebSphere Federation Server, three library files are added to the default directory path. For example, if the federated server is running on AIX, the wrapper library files that are added to the directory path are libdb2lsxml.a, libdb2lsxmlF.a, and libdb2lsxmlU.a. The default wrapper library file is libdb2lsxml.a. The other wrapper library files are used with specific wrapper options.

You must include the LIBRARY parameter in the CREATE WRAPPER statement and specify the default wrapper library file name.
The default directory paths and default wrapper library file names are listed in the following table.

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Directory path</th>
<th>Wrapper library file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/&lt;install_path&gt;/lib32/</td>
<td>libdb2lsxml.a</td>
</tr>
<tr>
<td></td>
<td>/usr/opt/&lt;install_path&gt;/lib64/</td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib32</td>
<td>libdb2lsxml.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib64</td>
<td></td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib32</td>
<td>libdb2lsxml.so</td>
</tr>
<tr>
<td></td>
<td>/opt/IBM/db2/&lt;install_path&gt;/lib64</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>%DB2PATH%\bin</td>
<td>db2lsxml.dll</td>
</tr>
</tbody>
</table>

<install_path> is the directory path where the federated server is installed on UNIX or Linux.

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

**CREATE WRAPPER statement - Examples for the XML wrapper**

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

**Registering a wrapper**

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

```sql
CREATE WRAPPER xml_wrapper LIBRARY 'libdb2lsxml.a';
```

- `xml_wrapper` A name that you assign to the XML wrapper. Duplicate wrapper names are not allowed.
- `LIBRARY 'libdb2lsxml.a'` The name of the wrapper library file for federated servers that use AIX operating systems.

**Registering a wrapper for an HTTP proxy server**

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

```sql
CREATE WRAPPER xml_wrapper LIBRARY 'libdb2lsxml.a'
  OPTIONS (PROXY_TYPE 'HTTP', PROXY_SERVER_NAME 'proxy_http',
           PROXY_SERVER_PORT '8080');
```

- `PROXY_TYPE 'HTTP'` Specifies the proxy type that is used to access the Internet when behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.
- `PROXY_SERVER_NAME 'proxy_http'` Specifies the proxy server name or IP address. This option is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.
**PROXY_SERVER_PORT '8080'**
Specifies the proxy server port number. This option is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

**Registering a wrapper for a SOCKS proxy server**

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

```sql
CREATE WRAPPER xml_wrapper LIBRARY 'libdb2lsxml.so'
    OPTIONS (PROXY_TYPE 'SOCKS', PROXY_SERVER_NAME 'proxy_socks',
              PROXY_SERVER_PORT '1081');
```

**LIBRARY 'libdb2lsxml.so'**
Specifies the name of the wrapper library file for federated servers that use Linux and Solaris operating systems.

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

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

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

**Registering the server definition for an XML data source**

You must register a server definition to retrieve XML documents. The server definition that you register depends on if you use a proxy server to access the XML documents.

**About this task**

To retrieve XML documents that are behind a firewall by using a proxy and a Uniform Resource Identifier (URI), you must include the proxy server options in the CREATE SERVER statement.

If you are not using a proxy server, you must still register a server definition because the hierarchy of federated objects requires that the XML files are associated with a specific server definition object.

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

**Procedure**

To register a server definition for an XML data source:
Choose the method that you want to use to register the server definition:

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Use the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects. Follow the steps in the wizard.</td>
</tr>
</tbody>
</table>
| From the command line   | Issue the CREATE SERVER statement. If you do not use a proxy server to access XML documents, the statement that you issue is:
|                         | CREATE SERVER server_definition_name WRAPPER wrapper_name;              |
|                         | If you use a proxy server to access XML documents, the statement that you issue is:
|                         | CREATE SERVER server_definition_name WRAPPER wrapper_name OPTIONS (PROXY_TYPE 'type', PROXY_SERVER_NAME 'server_name', PROXY_SERVER_PORT 'port_number'); |

If you use a proxy server to access XML documents, the protocol that you use might require that you specify an authorization ID and a password for the proxy server. You specify the authentication information as options when you register the server definition.

**CREATE SERVER statement - Examples for the XML wrapper**

Use the CREATE SERVER statement to register server definitions for the XML wrapper. The examples show the parameters that are required to access XML documents with and without a proxy server.

**Registering a server definition**

Even if you are not using a proxy server to access XML documents, you must still register a server definition. The hierarchy of federated objects requires that the XML files are associated with a specific server definition object. The statement that you issue to register a server definition is:

```
CREATE SERVER xml_server WRAPPER xml_wrapper;
```

`xml_server`  
A name that you assign to the XML server definition. Duplicate server definition names are not allowed.

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

**Server definitions when a proxy server is used**

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

- You want to retrieve data using a URI
- The URI used will retrieve data from behind a firewall, through a proxy
- The firewall or proxy used is HTTP or SOCKS
The options that you specify depend on the type of proxy server that you want to access.

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

**Registering a server definition for an HTTP proxy server**

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

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

- `xml_server_http` is a name that you assign to the XML server definition. Duplicate server definition names are not allowed.
- `WRAPPER xml_wrapper` is the wrapper name that you specified in the CREATE WRAPPER statement.
- `PROXY_TYPE 'HTTP'` specifies the proxy type that is used to access the Internet when behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.
- `PROXY_SERVER_NAME 'proxy_http'` specifies the proxy server name or IP address. This option is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.
- `PROXY_SERVER_PORT '8080'` specifies the proxy server port number. This option is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

**Registering a server definition for a SOCKS proxy server**

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

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

- `xml_server_socks` is a name that you assign to the XML server definition. Duplicate server definition names are not allowed.
- `WRAPPER xml_wrapper` is the wrapper name that you specified in the CREATE WRAPPER statement.
- `PROXY_TYPE 'SOCKS'` specifies the proxy type that is used to access the Internet when behind a firewall. The valid values are 'NONE', 'HTTP', or 'SOCKS'.
- `PROXY_SERVER_NAME 'proxy_socks'` specifies the proxy server name or IP address. This option is required if the value for the PROXY_TYPE server option is 'HTTP' or 'SOCKS'.

Chapter 20. Configuring access to XML data sources
**PROXY_SERVER_PORT '1081'**

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

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

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

```sql
CREATE SERVER xml_server_socks
WRAPPER xml_wrapper
OPTIONS (PROXY_TYPE 'SOCKS', PROXY_SERVER_NAME 'proxy_socks',
        PROXY_SERVER_PORT '1081', PROXY_AUTHID 'Martin',
        PROXY_PASSWORD 'not4me');
```

**PROXY_AUTHID 'Martin'**

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

**PROXY_PASSWORD 'not4me'**

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

**Specifying a time limit on proxy server responses**

In addition to the required server options for proxy servers, you can also specify the SOCKET_TIMEOUT server option when you register a server definition. The SOCKET_TIMEOUT server option specifies the maximum amount of time, in minutes, that the federated server will wait for the results from the proxy server. If you do not specify the SOCKET_TIMEOUT server option, there is no time limit. The federated server will wait indefinitely for the results from the proxy server.

To register a server definition and specify how long the federated server waits for a response from the proxy server, use the following statement:

```sql
CREATE SERVER xml_server_http
WRAPPER xml_wrapper
OPTIONS (PROXY_TYPE 'HTTP', PROXY_SERVER_NAME 'proxy_http',
        PROXY_SERVER_PORT '8080', SOCKET_TIMEOUT '12');
```

**SOCKET_TIMEOUT 12**

Specifies that the federated server will wait 12 minutes for a response from the proxy server.

**Access XML files using a proxy server**

If your network uses a proxy server, you must specify information about the proxy server when you register the wrapper or server definition for XML data sources.

The options that you specify depend on the type of proxy server that you want to access, and whether you are using Secure Socket Layer (SSL) or Transport Layer Security (TLS) protocols.

You can specify the proxy and SSL options when you register a wrapper or a server definition:

- If you specify these options when you register the wrapper, the nicknames that are associated with that wrapper will use the options that are set for the wrapper.
If you specify these options when you register a server definition, the nicknames that are associated with that server definition will use the options that are set for the server definition.

If you specify different values for these options when you register the wrapper and the server definition, the values that are set for the server definition take precedence over the values that are set for the wrapper.

The XML wrapper has a validation feature that might have limitations when it is used with the proxy server. The conditions in which you will see this limitation are:

- You are using the proxy feature, at the server level, and you have set the various proxy options
- The XML instance document contains a reference to an external XML schema that is located outside the firewall

If you have one of these conditions, try to change the location of your XML schema to a location inside the firewall. If you change the XML schema location, you must update the XML instance document with the new location of the XML schema.

**Proxy server options**

The following table lists the options that must you specify for each type of proxy server:

*Table 88. Options that are required with proxy servers*

<table>
<thead>
<tr>
<th>Type of proxy server</th>
<th>Required wrapper or server options</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP or SOCKS without authentication</td>
<td>PROXY_TYPE</td>
</tr>
<tr>
<td></td>
<td>PROXY_SERVER_NAME</td>
</tr>
<tr>
<td></td>
<td>PROXY_SERVER_PORT</td>
</tr>
<tr>
<td>HTTP or SOCKS with authentication</td>
<td>PROXY_TYPE</td>
</tr>
<tr>
<td></td>
<td>PROXY_SERVER_NAME</td>
</tr>
<tr>
<td></td>
<td>PROXY_SERVER_PORT</td>
</tr>
<tr>
<td></td>
<td>PROXY_AUTHID</td>
</tr>
<tr>
<td></td>
<td>PROXY_PASSWORD</td>
</tr>
</tbody>
</table>

**SSL server options**

The following table lists the options that you must specify when you use the SSL protocols:

*Table 89. Options that are required with SSL protocols*

<table>
<thead>
<tr>
<th>Federated object</th>
<th>Required options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wrapper</td>
<td>SSL_KEYSTORE_FILE</td>
</tr>
<tr>
<td></td>
<td>SSL_KEYSTORE_PASSWORD</td>
</tr>
<tr>
<td></td>
<td>SSL_VERIFY_SERVER_CERTIFICATE</td>
</tr>
<tr>
<td>Server definition</td>
<td>SSL_KEYSTORE_FILE</td>
</tr>
<tr>
<td></td>
<td>SSL_KEYSTORE_PASSWORD</td>
</tr>
<tr>
<td></td>
<td>SSL_VERIFY_SERVER_CERTIFICATE</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_CERTIFICATE_LABEL</td>
</tr>
</tbody>
</table>
**Nicknames for XML data sources**

You must register a nickname for each XML document that you want to access. Use these nicknames, instead of the document names, when you query XML data sources.

You can register nicknames for XML data sources by using the Control Center or the command line. The Control Center simplifies the process for creating the XML nicknames. Before you register the nicknames, you should understand:

**Data associations between nicknames and XML documents**

Nicknames correspond to the tree structure of your XML document data. Parent nicknames and child nicknames correspond to the root structure and nested elements of the data tree structure. These parent and child nicknames are connected by primary and foreign keys that are specified with the CREATE NICKNAME statement.

Each nickname is defined by XPath expressions that perform the following functions:
- Identifies the XML elements that represent individual tuples
- Specifies how to extract the column values from each element

The XML wrapper uses XPath expressions to establish a correspondence between the data in the XML document and the rows in a relational table. These XPath expressions identify the values within the XML document and determine how these values correspond to the columns of each row. The XML wrapper reads the XML document data only. The XML wrapper does not update this data.

When you create a nickname, you choose options that specify the association between the nickname and the XML document. Nicknames are associated with your XML documents either in a fixed manner or with source names that you specify.

With a fixed association, the nickname represents data from specific XML documents. These XML documents include:

**One local file**

You specify one XML file as your XML document.

**Multiple local files in a directory path**

You specify a directory path in which multiple XML files reside. The XML files in this directory path provide the XML document data to the nickname. All of the XML files must have the same configuration. If any XML file in the directory has a configuration that is different from the configuration of the nickname, the XML wrapper returns null values when it processes that XML data file. The directory must be either local to the federated server or accessible from a shared file system.

When scanning the directory, the XML wrapper retains and parses only those files with a .xml extension. The XML wrapper ignores all other files, including files with a .txt extension, files with a .xsd extension, and files without extensions.

Use the FILE_PATH option in the CREATE NICKNAME statement to specify fixed data from a file. Use the DIRECTORY_PATH option to specify fixed data from a directory.
When the source data is specified while the query is running, you can use the nickname to represent data from any XML document source whose schema matches the nickname definition. These XML documents include:

**Uniform Reference Identifiers**
A remote XML file that a URI refers to supplies the XML document data to the nickname. Specify this document source by using the DOCUMENT ‘URI’ nickname column option.

**Relational columns**
Columns from a relational table, view, or nickname are used as input to your XML document. Specify this document source by using the DOCUMENT ‘COLUMN’ nickname column option.

**File**
A single file that contains XML data is supplied as input while the query runs. Specify this document source by using the DOCUMENT ‘FILE’ nickname column option.

**Directory**
Multiple XML files under a specified directory path supply the data while the query runs. Specify this document source by using the DOCUMENT ‘DIRECTORY’ nickname column option.

You specify the DOCUMENT column option to indicate that the source data is supplied at query time. Specify either URI, COLUMN, FILE, or DIRECTORY with the DOCUMENT column to indicate the type of XML document source.

You cannot specify a FILE_PATH option or a DIRECTORY_PATH option with a DOCUMENT column option.

Use the STREAMING option to separate the XML document data into fragments. You can use the STREAMING option with data that is in a fixed format or data that is from source names that you specify when you run a query. The XML wrapper processes the resulting stream of XML data and extracts the information that is requested by a query fragment. The XML wrapper parses one fragment at a time. Because fragments are parsed one at a time, total memory use decreases. The processing time required to run the entire query increases depending on the memory capacity of your server. Therefore, use the STREAMING option to only parse large XML documents that are 50 megabytes or more.

You can also choose nickname option values that help you optimize queries that retrieve large amounts of XML data or data that contains multiple nested elements. These options include:
- INSTANCE_PARSE_TIME
- XPATH_EVAL_TIME
- NEXT_TIME

You can set values for these options to test and optimize the XML query. These option values control the processing time that is needed to locate elements and to parse the data in the rows of the XML document.

**The cost model facility for the XML wrapper**
The XML wrapper provides a cost model facility to optimize queries on nicknames that correspond to your XML source documents.

When you create a nickname by using the CREATE NICKNAME statement, you can specify the following nickname options to support the cost model facility:
- INSTANCE_PARSE_TIME
• **XPATH_EVAL_TIME**

You can use the default values for these nickname options. Or you can set the values for these nickname options to optimize queries on the root and nonroot nicknames that you create.

The `INSTANCE_PARSE_TIME` nickname option is the amount of time, in milliseconds, that is required to read and parse one row-producing root element of the root nickname. The parsing time includes all contained row-producing nonroot elements. For example if the root nickname is customers, the nonroot elements are all of the elements that correspond to the orders, payments, and items of each customer. The XML wrapper builds a structure in memory to represent these row-producing root and nonroot elements.

The `XPATH_EVAL_TIME` nickname option is the amount of time, in milliseconds, that is required to evaluate the XPath expressions that locate the data corresponding to a row of the nickname. The XPath expressions that are evaluated include the XPath expressions that locate the actual rows and the XPath expressions that locate column values within these rows.

**Namespaces for XML data sources**

Use the NAMESPACES nickname column option to identify the elements or attributes that are part of a namespace.

You can specify the NAMESPACES nickname option when you register nicknames. The value of the NAMESPACES nickname column option is a comma-separated list of name and value pairs. The XML wrapper uses the name and value pairs to resolve the namespace prefixes that are in the nickname XPath expressions. The prefixes that are used in the XPath expressions are processed by the XPath processor.

In the following example, the XML document includes the name, code, and description information for three products. The XML document declares two namespaces, `http://www.one.com` and `http://www.two.com`, and has one default namespace `http://www.default.com`. The product element is associated with the `ns1` namespace. The product element contains the name and code attributes and the `desc` element. The name attribute is not associated with a namespace. The code attribute is associated with the `ns2` namespace. The `desc` element is associated with the default namespace.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<doc xmlns:ns1="http://www.one.com" xmlns:ns2="http://www.two.com"
     xmlns="http://www.default.com">
  <ns1:product name="Computer" ns2:code="ABC123"
                <desc>The Computer product description"/></desc>
  <ns1:product name="Keyboard" ns2:code="EFG456"
                <desc>The Keyboard product description"/></desc>
  <ns1:product name="Mouse" ns2:code="HIJ789"
                <desc>The Mouse product description"/></desc>
</ns1:product>
</doc>
```

The following table shows the namespace that is associated with each element and attribute in the XML document.

---

**Configuration Guide for Federated Data Sources**
### Table 90. The elements, attributes, and namespaces in the XML document

<table>
<thead>
<tr>
<th>Element or attribute</th>
<th>Namespace in XML document</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>product: An element in the XML document.</td>
<td>ns1=&quot;<a href="http://www.one.com">http://www.one.com</a>&quot;</td>
<td>None</td>
</tr>
<tr>
<td>name: An attribute of the product element in the XML document.</td>
<td>None</td>
<td>The name attribute is not associated with a namespace.</td>
</tr>
<tr>
<td>code: An attribute of the product element in the XML document.</td>
<td>ns2=&quot;<a href="http://www.two.com">http://www.two.com</a>&quot;</td>
<td>None</td>
</tr>
</tbody>
</table>

When you register the nickname for the XML document, you define three columns to correspond to the elements and attributes in the XML document. You specify the namespace information in the NAMESPACES nickname option. For example:

```sql
CREATE NICKNAME products
    (name VARCHAR(16) OPTIONS (XPATH '@name'),
     code VARCHAR(16) OPTIONS (XPATH '@pre2:code')
    description VARCHAR(256) OPTIONS (XPATH './default:desc'))
FOR SERVER xml_server
OPTIONS (FILE_PATH '/home/mbreining/sql/xml/namespaces.xml',
    XPATH '//pre1:name',
    NAMESPACES 'pre1="http://www.one.com", pre2="http://www.two.com",
    default="http://www.default.com"');
```

The following table shows how the XML document namespaces correspond to the values that you specify in the CREATE NICKNAME statement.

### Table 91. XML document namespaces and corresponding values from the CREATE NICKNAME statement.

<table>
<thead>
<tr>
<th>Namespace in XML document</th>
<th>Column name in the CREATE NICKNAME statement</th>
<th>Value in the XPATH column option</th>
<th>Value in the NAMESPACES nickname option</th>
</tr>
</thead>
<tbody>
<tr>
<td>The namespace is ns1=&quot;<a href="http://www.one.com">http://www.one.com</a>&quot;</td>
<td>None</td>
<td>None</td>
<td>The value is pre1=&quot;<a href="http://www.one.com">http://www.one.com</a>&quot;. The value is a prefix that you define (pre1) and the unique identifier for the namespace (&quot;<a href="http://www.one.com">http://www.one.com</a>&quot;).</td>
</tr>
<tr>
<td>None. The attribute is not associated with a namespace.</td>
<td>Name</td>
<td>The value is @name. The value is an attribute that is in the XML document (name).</td>
<td>None</td>
</tr>
<tr>
<td>The namespace is ns2=&quot;<a href="http://www.two.com">http://www.two.com</a>&quot;</td>
<td>Code</td>
<td>The value is @pre2:code. The value is a prefix that you define (pre2) and an attribute that is in the XML document (code).</td>
<td>The value is pre2=&quot;<a href="http://www.two.com">http://www.two.com</a>&quot;. The value is a prefix that you define (pre2) and the unique identifier for the namespace (&quot;<a href="http://www.two.com">http://www.two.com</a>&quot;).</td>
</tr>
</tbody>
</table>
Table 91. XML document namespaces and corresponding values from the CREATE NICKNAME statement. (continued)

<table>
<thead>
<tr>
<th>Namespace in XML document</th>
<th>Column name in the CREATE NICKNAME statement</th>
<th>Value in the XPATH column option</th>
<th>Value in the NAMESPACES nickname option</th>
</tr>
</thead>
<tbody>
<tr>
<td>The namespace is &quot;<a href="http://www.default.com">http://www.default.com</a>&quot; The default namespace does not contain a prefix.</td>
<td>Description</td>
<td>The value is ./default:desc. The value is a prefix that you define (default) and an element that is in the XML document (desc).</td>
<td>The value is default=&quot;<a href="http://www.default.com">http://www.default.com</a>&quot;. The value is a prefix that you define (default) and the unique identifier for the namespace ('<a href="http://www.default.com">http://www.default.com</a>').</td>
</tr>
</tbody>
</table>

The NAMESPACES nickname option uses packed descriptors to support strings that are greater than 256 characters.

For more information about XML namespaces, see the explanation for namespaces on the W3C Web site.

Registering nicknames for XML data sources

For each XML server definition that you register, you must register a nickname for each XML document that you want to access. Use these nicknames, instead of the XML document names, when you query the XML data sources.

Restrictions

If you attempt to access XML data sources that are on a shared drive from a federated server that runs Windows 2003, your queries might fail. This is a limitation of Windows 2003. You can avoid this problem by specifying the absolute path in the FILE_PATH option in the CREATE NICKNAME statement.

You must create nicknames that correspond to the tree structure of your XML data source. Parent nicknames correspond to the root structure of the tree. Child nicknames correspond to the elements that are nested within the element for the parent nickname.

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

Procedure

To register nicknames for XML data sources:

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the Control Center</td>
<td>Use the Federated Objects wizard. Right-click the Federated Database Objects folder and click Create Federated Objects. Follow the steps in the wizard.</td>
</tr>
<tr>
<td>Method</td>
<td>Procedure</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>From the command line</td>
<td>Issue the CREATE NICKNAME statement. Nicknames can be up to 128 characters in length.</td>
</tr>
</tbody>
</table>

**CREATE NICKNAME statement - examples for XML wrapper:**

Use the CREATE NICKNAME statement to register nicknames for XML documents. There is a complete example, which shows how to create parent and child nicknames, and examples for specific column options.

**Recommendation:** Do not use the self or descendant operator // when you specify XPATH columns and nickname options in your queries. The self or descendant operator is an XPath operator. Using self or descendant operator can decrease federated server performance.

**Complete example**

The following example shows how to create nicknames for XML data sources by using the sample XML file shown in the following sample XML file.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<doc>
  <customer id='123'>
  </customer>
  <customer id='124'>
</doc>
```

**Figure 40. Sample XML file**

**The parent nickname**

The first step is to create the parent nickname, customers. To create the nickname, issue the following statement:

```
CREATE NICKNAME customers
  
  id         VARCHAR(5)  OPTIONS(XPATH '.,/id')
```
This statement creates the customers nickname over multiple XML files under the specified directory path, /home/db2user.

The STREAMING nickname option indicates that the XML source data is separated and processed by node, in this example by customer record. When the STREAMING nickname option is used, the wrapper does not storing the entire XML document into memory. Instead, the XML wrapper divides the document into multiple sections which are parsed individually and sequentially. The STREAMING nickname option should be used only with large XML documents. The performance of your queries is impacted when you use this option.

**The child nicknames**

The next step is to create the child nicknames for the orders, payments, and items elements.

Issue the following statement to create the orders child nickname:

```
CREATE NICKNAME orders

    amount INTEGER OPTIONS(XPATH './amount'),
    date VARCHAR(10) OPTIONS(XPATH './date'),
    oid VARCHAR(10) OPTIONS(PRIMARY_KEY 'YES'),
    cid VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(FOREIGN_KEY 'CUSTOMERS'))
FOR SERVER xml_server OPTIONS(XPATH './order');
```

Issue the following statement to create the payments child nickname:

```
CREATE NICKNAME payments

    number INTEGER OPTIONS(XPATH './number'),
    date VARCHAR(10) OPTIONS(XPATH './date'),
    cid VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(FOREIGN_KEY 'CUSTOMERS'))
FOR SERVER xml_server OPTIONS(XPATH './payment');
```

Issue the following statement to create the items child nickname:

```
CREATE NICKNAME items

    name VARCHAR(20) OPTIONS(XPATH './name'),
    quantity INTEGER OPTIONS(XPATH './@quant'),
    oid VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(FOREIGN_KEY 'ORDERS'))
FOR SERVER xml_server OPTIONS(XPATH './item');
```

**Nickname column option examples**

The following examples show you how to include the DOCUMENT nickname column options when you create nicknames. The examples also show you how those options are used in queries.
The following examples show you how to include the DOCUMENT nickname column options when you create nicknames. The examples also show you how to use those options in queries.

The following CREATE NICKNAME example shows the use of the DOCUMENT 'FILE' nickname column option:

```
CREATE NICKNAME customers
(
  doc VARCHAR(100) OPTIONS(DOCUMENT 'FILE'),
  name VARCHAR(16) OPTIONS(XPATH './name'),
  address VARCHAR(30) OPTIONS(XPATH './address/@street'),
  cid VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(PRIMARY_KEY 'YES'))
FOR SERVER xml_server
OPTIONS(XPATH '/doc/customer');
```

You can then run the following query on the customers nickname, specifying the location of the XML document in the WHERE clause:

```
SELECT * FROM customers WHERE doc = '/home/db2user/Customers.xml';
```

**DOCUMENT 'DIRECTORY' example**

The following CREATE NICKNAME example shows the use of the DOCUMENT 'DIRECTORY' nickname column option:

```
CREATE NICKNAME customers
(
  doc VARCHAR(100) OPTIONS(DOCUMENT 'DIRECTORY'),
  name VARCHAR(16) OPTIONS(XPATH './name'),
  address VARCHAR(30) OPTIONS(XPATH './address/@street'),
  cid VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(PRIMARY_KEY 'YES'))
FOR SERVER xml_server
OPTIONS(XPATH '/doc/customer');
```

You can then run the following query on the customers nickname:

```
SELECT name FROM customers WHERE doc = '/home/data/xml';
```

This query retrieves the XML documents that are located under the directory path `/home/data/xml`, which is specified in the WHERE clause.

**DOCUMENT 'URI' example**

The following CREATE NICKNAME example shows the use of the DOCUMENT 'URI' nickname column option:

```
CREATE NICKNAME customers
(
  doc VARCHAR(100) OPTIONS(DOCUMENT 'URI'),
  name VARCHAR(16) OPTIONS(XPATH './name'),
  address VARCHAR(30) OPTIONS(XPATH './address/@street'),
  cid VARCHAR(16) FOR BIT DATA NOT NULL OPTIONS(PRIMARY_KEY 'YES'))
FOR SERVER xml_server
OPTIONS(XPATH '/doc/customer');
```

You can then run the following query on the customers nickname to retrieve the XML data from the remote location:

```
SELECT * FROM customers WHERE doc = 'http://www.lg-mv.org/foo.xml';
```
**DOCUMENT ‘COLUMN’ example**

The following CREATE NICKNAME example shows the use of the DOCUMENT ‘COLUMN’ nickname column option:

```sql
CREATE NICKNAME emp
(
    doc VARCHAR(500) OPTIONS(DOCUMENT ‘COLUMN’),
    fname VARCHAR(16) OPTIONS(XPATH '@first'),
    lname VARCHAR(16) OPTIONS(XPATH '@last'))
FOR SERVER xml_server
OPTIONS(XPATH '/doc/name');
```

You can then run one of the following queries on the emp nickname to retrieve the XML data:

```sql
SELECT * FROM emp WHERE doc = '<?xml version="1.0" encoding="UTF-8"?><doc>
   <title> employees </title>
   <name first="David" last="Marston"/>
   <name first="Donald" last="Leslie"/>
   <name first="Emily" last="Farmer"/>
   <name first="Myriam" last="Midy"/>
   <name first="Lee" last="Tran"/>
   <name first="Lili" last="Farmer"/>
   <name first="Sanjay" last="Kumar"/>
</doc>'; 
```

or

```sql
SELECT * FROM emp WHERE doc = (SELECT * FROM xml_tab);
```

The xml_tab table contains one column that is populated with the XML data.

---

**Queries for XML data sources**

Before you create queries to access XML data sources, there are actions that you can take to optimize the query performance.

**Federated views**

You can use federated views to ensure that the queries that join the pieces of an XML nickname hierarchy run correctly.

**Avoid the self or descendant operator**

Do not use the self or descendant operator // when you specify XPATH columns and nickname options when you create the XML nicknames. The self or descendant operator is an XPath operator, and using the self or descendant operator can decrease federated server performance.

**Windows 2003 federated servers**

If you attempt to access XML data sources that are on a shared drive from a federated server that runs Windows 2003, your query might fail with the following error message:

SQL1822N Unexpected error code "ERRNO = 2" received from data source "XML_SERVER". Associated text and tokens are "Unable to read file". SQLSTATE=560BD
This is a limitation of Windows 2003. You can avoid this problem by specifying the absolute path in the FILE_PATH option in the CREATE NICKNAME statement.

The following example shows a CREATE NICKNAME statement with an abbreviated path specified in the FILE_PATH option:

```
CREATE NICKNAME customers
(
  id  VARCHAR(5) OPTIONS(XPATH './@id'),
  name VARCHAR(16) OPTIONS(XPATH './name'),
  address VARCHAR(30) OPTIONS(XPATH './address/@street'),
  cid  VARCHAR(16) FOR BIT DATA NOT NULL
  OPTIONS(PRIMARY_KEY 'YES'))
FOR SERVER xml_server
  OPTIONS(DIRECTORY_PATH '\home\db2user',
         XPATH '/doc/customer', STREAMING 'YES');
```

Queries that use this nickname might fail because you specified the abbreviated path.

For federated servers that run Windows 2003, specify the absolute path in the FILE_PATH option in the CREATE NICKNAME statement.

For example:

```
CREATE NICKNAME customers
(
  id  VARCHAR(5) OPTIONS(XPATH './@id'),
  name VARCHAR(16) OPTIONS(XPATH './name'),
  address VARCHAR(30) OPTIONS(XPATH './address/@street'),
  cid  VARCHAR(16) FOR BIT DATA NOT NULL
  OPTIONS(PRIMARY_KEY 'YES'))
FOR SERVER xml_server
  OPTIONS(DIRECTORY_PATH '\home\db2user',
         XPATH '/doc/customer', STREAMING 'YES');
```

Creating federated views for the XML wrapper nicknames

You can create federated views over the hierarchy of nicknames that describe an XML document. Defining federated views ensures that the queries that join the pieces of an XML nickname hierarchy run properly.

About this task

A federated view is a view in the federated database that references a nickname, instead of a data source table.

In the XML nickname hierarchy the root nickname and queries that join columns, other than the special PRIMARY_KEY and FOREIGN_KEY columns, are not impacted by using federated views.

When you create federated views for XML nicknames, you should include all of the required predicates and a full path to the root directory.

Procedure

To create federated views for XML nicknames:

Use the CREATE VIEW statement to define a view for each nonroot nickname. The view should be a join of all the nicknames on the path to the root nickname.
1. In the WHERE clause of the view, define the PRIMARY_KEY and FOREIGN_KEY columns as the join predicates.

2. In the SELECT list, include all the columns of the nonroot nickname except the column that is designated with the FOREIGN_KEY nickname column option. In the SELECT list, include the column of the parent nickname designated with the PRIMARY_KEY option.

**CREATE VIEW statement - examples for the XML wrapper**

Use the CREATE VIEW statement to create federated views for nonroot nicknames. This example includes a sample XML file, the statements that you use to create the views, and shows how you can use the views in a query.

You can create federated views over the hierarchy of nicknames that describe an XML document to ensure that the queries that join pieces of an XML nickname hierarchy run correctly. When you specify a federated view in a query, data is retrieved from the remote data source.

The following examples show you how to create federated views for nonroot nicknames to describe XML source documents.

**Sample XML file**

The following examples are based on this sample XML file.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<doc>
  <customer id='123'>
    <name>...</name>
    <address>...</address>
    ...
    <order>
      <amount>...</amount>
      <date>...</date>
      <item quant='12'>
        <name>...</name>
      </item>
      <item quant='4'>...</item>
      ...
    </order>
    <order>...</order>
    ...
    <payment>
      <number>...</number>
      <date>...</date>
    </payment>
    <payment>...</payment>
    ...
  </customer>
  <customer id='124'>...</customer>
</doc>
```

*Figure 41. Sample XML file*

**CREATE VIEW statements for the nonroot nicknames**

The following example shows how to create a federated view for the nonroot nickname order:
CREATE VIEW order_view AS
  SELECT o.amount, o.date, o.oid, c.cid
  FROM customers c, orders o
  WHERE c.cid = o.cid;

The following example shows how to create a federated view for the nonroot nickname payment:
CREATE VIEW payment_view AS
  SELECT p.number, p.date, c.cid
  FROM customers c, payments p
  WHERE c.cid = p.cid;

The following example shows how to create a federated view for the nonroot nickname item:
CREATE VIEW item_view AS
  SELECT i.quantity, i.name, o.oid
  FROM customers c, orders o, items i
  WHERE c.cid = o.cid AND o.oid = i.oid;

A query that uses the federated views

Queries that are submitted to the federated views are processed correctly because the join path to the root directory is specified in the WHERE clause.

For example, the following query uses the customer identification number and the date that an order was placed to return the amount that was ordered and the amount of the payment due. Instead of using the nicknames in the query, the views are specified in the FROM clause.
SELECT o.amount, p.amount
FROM order_view o, payment_view p
WHERE p.date = o.date AND
  p.cid = o.cid;

Query optimization tips for the XML cost model facility

The cost model facility for the XML wrapper helps optimize queries on the nicknames that you create.

The cost model facility uses the following nickname options of the CREATE NICKNAME statement:
- INSTANCE_PARSE_TIME
- XPATH_EVAL_TIME

You can specify values for these nickname options when you issue a CREATE NICKNAME statement to register a nickname for an XML data source.

The cost model facility uses these parameter values to determine the amount of time required to parse the data in each row of an XML source document. The parameter values are also used to evaluate the XPath expression for the nickname.

You can use the default values for these nickname options. However, if you want to optimize queries on large or complex XML source structures for the nicknames that you create, use the following example as a guide.
An example of optimizing a large query

Your XML document has a relational schema with four nicknames:

- customers
- orders
- payments
- items

The root nickname is customers.

Run queries on each nickname. Run each query on a sample of the XML data that is typical for your environment.

For example:
```
SELECT * from customers;
SELECT * from orders;
SELECT * from payments;
SELECT * from items;
```

Make a note of the amount of time (in milliseconds) that is required to run each query by using the db2batch command, or an equivalent command or utility. You can use the db2batch command to obtain an output file that contains the time required to run queries. Make a note of the number of tuples that are returned.

For each nickname, use the following formulas to determine the optimal values for the INSTANCE_PARSE_TIME and XPATH_EVAL_TIME nickname options:

```
INSTANCE_PARSE_TIME =
(75% \times \text{run time of SELECT }\ast \text{ query}) + \text{number of tuples returned}

XPATH_EVAL_TIME =
(25% \times \text{run time of SELECT }\ast \text{ query}) + \text{number of tuples returned}
```

For the root nickname (in this example, customers), use the calculated values for the INSTANCE_PARSE_TIME and XPATH_EVAL_TIME nickname options.

For nonroot nicknames, (in this example, orders, payments, and items), use only the calculated value for the XPATH_EVAL_TIME parameter. The INSTANCE_PARSE_TIME parameter value is not applicable for nonroot nicknames.

You can use these formulas as a guide for tuning your queries. The optimal values for these nickname options also depend on the complexity of your XML source documents and on the speed of the processor that you are using.

XML data source - example queries

Examples of queries using XML nicknames.

These examples use the customers, orders, and items nicknames.

A query that returns a specific value from the XML documents

When the following SELECT statement is run, the wrapper returns the names of all of the customers:
```
SELECT name FROM customers;
```
A query that returns all of the records for a specific customer

When the following SELECT statement is run, the wrapper returns all of the records in which the customer name is Chang:

```
SELECT * FROM customers
  WHERE name='Chang';
```

A query that returns specific values based on a join condition between a parent and child nickname

When the following SELECT statement is run, the wrapper returns the customer names and amounts for each order placed by each customer. You must specify the join, c.cid=o.cid, to indicate the parent-child relationship between the customers nickname and the orders nickname.

```
SELECT c.name, o.amount FROM customers c, orders o
  WHERE c.cid=o.cid;
```

A query that shows how to specify join conditions between a parent and several child nicknames

When the following SELECT statement is run, the wrapper returns the customer addresses, order amounts, and item names for each order and item of each customer. You must specify the two joins to maintain the parent-child relationships.

```
SELECT c.address, o.amount, i.name FROM customers c, orders o, items i
  WHERE c.cid=o.cid AND o.oid=i.oid;
```

Queries that show how specify an XML document in a query

The following examples show you how to write queries by using a nickname that specifies a DOCUMENT nickname column option rather than a FILE_PATH nickname option.

The CREATE NICKNAME statement that is used to create the customers nickname is:

```
CREATE NICKNAME customers
  (  
    doc VARCHAR(100) OPTIONS(DOCUMENT 'FILE'),
    name VARCHAR(16) OPTIONS(XPATH './name'),
    address VARCHAR(30) OPTIONS(XPATH './address/@street'),
    cid VARCHAR(16) OPTIONS(PRIMARY_KEY 'YES'))
  FOR SERVER xml_server
  OPTIONS(XPATH '/doc/customer');
```

The following query selects all of the data from the XML file Customers.xml with a path of /home/db2user/Customers.xml:

```
SELECT * FROM customers
  WHERE doc='/home/db2user/Customers.xml';
```

The following query selects the names of customers and the dates of their orders from the Customers.xml file for each order with an amount over 1000. The path /home/db2user/Customers.xml specifies the location of the Customers.xml file.

```
SELECT c.name, o.date FROM customers c, orders o
  WHERE c.doc='/home/db2user/Customers.xml' AND o.amount > 1000;
```
Chapter 21. Data source support for federated features

Refer to this table when you want to know whether or not a data source supports a specific federated feature.

Before you can use some of these features, you may need to set specific wrapper or server options or perform other tasks to enable the functionality. For more information, see the specific topics on each feature.

Table 92. Features and supported data sources

<table>
<thead>
<tr>
<th>Feature</th>
<th>Data sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application savepoints with WRITE operations against nicknames</td>
<td>DB2 for Linux, UNIX, and Windows</td>
</tr>
<tr>
<td>Asynchrony optimization</td>
<td>All data sources</td>
</tr>
<tr>
<td>Cache tables</td>
<td>DB2 family</td>
</tr>
<tr>
<td></td>
<td>Informix</td>
</tr>
<tr>
<td></td>
<td>Microsoft SQL Server</td>
</tr>
<tr>
<td></td>
<td>Oracle</td>
</tr>
<tr>
<td></td>
<td>Sybase</td>
</tr>
<tr>
<td>Data import into nicknames</td>
<td>DB2 family</td>
</tr>
<tr>
<td></td>
<td>Informix</td>
</tr>
<tr>
<td></td>
<td>Microsoft SQL Server</td>
</tr>
<tr>
<td></td>
<td>Oracle</td>
</tr>
<tr>
<td></td>
<td>Sybase</td>
</tr>
<tr>
<td></td>
<td>Teradata</td>
</tr>
<tr>
<td>Error tolerance in nested table expressions</td>
<td>DB2 family</td>
</tr>
<tr>
<td></td>
<td>Informix</td>
</tr>
<tr>
<td></td>
<td>Microsoft SQL Server</td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
</tr>
<tr>
<td></td>
<td>Oracle</td>
</tr>
<tr>
<td></td>
<td>Sybase</td>
</tr>
<tr>
<td></td>
<td>Teradata</td>
</tr>
<tr>
<td>External user mapping repository</td>
<td>All data sources</td>
</tr>
<tr>
<td>Federated health indicators</td>
<td>DB2 family</td>
</tr>
<tr>
<td></td>
<td>Excel</td>
</tr>
<tr>
<td></td>
<td>Informix</td>
</tr>
<tr>
<td></td>
<td>Microsoft SQL Server</td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
</tr>
<tr>
<td></td>
<td>Oracle</td>
</tr>
<tr>
<td></td>
<td>Sybase</td>
</tr>
<tr>
<td></td>
<td>Table-structured files</td>
</tr>
<tr>
<td></td>
<td>Teradata</td>
</tr>
<tr>
<td></td>
<td>XML (root nicknames only)</td>
</tr>
<tr>
<td>Federated procedures</td>
<td>DB2 family, in trusted mode</td>
</tr>
<tr>
<td></td>
<td>Oracle, in trusted mode</td>
</tr>
<tr>
<td></td>
<td>Microsoft SQL Server, in trusted mode</td>
</tr>
<tr>
<td></td>
<td>Sybase, in fenced mode, with the federated server installed on UNIX</td>
</tr>
<tr>
<td></td>
<td>Sybase, in fenced mode or trusted mode, with the federated server installed on Linux or Microsoft Windows</td>
</tr>
</tbody>
</table>
Table 92. Features and supported data sources (continued)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Data sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Federated trusted contexts</td>
<td>DB2 for Linux, UNIX, and Windows Version 9.5</td>
</tr>
<tr>
<td></td>
<td>DB2 for z/OS Version 9</td>
</tr>
<tr>
<td></td>
<td>Oracle</td>
</tr>
<tr>
<td>HTTP proxy</td>
<td>Entrez</td>
</tr>
<tr>
<td></td>
<td>Web services</td>
</tr>
<tr>
<td></td>
<td>XML</td>
</tr>
<tr>
<td>Connection-level isolation</td>
<td>DB2 family</td>
</tr>
<tr>
<td></td>
<td>Informix</td>
</tr>
<tr>
<td></td>
<td>Microsoft SQL Server</td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
</tr>
<tr>
<td></td>
<td>Oracle</td>
</tr>
<tr>
<td></td>
<td>Sybase</td>
</tr>
<tr>
<td>Label-based access control</td>
<td>DB2 for Linux, UNIX, and Windows Version 9.1 and 9.5</td>
</tr>
<tr>
<td></td>
<td>Oracle</td>
</tr>
<tr>
<td>LOB read and write operations</td>
<td>DB2 for z/OS</td>
</tr>
<tr>
<td></td>
<td>DB2 for Linux, UNIX, and Windows</td>
</tr>
<tr>
<td></td>
<td>DB2 for System i™</td>
</tr>
<tr>
<td></td>
<td>Oracle</td>
</tr>
<tr>
<td>LOB read-only operations</td>
<td>BioRS*</td>
</tr>
<tr>
<td></td>
<td>BLAST*</td>
</tr>
<tr>
<td></td>
<td>Entrez*</td>
</tr>
<tr>
<td></td>
<td>HMMER*</td>
</tr>
<tr>
<td></td>
<td>Informix</td>
</tr>
<tr>
<td></td>
<td>Microsoft SQL Server</td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
</tr>
<tr>
<td></td>
<td>Script</td>
</tr>
<tr>
<td></td>
<td>Sybase</td>
</tr>
<tr>
<td></td>
<td>Teradata</td>
</tr>
<tr>
<td></td>
<td>Web services</td>
</tr>
<tr>
<td></td>
<td>WebSphere Business Integration*</td>
</tr>
<tr>
<td></td>
<td>XML</td>
</tr>
<tr>
<td>Materialized query tables</td>
<td>All data sources, with specific restrictions</td>
</tr>
<tr>
<td>Nickname statistics update facility</td>
<td>BioRS*</td>
</tr>
<tr>
<td></td>
<td>DB2 family</td>
</tr>
<tr>
<td></td>
<td>Excel</td>
</tr>
<tr>
<td></td>
<td>Informix</td>
</tr>
<tr>
<td></td>
<td>Microsoft SQL Server</td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
</tr>
<tr>
<td></td>
<td>Oracle</td>
</tr>
<tr>
<td></td>
<td>Sybase</td>
</tr>
<tr>
<td></td>
<td>Table-structured files</td>
</tr>
<tr>
<td></td>
<td>Teradata</td>
</tr>
<tr>
<td></td>
<td>XML (root nicknames only)</td>
</tr>
<tr>
<td>Pass-through sessions</td>
<td>DRDA</td>
</tr>
<tr>
<td></td>
<td>Informix</td>
</tr>
<tr>
<td></td>
<td>Oracle</td>
</tr>
<tr>
<td></td>
<td>Microsoft SQL Server</td>
</tr>
<tr>
<td></td>
<td>Sybase</td>
</tr>
<tr>
<td></td>
<td>Teradata</td>
</tr>
<tr>
<td>Remote XML data type</td>
<td>DB2 for Linux, UNIX, and Windows</td>
</tr>
<tr>
<td></td>
<td>XML wrapper</td>
</tr>
</tbody>
</table>
Table 92. Features and supported data sources (continued)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Data sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCKS proxy</td>
<td>Blast*, BioRS*, Entrez*, HMMER*, Script, Web services, XML</td>
</tr>
<tr>
<td>Secure Socket Layer (SSL)</td>
<td>Web services, XML</td>
</tr>
<tr>
<td>Statement-level isolation</td>
<td>DB2 family, Microsoft SQL Server</td>
</tr>
<tr>
<td>Two-phase commit transactions</td>
<td>DB2 for Linux, UNIX, and Windows, in trusted mode, DB2 for System i, in trusted mode, DB2 for z/OS, in trusted mode, Informix, in trusted mode, Microsoft SQL Server, in trusted mode, with the federated server installed on Microsoft Windows, Oracle, in trusted mode, Sybase, in trusted mode, with the federated server installed on Microsoft Windows, Sybase, in fenced mode, with the federated server installed on UNIX</td>
</tr>
<tr>
<td>Unicode support</td>
<td>All data sources</td>
</tr>
</tbody>
</table>

* As of IBM WebSphere Federation Server Version 9.5, this wrapper is deprecated.
Chapter 22. Data source options reference

Each data source supports specific wrapper, server, user mapping, nickname, and column options.

Blast options reference

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, user mapping, nickname, and column options.

The following tables list the options that apply to this data source and identify the required options that you must specify.

Wrapper options

Table 93. Wrapper options for Blast

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Required. Specifies whether the wrapper runs in fenced mode or in trusted mode. Valid values are Y and N. The default is N; the wrapper runs in trusted mode.</td>
</tr>
<tr>
<td>PROXY_SERVER_NAME</td>
<td>Specifies the DNS name or IP address of the proxy server. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
<tr>
<td>PROXY_SERVER_PORT</td>
<td>Specifies the port or service name for the proxy service on the proxy server.</td>
</tr>
<tr>
<td>PROXY_TYPE</td>
<td>Specifies the proxy type to use to access the Internet when the federated server is behind a firewall. Valid values are NONE and SOCKS. The default value is NONE.</td>
</tr>
</tbody>
</table>

Server options

Table 94. Server options for Blast

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAEMON_PORT</td>
<td>Specifies the port number on which the BLAST daemon listens for BLAST job requests. The default value is 4007. <strong>Important:</strong> The port number must be the same as the DAEMON_PORT option in the daemon configuration file. If a service name is configured for the BLAST daemon, use a TCP/IP service name.</td>
</tr>
</tbody>
</table>
### Table 94. Server options for Blast (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_QUERY</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. The default is 1. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests.</td>
</tr>
<tr>
<td>NODE</td>
<td>Required. Specifies the DNS host name or IP address of the system on which the BLAST daemon runs. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
<tr>
<td>PROXY_AUTHID</td>
<td>Specifies the user name for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY_PASSWORD</td>
<td>Specifies the password for proxy server authentication. The password is encrypted when it is stored in the federated database catalog.</td>
</tr>
<tr>
<td>PROXY_SERVER_NAME</td>
<td>Specifies the name or IP address of the proxy server. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
<tr>
<td>PROXY_SERVER_PORT</td>
<td>Specifies the port or service name for the proxy service on the proxy server. Valid values are a decimal port number from 1 to 32760 or a service name.</td>
</tr>
<tr>
<td>PROXY_TYPE</td>
<td>Specifies the proxy type to use to access the Internet when the federated server is behind a firewall. Valid values are NONE and SOCKS. The default value is NONE.</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>Specifies the maximum time, in minutes, that the federated server waits for a response from the remote server. The default is 60.</td>
</tr>
</tbody>
</table>

### User mapping options

#### Table 95. User mapping options for Blast

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROXY_AUTHID</td>
<td>Specifies the user name for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY_PASSWORD</td>
<td>Specifies the password for proxy server authentication. The password is encrypted when it is stored in the federated database catalog.</td>
</tr>
</tbody>
</table>
**Nickname options**

*Table 96. Nickname options for Blast*

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATASOURCE</td>
<td>Required. Specifies the name of the data source on which the BLAST search runs. You must specify the same name that you specified in the configuration file for the BLAST daemon.</td>
</tr>
<tr>
<td>PROCESSORS</td>
<td>Specifies the number of processors to use when a BLAST query is evaluated. The valid value is an integer from 1 to the total number of processors that are available on the BLAST server. This option corresponds to the blastall -a option.</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>Specifies the maximum time, in minutes, to wait for a response from the data source server. The default is 60 minutes.</td>
</tr>
</tbody>
</table>

**Column options**

*Table 97. Column options for Blast*

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>Specifies a default value for a BLAST input column. The new value overrides the preset default value. The new value must be of the same type as the indicated value for a given column.</td>
</tr>
<tr>
<td>DELIMITER</td>
<td>Required, unless you want the last specified column to contain the remainder of the definition line. Specifies one or more delimiter characters that signal the end of the definition line. If you specify more than one character, then the first occurrence of one of the characters signals the end of the field. The default is end of line.</td>
</tr>
<tr>
<td>INDEX</td>
<td>Specifies the ordinal number of the column.</td>
</tr>
</tbody>
</table>

**BioRS options reference**

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, user mapping, nickname, and column options.

**Wrapper options**

The following tables list the options that apply to this data source and identify the required options that you must specify in the CREATE WRAPPER and CREATE SERVER statements.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Required. Specifies whether the wrapper runs in fenced mode or in trusted mode. Valid values are Y and N. The default is N; the wrapper runs in trusted mode.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in written in Java™, specifies a case-sensitive string for the class name that corresponds to the user mapping repository class. For example, &quot;UserMappingRepositoryLDAP&quot;. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN_LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
<tr>
<td>PROXY_SERVER_NAME</td>
<td>Specifies the name or IP address of the proxy server. This option is required if the value of PROXY_TYPE is HTTP or SOCKS. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
<tr>
<td>PROXY_SERVER_PORT</td>
<td>Specifies the port or service name for the proxy service on the proxy server. This option is required if the value of PROXY_TYPE is HTTP or SOCKS. Valid values are a decimal port number from 1 to 32760 or a service name.</td>
</tr>
<tr>
<td>PROXY_TYPE</td>
<td>Specifies the proxy type to use to access the Internet when the federated server is behind a firewall. Valid values are NONE, HTTP, and SOCKS. The default value is NONE. If you set this option to HTTP or SOCKS, you must also specify the PROXY_SERVER_NAME and PROXY_SERVER_PORT.</td>
</tr>
</tbody>
</table>
## Server options

Table 99. Server options for BioRS

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASE_SENSITIVE</td>
<td>Specifies whether the BioRS server treats names in a case-sensitive manner. Valid values are Y and N. The default is Y; names are treated 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 and the configuration parameter must specify the same case sensitivity. If you need to change the value of the CASE_SENSITIVE option after you create the server definition, you must drop the server definition and create the definition and all nicknames again.</td>
</tr>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_QUERY</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. The default is 1. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests.</td>
</tr>
<tr>
<td>DB2 UM PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in written in Java, specifies a case-sensitive string for the class name that corresponds to the user mapping repository class. For example, &quot;UserMappingRepositoryLDAP&quot;. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
<tr>
<td>DB2 UM PLUGINLANG</td>
<td>Specifies the language for the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
<tr>
<td>NODE</td>
<td>Required. Specifies the DNS host name or IP address of the system on which the BioRS query tool is available. Valid IP addresses are in IPv4 (dot-separated) format or IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured. The default value is localhost.</td>
</tr>
<tr>
<td>PORT</td>
<td>Specifies the port for connections to the BioRS server. Valid values are a numeric port or a TCP/IP service name. The default is 5014.</td>
</tr>
<tr>
<td>PROXY AUTHID</td>
<td>Specifies the user name for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY PASSWORD</td>
<td>Specifies the password for proxy server authentication.</td>
</tr>
</tbody>
</table>
### Server options for BioRS (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROXY_SERVER_NAME</td>
<td>Specifies the name or IP address of the proxy server. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
<tr>
<td>PROXY_SERVER_PORT</td>
<td>Specifies the port or service name for the proxy service on the proxy server. Valid values are a decimal port number from 1 to 32760 or a service name.</td>
</tr>
<tr>
<td>PROXY_TYPE</td>
<td>Specifies the proxy type to use to access the Internet when the federated server is behind a firewall. Valid values are NONE, HTTP, and SOCKS. The default value is NONE.</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>Specifies the maximum time, in minutes, that the federated server waits for a response from the remote server. The default is 10.</td>
</tr>
</tbody>
</table>

### User mapping options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUEST</td>
<td>Specifies that the GUEST BioRS authentication ID is used to perform operations. Valid values are Y and N. The default is N; the GUEST BioRS ID is not used. This option is not valid if you specify the REMOTE_AUTHID and REMOTE_PASSWORD options.</td>
</tr>
<tr>
<td>PROXY_AUTHID</td>
<td>Specifies the user name for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY_PASSWORD</td>
<td>Specifies the password for proxy server authentication. The password is encrypted when it is stored in the federated database catalog.</td>
</tr>
<tr>
<td>REMOTE_AUTHID</td>
<td>Specifies the remote user ID to which the local user ID is mapped. If you do not specify this option, the ID that is used to connect to the federated database is used.</td>
</tr>
<tr>
<td>REMOTE_PASSWORD</td>
<td>Specifies the remote password for the remote user ID. If you do not specify this option, the password that is used to connect to the federated database is used.</td>
</tr>
</tbody>
</table>
### Nickname options

*Table 101. Nickname options for BioRS*

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_OBJECT</td>
<td>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. The name also specifies the relationship of the nickname to other nicknames. The case sensitivity of this option depends on the case sensitivity of the BioRS server and on the value of the CASESENSITIVE server option. You cannot use the ALTER NICKNAME statement to change or delete this name. If the name of the BioRS databank changes, you must delete the nickname and then create it again.</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>Specifies the maximum time, in minutes, to wait for a response from the data source server. The default is 10.</td>
</tr>
</tbody>
</table>

### Column options

*Table 102. Column options for BioRS*

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELEMENT_NAME</td>
<td>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 CASESENSITIVE server option. You must specify the BioRS element name only if it is different from the column name.</td>
</tr>
<tr>
<td>IS_INDEXED</td>
<td>Specifies whether the corresponding column is indexed and, therefore, can be referenced in a predicate. Valid values are Y and N. The default is N; the column is not indexed.</td>
</tr>
<tr>
<td>REFERENCED_OBJECT</td>
<td>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 CASESENSITIVE server option. This option is valid only for columns that have the BioRS data type of Reference.</td>
</tr>
</tbody>
</table>

### DB2 database options reference

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, user mapping, nickname, and column options.

**Wrapper options**

The following tables list the options that apply to DB2 data sources and identify the required options that you must specify.
### Table 103. Wrapper options for DB2 data sources

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Required. Specifies whether the wrapper runs in fenced mode or in trusted mode. Valid values are Y and N. The default is N; the wrapper runs in trusted mode.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in written in Java, specifies a case-sensitive string for the class name that corresponds to the user mapping repository class. For example, &quot;UserMappingRepositoryLDAP&quot;. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN_LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
</tbody>
</table>

### Server options

### Table 104. Server options for DB2 data sources

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLLATING_SEQUENCE</td>
<td>Specifies whether the data source uses the same default collating sequence as the federated database. Valid values are Y, N, and I. I specifies a case-insensitive. The default is Y. The collating sequence specified for the federated server must match the collating sequence on the remote data source.</td>
</tr>
<tr>
<td>COMM_RATE</td>
<td>Specifies the communication rate, in megabytes per second, between the federated server and the data source server. Valid values are whole numbers the are greater than 0 and less than 2147483648. The default is 2.</td>
</tr>
<tr>
<td>CPU_RATIO</td>
<td>Specifies how much faster or slower the data source CPU is when compared to federated server CPU. Valid values are greater than 0 and less than 1x1023. The default is 1.0. Values can be expressed in any valid double notation, for example, 123E10, 123, or 1.21E4. A setting of 1 indicates that the federated server and the data source server have the same CPU speed; a 1:1 ratio. A setting of 0.5 indicates that the federated server CPU speed is 50% slower than the data source CPU. A setting of 2 indicates that the federated CPU is twice as fast as the data source CPU.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DBNAME</td>
<td>Required. Specifies the specific database to use for the initial remote DB2 database connection. This specific database is the database alias for the remote DB2 database that is cataloged on the federated server by using the CATALOG DATABASE command or the DB2 Configuration Assistant.</td>
</tr>
<tr>
<td>DB2_MAXIMAL_PUSHDOWN</td>
<td>Specifies the primary criteria that the query optimizer uses to choose an access plan. Valid values are Y and N. The default is N; the query optimizer chooses the plan that has the lowest estimated cost. Y specifies that the query optimizer choose the access plan that pushes down the most query operations to the data source.</td>
</tr>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_QUERY</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. The default is 1. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests.</td>
</tr>
<tr>
<td>DB2_TWO_PHASE_COMMIT</td>
<td>Specifies whether the federated server connects to the data source in two-phase commit protocol or one-phase commit protocol. Valid values are Y and N. The default is N; the federated server uses the one-phase commit protocol to connect. Y specifies that the federated server uses two-phase commit protocol to connect.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in written in Java, specifies a case-sensitive string for the class name that corresponds to the user mapping repository class. For example, &quot;UserMappingRepositoryLDAP&quot;. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN_LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
<tr>
<td>FED_PROXY_USER</td>
<td>Specifies the authorization ID to use to establish all outbound trusted connections when the inbound connection is non-trusted. The user whose ID is specified in this option must have a user mapping that specifies both the REMOTE_AUTHID and a REMOTE_PASSWORD options.</td>
</tr>
</tbody>
</table>

**Note:** This option is valid only for DB2 for Linux, UNIX, and Windows, Version 9.5 and for DB2 for z/OS Version 9.
Table 104. Server options for DB2 data sources (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOLD_ID</td>
<td>Specifies the case for the user ID that is sent to the data source. There is no default value; the federated server sends the user ID in uppercase; then if the uppercase user ID fails, the server sends the user ID in lowercase. Valid values are U (uppercase), L (lowercase), and N (null). Avoid using the null setting, which might result in poor performance.</td>
</tr>
<tr>
<td>FOLD_PW</td>
<td>Specifies the case for the password that is sent to the data source. There is no default value; the federated server sends the password in uppercase; then if the uppercase password fails, the server sends the password in lowercase. Valid values are U (uppercase), L (lowercase), and N (null). Avoid using the null setting, which might result in poor performance.</td>
</tr>
<tr>
<td>IO_RATIO</td>
<td>Specifies how much faster or slower the data source I/O system runs when compared to the federated server I/O system. Valid values are greater than 0 and less than 1x10^23. The default is 1.0. Values can be expressed in any valid double notation, for example, 123E10, 123, or 1.21E4. A setting of 1 indicates that the federated server and the data source server have the same I/O speed; a 1:1 ratio. A setting of 0.5 indicates that the federated server speed is 50% slower than the data source speed. A setting of 2 indicates that the federated speed is twice as fast as the data source speed.</td>
</tr>
<tr>
<td>OLD_NAME_GEN</td>
<td>Specifies how to convert the column names and index names that are in the data source into nickname column names and local index names for the federated server. Valid values are Y and N. The default is N; the generated names closely match the names in the data source. Y specifies that the generated names are the same as the names that were created in IBM WebSphere Federation Server Version 9 and earlier. Thus, the names might not closely match the data source names.</td>
</tr>
</tbody>
</table>
### Server options for DB2 data sources (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUSHDOWN</td>
<td>Specifies whether the federated server allows the data source to evaluate operations. Valid values are Y and N. The default is Y; the data source evaluates operations. N specifies that the federated server send 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 are not included in any SQL that the federated server sends to the data source.</td>
</tr>
</tbody>
</table>

### User mapping options

#### Table 105. User mapping options for DB2 data sources

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FED_PROXY_USER</td>
<td>Specifies the authorization ID to use to establish all outbound trusted connections when the inbound connection is non-trusted. The user whose ID is specified in this option must have a user mapping that specifies both a REMOTE_AUTHID and a REMOTE_PASSWORD. <strong>Important:</strong> If you specify the FED_PROXY_USER user mapping option, you must also specify the FED_PROXY_USER server option. This option is valid only for DB2 for Linux, UNIX, and Windows, Version 9.5 and for DB2 for z/OS Version 9.</td>
</tr>
<tr>
<td>ACCOUNTING_STRING</td>
<td>Required if accounting information must be passed. Specifies a DRDA accounting string. Valid values include any string that has 255 characters or fewer.</td>
</tr>
<tr>
<td>REMOTE_AUTHID</td>
<td>Specifies the remote user ID to which the local user ID is mapped. If you do not specify this option, the ID that is used to connect to the federated database is used.</td>
</tr>
<tr>
<td>REMOTE_PASSWORD</td>
<td>Specifies the remote password for the remote user ID. If you do not specify this option, the password that is used to connect to the federated database is used.</td>
</tr>
</tbody>
</table>
Table 105. User mapping options for DB2 data sources (continued)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE_TRUSTEDCONTEXT</td>
<td>Specifies whether the user mapping is trusted. Valid values are Y and N. The default is N; the user mapping is not trusted and can be used only in non-trusted federated outbound connections. Y specifies that the user mapping is trusted and can be used in both trusted and non-trusted outbound federated connections. Note: This option is valid only for DB2 for Linux, UNIX, and Windows, Version 9.5 and for DB2 for z/OS Version 9.</td>
</tr>
</tbody>
</table>

Column options

Table 106. Column options for DB2 data sources

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMERIC_STRING</td>
<td>Specifies how to treat numeric strings. The default is N. If the data source string column contains only numeric strings and no other characters, including blanks, set the NUMERIC_STRING option to Y. When NUMERIC_STRING is set to Y for a column, the query optimizer recognizes that the column contains no blanks that could interfere with the sorting of the data in the column. 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.</td>
</tr>
<tr>
<td>XML_ROOT</td>
<td>Specifies the XML root element to add to the values of an XML column that references an XML sequence. This option ensures that the values of the XML column is a well-formed XML document.</td>
</tr>
</tbody>
</table>

Entrez options reference

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, user mapping, nickname, and column options.

Wrapper options

The following tables list the options that apply to this data source and identify the required options that you must specify.

Table 107. Wrapper options for Entrez

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Required. Specifies whether the wrapper runs in fenced mode or in trusted mode. Valid values are Y and N. The default is N; the wrapper runs in trusted mode.</td>
</tr>
</tbody>
</table>
Table 107. Wrapper options for Entrez (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMAIL</td>
<td>Required. Specifies the email address to include with all queries. NCBI sends mail to this address if there are problems.</td>
</tr>
<tr>
<td>PROXY_TYPE</td>
<td>Specifies the proxy type to use to access the Internet when the federated server is behind a firewall. Valid values are NONE, HTTP, and SOCKS. The default is NONE.</td>
</tr>
<tr>
<td>PROXY_SERVER_NAME</td>
<td>Specifies the proxy server name or the IP address. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 only if IPv6 is configured.</td>
</tr>
<tr>
<td>PROXY_SERVER_PORT</td>
<td>Specifies the port or service name for the proxy service on the proxy server.</td>
</tr>
</tbody>
</table>

Server options

Table 108. Server options for Entrez

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_QUERY</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. The default is 1. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests.</td>
</tr>
<tr>
<td>MAX_ROWS</td>
<td>Specifies the maximum number of rows that a query returns. Valid values are any number that is greater than or equal to zero. 0 specifies that there is no limit. Setting MAX_ROWS to 0 or a high number affects query performance. For OMIM, this options limits the number of records that are returned from the root nickname. For example, if MAX_ROWS is set to 25, a maximum of 25 records for the root nickname and all of the records for the child nicknames are returned. The default for Microsoft Windows systems is 2000 rows. The default for UNIX systems is 5000 rows.</td>
</tr>
<tr>
<td>PROXY_AUTHID</td>
<td>Specifies the user name for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY_PASSWORD</td>
<td>Specifies the password for proxy server authentication. The password is encrypted when it is stored in the federated database catalog.</td>
</tr>
<tr>
<td>PROXY_SERVER_NAME</td>
<td>Specifies the name or IP address of the proxy server. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 only if IPv6 is configured.</td>
</tr>
</tbody>
</table>
Table 108. Server options for Entrez (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROXY_SERVER_PORT</td>
<td>Specifies the port or service name for the proxy service on the proxy server. Valid values are a decimal port number from 1 to 32760 or a service name.</td>
</tr>
<tr>
<td>PROXY_TYPE</td>
<td>Specifies the proxy type to use to access the Internet when the federated server is behind a firewall. Valid values are NONE, HTTP, and SOCKS. The default is NONE.</td>
</tr>
<tr>
<td>SOCKET_TIMEOUT</td>
<td>Specifies the maximum time, in minutes, that the federated server waits for results from the proxy server. Valid values are numbers greater than or equal to 0. The default is 0; the federated server waits an unlimited amount of time.</td>
</tr>
</tbody>
</table>

User mapping options

Table 109. User mapping options for Entrez

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROXY_AUTHID</td>
<td>Specifies the user name for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY_PASSWORD</td>
<td>Specifies the password for proxy server authentication. The password is encrypted when it is stored in the federated database catalog.</td>
</tr>
</tbody>
</table>

Nickname options

Table 110. Nickname options for Entrez

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_OBJECT</td>
<td>Specifies the name of the Entrez object type that is associated with the nickname. The option determines the schema and NCBI database for the nickname and its relationship to other nicknames. The value is case-sensitive.</td>
</tr>
<tr>
<td>PARENT</td>
<td>Specified only for a child nickname whose parent is renamed by the REMOTE_OBJECT option. The PARENT option associates a child with a parent when multiple nickname families are defined within a database schema. The value is case-sensitive.</td>
</tr>
</tbody>
</table>

Excel options reference

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, and nickname options.
Wrapper options

The following tables list the options that apply to this data source and identify the required options that you must specify.

Table 111. Wrapper options for Excel

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Required. Specifies whether the wrapper runs in fenced mode or in trusted mode. Valid values are Y and N. The default is N; the wrapper runs in trusted mode.</td>
</tr>
</tbody>
</table>

Server options

Table 112. Server options for Excel

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_QUERY</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. The default is 1. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests.</td>
</tr>
</tbody>
</table>

Nickname options

Table 113. Nickname options for Excel

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE_PATH</td>
<td>Required. Specifies the fully qualified directory path and file name of the Excel spreadsheet that you want to access.</td>
</tr>
<tr>
<td>RANGE</td>
<td>Specifies the range of cells to use, for example, A1:C100. The value before the colon specifies the top left cell of the range. The value after the colon specifies the bottom right cell of the range.</td>
</tr>
</tbody>
</table>

HMMER options reference

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, user mapping, and nickname options.

Wrapper options

The following tables list the options that apply to this data source and identify the required options that you must specify.
### Wrapper options for HMMER

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Specifies whether the wrapper runs in fenced mode or in trusted mode. Valid values are Y and N. The default is N; the wrapper runs in trusted mode.</td>
</tr>
<tr>
<td>PROXY_TYPE</td>
<td>Specifies the proxy type to use to access the Internet when the federated server is behind a firewall. Valid values are NONE and SOCKS. The default value is NONE.</td>
</tr>
<tr>
<td>PROXY_SERVER_NAME</td>
<td>Specifies the name or IP address of the proxy server. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
<tr>
<td>PROXY_SERVER_PORT</td>
<td>Specifies the name or IP address of the proxy server. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
</tbody>
</table>

### Server options

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAEMON_PORT</td>
<td>Required. Specifies the port number on which the HMMER daemon listens for HMMER job requests. The default is 4098. The port number must be the same as the DAEMON_PORT option in the daemon configuration file. If a service name is configured for the HMMER daemon, use a TCP/IP service name.</td>
</tr>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_QUERY</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests. The default is 1.</td>
</tr>
<tr>
<td>HMMPFAM_OPTIONS</td>
<td>Specifies hmmpfam options 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 and the additional options -xnu --pvm. This option is valid only with type PFAM.</td>
</tr>
</tbody>
</table>
Table 115. Server options for HMMER (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMMSEARCH_OPTIONS</td>
<td>Allows the user to provide additional command line options to the hmmsearch command. This option is valid only with type SEARCH.</td>
</tr>
<tr>
<td>NODE</td>
<td>Required. Specifies the DNS host name or IP address of the system on which the HMMER daemon runs. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
<tr>
<td>PROCESSORS</td>
<td>Specifies the number of processors for the HMMER program to use. If this option is not specified, the default is to use all of the CPUs in the system. This option is equivalent to the –cpu option of the hmmpfam or hmmsearch command.</td>
</tr>
<tr>
<td>PROXY_AUTHID</td>
<td>Specifies the user name for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY_PASSWORD</td>
<td>Specifies the password for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY_SERVER_NAME</td>
<td>Specifies the name or IP address of the proxy server. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
<tr>
<td>PROXY_SERVER_PORT</td>
<td>Specifies the port or service name for the proxy service on the proxy server. Valid values are a decimal port number from 1 to 32760 or a service name.</td>
</tr>
<tr>
<td>PROXY_TYPE</td>
<td>Specifies the proxy type to use to access the Internet when the federated server is behind a firewall. Valid values are NONE and SOCKS. The default value is NONE.</td>
</tr>
</tbody>
</table>

User mapping options

Table 116. User mapping options for HMMER

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROXY_AUTHID</td>
<td>Specifies the user name for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY_PASSWORD</td>
<td>Specifies the password for proxy server authentication. The password is encrypted when it is stored in the federated database catalog.</td>
</tr>
</tbody>
</table>
**Nickname options**

*Table 117. Nickname options for HMMER*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATASOURCE</td>
<td>Required. For type PFAM, this option specifies the name of the HMM Profile</td>
</tr>
<tr>
<td></td>
<td>database that HMMPFAM will search. This value must be the same as the one</td>
</tr>
<tr>
<td></td>
<td>used in the configuration file of the HMMER daemon. For type SEARCH, this</td>
</tr>
<tr>
<td></td>
<td>option specifies the name of the sequence file that HMMSEARCH will search.</td>
</tr>
<tr>
<td></td>
<td>This value must be the same as the one used in the configuration file of the</td>
</tr>
<tr>
<td></td>
<td>HMMER daemon.</td>
</tr>
<tr>
<td>HMMTYPE</td>
<td>Specifies the alphabet that is used in both models and gene sequences.</td>
</tr>
<tr>
<td></td>
<td>Valid values are PROTEIN and NUCLEIC. The default is PROTEIN. NUCLEIC is</td>
</tr>
<tr>
<td></td>
<td>equivalent to the -n option of the hmmpfam command.</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>Required. Specifies the maximum time, in minutes, to wait for a response</td>
</tr>
<tr>
<td></td>
<td>from the data source server. The default is 60 minutes.</td>
</tr>
</tbody>
</table>

**Informix options reference**

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, user mapping, and column options.

**Wrapper options**

The following tables list the options that apply to this data source and identify the required options that you must specify.

*Table 118. Wrapper options for Informix*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Required. Specifies whether the wrapper runs in fenced mode or in trusted</td>
</tr>
<tr>
<td></td>
<td>mode. Valid values are Y and N. The default is N; the wrapper runs in</td>
</tr>
<tr>
<td></td>
<td>trusted mode.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in</td>
</tr>
<tr>
<td></td>
<td>written in Java, specifies a case-sensitive string for the class name that</td>
</tr>
<tr>
<td></td>
<td>corresponds to the user mapping repository class. For example, &quot;UserMappingRepositoryLDAP&quot;. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN_LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java</td>
</tr>
<tr>
<td></td>
<td>and C. The default is Java.</td>
</tr>
</tbody>
</table>
### Server options

*Table 119. Server options for Informix*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLLATING_SEQUENCE</td>
<td>Specifies whether the data source uses the same default collating sequence as the federated database. Valid values are Y, N, and I. I specifies a case-insensitive. The default is Y. The collating sequence specified for the federated server must match the collating sequence on the remote data source.</td>
</tr>
<tr>
<td>COMM_RATE</td>
<td>Specifies the communication rate, in megabytes per second, between the federated server and the data source server. Valid values are whole numbers that are greater than 0 and less than 2147483648. The default is 2.</td>
</tr>
<tr>
<td>CPU_RATIO</td>
<td>Specifies how much faster or slower the data source CPU is when compared to federated server CPU. Valid values are greater than 0 and less than 1E7. The default is 1.0. Values can be expressed in any valid double notation, for example, 123E10, 123, or 1.21E4. A setting of 1 indicates that the federated server and the data source server have the same CPU speed; a 1:1 ratio. A setting of 0.5 indicates that the federated server CPU speed is 50% slower than the data source CPU. A setting of 2 indicates that the federated CPU is twice as fast as the data source CPU.</td>
</tr>
<tr>
<td>DBNAME</td>
<td>Required. Specifies the name of the Informix database that you want to access.</td>
</tr>
<tr>
<td>DB2_MAXIMAL_PUSHDOWN</td>
<td>Specifies the primary criteria that the query optimizer uses to choose an access plan. Valid values are Y and N. The default is N; the query optimizer chooses the plan that has the lowest estimated cost. Y specifies that the query optimizer choose the access plan that pushes down the most query operations to the data source. If more than one access plan meets this criteria, plan that has the lowest cost is chosen.</td>
</tr>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_QUERY</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. The default is 1. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DB2_TWO_PHASE_COMMIT</td>
<td>Specifies whether the federated server connects to the data source in two-phase commit protocol or one-phase commit protocol. Valid values are Y and N. The default is N; the federated server uses the one-phase commit protocol to connect. Y specifies that the federated server uses two-phase commit protocol to connect.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in written in Java, specifies a case-sensitive string for the class name that corresponds to the user mapping repository class. For example, &quot;UserMappingRepositoryLDAP&quot;. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN_LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
<tr>
<td>FOLD_ID</td>
<td>Specifies the case for the user ID that is sent to the data source. There is no default value; the federated server sends the user ID in uppercase; then if the uppercase user ID fails, the server sends the user ID in lowercase. Valid values are U (uppercase), L (lowercase), and N (null). Avoid using the null setting, which might result in poor performance.</td>
</tr>
<tr>
<td>FOLD_PW</td>
<td>Specifies the case for the password that is sent to the data source. There is no default value; the federated server sends the password in uppercase; then if the uppercase password fails, the server sends the password in lowercase. Valid values are U (uppercase), L (lowercase), and N (null). Avoid using the null setting, which might result in poor performance.</td>
</tr>
<tr>
<td>INFORMIX_CLIENT_LOCALE</td>
<td>Specifies the CLIENT_LOCALE to use for the connection between the federated server and the data source server. The value is any valid Informix locale. If you do not specify this option, the CLIENT_LOCALE environment variable is set to the value that is specified in the db2dj.ini file. If the db2dj.ini does not specify the CLIENT_LOCALE environment variable, the INFORMIX_CLIENT_LOCALE is set to the Informix locale that most closely matches the code page and territory of the federated database.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INFORMIX_DB_LOCALE</td>
<td>Specifies the Informix DB_LOCALE to use for the connection between the federated server and the data source server. If the INFORMIX_DB_LOCALE option is not specified, the Informix DB_LOCALE environment variable is set to the value specified in the db2dj.ini file. If the db2dj.ini file does not specify a value, the Informix DB_LOCALE environment variable is not set.</td>
</tr>
<tr>
<td>INFORMIX_LOCK_MODE</td>
<td>Specifies the lock mode to set for an Informix data source. The Informix wrapper issues the SET LOCK MODE command immediately after connecting to an Informix data source. Valid values are W, N, and a number. The default is W; the wrapper waits an unlimited amount of time for the lock to be released. N specifies not to wait; an error is returned immediately. Use a number to specify the maximum amount of time, in seconds, to wait. If a deadlock or timeout occurs, use the ALTER SERVER statement to change the value of the INFORMIX_LOCK_MODE option. For example: ALTER SERVER TYPE informix VERSION 9 WRAPPER informix OPTIONS (ADD informix_lock_mode '60')</td>
</tr>
<tr>
<td>IO_RATIO</td>
<td>Specifies how much faster or slower the data source I/O system runs when compared to the federated server I/O system. Valid values are greater than 0 and less than 1x10^23. The default is 1.0. Values can be expressed in any valid double notation, for example, 123E10, 123, or 1.21E4. A setting of 1 indicates that the federated server and the data source server have the same I/O speed; a 1:1 ratio. A setting of 0.5 indicates that the federated server speed is 50% slower than the data source speed. A setting of 2 indicates that the federated speed is twice as fast as the data source speed.</td>
</tr>
<tr>
<td>IUD_APP_SVPT_ENFORCE</td>
<td>Specifies whether the federated server enforces the use of application savepoint statements. Valid values are Y and N. The default is Y; if the data source does not enforce application savepoint statements and an error occurs during an insert, update, or delete operation, the federated server rolls back the transaction and SQL error code SQL1476N is returned. It is recommended that you use the default setting.</td>
</tr>
</tbody>
</table>
Table 119. Server options for Informix (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NODE</td>
<td>Required. Specifies the name by which the data source is defined as an instance to its relational database management system.</td>
</tr>
<tr>
<td>OLD_NAME_GEN</td>
<td>Specifies how to convert the column names and index names that are in the data source into nickname column names and local index names for the federated server. Valid values are Y and N. The default is N; the generated names closely match the names in the data source. Y specifies that the generated names are the same as the names that were created in IBM WebSphere Federation Server Version 9 and earlier. Thus, the names might not closely match the data source names.</td>
</tr>
<tr>
<td>PUSHDOWN</td>
<td>Specifies whether the federated server allows the data source to evaluate operations. Valid values are Y and N. The default is Y; the data source evaluates operations. N specifies that the federated server send 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 are not included in any SQL that the federated server sends to the data source.</td>
</tr>
</tbody>
</table>

User mapping options

Table 120. User mapping options for Informix

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_AUTHID</td>
<td>Specifies the remote user ID to which the local user ID is mapped. If you do not specify this option, the ID that is used to connect to the federated database is used.</td>
</tr>
<tr>
<td>REMOTE_PASSWORD</td>
<td>Specifies the remote password for the remote user ID. If you do not specify this option, the password that is used to connect to the federated database is used.</td>
</tr>
</tbody>
</table>
Column options

Table 121. Column options for Informix

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMERIC_STRING</td>
<td>Specifies how to treat numeric strings. The default is N. If the data source string column contains only numeric strings and no other characters, including blanks, set the NUMERIC_STRING option to Y. When NUMERIC_STRING is set to Y for a column, the query optimizer recognizes that the column contains no blanks that could interfere with the sorting of the data in the column. 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.</td>
</tr>
</tbody>
</table>

Microsoft SQL Server options reference

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, user mapping, and column options.

Wrapper options

The following tables list the options that apply to this data source and identify the required options that you must specify.

Table 122. Wrapper options for Microsoft SQL Server

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Required. Specifies whether the wrapper runs in fenced mode or in trusted mode. Valid values are Y and N. The default is N; the wrapper runs in trusted mode.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in written in Java, specifies a case-sensitive string for the class name that corresponds to the user mapping repository class. For example, &quot;UserMappingRepositoryLDAP&quot;. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN_LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
</tbody>
</table>
### Server options

Table 123. Server options for Microsoft SQL Server

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODEPAGE</td>
<td>Specifies the code page that corresponds to the coded character set of the data source client configuration. On UNIX and Microsoft Windows systems that use a non-Unicode federated database, the default is the code page of the federated database. On UNIX systems that use a Unicode federated database, the default is 1208. On Windows systems that use a Unicode federated database, the default is 1202.</td>
</tr>
<tr>
<td>COLLATING_SEQUENCE</td>
<td>Specifies whether the data source uses the same default collating sequence as the federated database. Valid values are Y, N, and I. I specifies a case-insensitive. The default is Y. The collating sequence specified for the federated server must match the collating sequence on the remote data source.</td>
</tr>
<tr>
<td>COMM_RATE</td>
<td>Specifies how much faster or slower the data source CPU is when compared to federated server CPU. Valid values are greater than 0 and less than 1x10^3. The default is 1.0. Values can be expressed in any valid double notation, for example, 123E10, 123, or 1.21E4. A setting of 1 indicates that the federated server and the data source have the same CPU speed; a 1:1 ratio. A setting of 0.5 indicates that the federated server CPU speed is 50% slower than the data source CPU. A setting of 2 indicates that the federated CPU is twice as fast as the data source CPU.</td>
</tr>
<tr>
<td>CPU_RATIO</td>
<td>Specifies how much faster or slower the CPU of the data source runs when compared to the speed of the federated server’s CPU. Valid values are greater than 0 and less than 1x10^3. The default is 1.0. Values can be expressed in any valid double notation, for example, 123E10, 123, or 1.21E4.</td>
</tr>
<tr>
<td>DBNAME</td>
<td>Required. Specifies the alias for the database that you want to access. The value is case-sensitive.</td>
</tr>
<tr>
<td>DB2_MAXIMAL_PUSHDOWN</td>
<td>Specifies the primary criteria that the query optimizer uses to choose an access plan. Valid values are Y and N. The default is N; the query optimizer chooses the plan that has the lowest estimated cost. Y specifies that the query optimizer choose the access plan that pushes down the most query operations to the data source. If more than one access plan meets this criteria, plan that has the lowest cost is chosen.</td>
</tr>
</tbody>
</table>
Table 123. Server options for Microsoft SQL Server (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_QUERY</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. The default is 1. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>DB2_TWOPHASE_COMMIT</td>
<td>Specifies whether the federated server connects to the data source in two-phase commit protocol or one-phase commit protocol. Valid values are Y and N. The default is N; the federated server uses the one-phase commit protocol to connect. Y specifies that the federated server uses two-phase commit protocol to connect. <strong>Important:</strong> If you set this option to Y, you must also specify the XA_OPEN_STRING_OPTION.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>DB2_UM_PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in written in Java, specifies a case-sensitive string for the class name that corresponds to the user mapping repository class. For example, &quot;UserMappingRepositoryLDAP&quot;. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>DB2_UM_PLUGIN_LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>FOLD_ID</td>
<td>Specifies the case for the user ID that is sent to the data source. There is no default value; the federated server sends the user ID in uppercase; then if the uppercase user ID fails, the server sends the user ID in lowercase. Valid values are U (uppercase), L (lowercase), and N (null). Avoid using the null setting, which might result in poor performance.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>FOLD_PW</td>
<td>Specifies the case for the password that is sent to the data source. There is no default value; the federated server sends the password in uppercase; then if the uppercase password fails, the server sends the password in lowercase. Valid values are U (uppercase), L (lowercase), and N (null). Avoid using the null setting, which might result in poor performance.</td>
</tr>
</tbody>
</table>

Chapter 22. Data source options reference 413
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO_RATIO</td>
<td>Specifies how much faster or slower the data source I/O system runs when compared to the federated server I/O system. Valid values are greater than 0 and less than $1 \times 10^{23}$. The default is 1.0. Values can be expressed in any valid double notation, for example, 123E10, 123, or 1.21E4. A setting of 1 indicates that the federated server and the data source server have the same I/O speed; a 1:1 ratio. A setting of 0.5 indicates that the federated server speed is 50% slower than the data source speed. A setting of 2 indicates that the federated speed is twice as fast as the data source speed.</td>
</tr>
<tr>
<td>NODE</td>
<td>Required. If the federated server uses Microsoft Windows, the value for NODE is the system DSN name that you specify for the Microsoft SQL Server. If the federated server uses UNIX or Linux, the value for NODE is defined in the odbc.ini file. The value is case-sensitive.</td>
</tr>
<tr>
<td>OLD_NAME_GEN</td>
<td>Specifies how to convert the column names and index names that are in the data source into nickname column names and local index names for the federated server. Valid values are Y and N. The default is N; the generated names closely match the names in the data source. Y specifies that the generated names are the same as the names that were created in IBM WebSphere Federation Server Version 9 and earlier. Thus, the names might not closely match the data source names.</td>
</tr>
<tr>
<td>PASSWORD</td>
<td>Specifies whether or not passwords are sent to a data source. The default is Y.</td>
</tr>
<tr>
<td>PUSHDOWN</td>
<td>Specifies whether the federated server allows the data source to evaluate operations. Valid values are Y and N. The default is Y; the data source evaluates operations. N specifies that the federated server send 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 are not included in any SQL that the federated server sends to the data source.</td>
</tr>
<tr>
<td>XA_OPEN_STRING_OPTIONS</td>
<td>Required when DB2_TWO_PHASE_COMMIT is set to Y. Specifies the resource manager ID of the Microsoft SQL Server Registry.</td>
</tr>
</tbody>
</table>
User mapping options

Table 124. User mapping options for Microsoft SQL Server

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_AUTHID</td>
<td>Specifies the remote user ID to which the local user ID is mapped. If you do not specify this option, the ID that is used to connect to the federated database is used.</td>
</tr>
<tr>
<td>REMOTE_PASSWORD</td>
<td>Specifies the remote password for the remote user ID. If you do not specify this option, the password that is used to connect to the federated database is used.</td>
</tr>
</tbody>
</table>

Column options

Table 125. Column options for Microsoft SQL Server

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMERIC_STRING</td>
<td>Specifies how to treat numeric strings. The default is N. If the data source string column contains only numeric strings and no other characters, including blanks, set the NUMERIC_STRING option to Y. When NUMERIC_STRING is set to Y for a column, the query optimizer recognizes that the column contains no blanks that could interfere with the sorting of the data in the column. 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.</td>
</tr>
</tbody>
</table>

ODBC options reference

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, user mapping, and column options.

Wrapper options

The following tables list the options that apply to this data source and identify the required options that you must specify.

Table 126. Wrapper options for ODBC

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Required. Specifies whether the wrapper runs in fenced mode or in trusted mode. Valid values are Y and N. The default is N; the wrapper runs in trusted mode. Important: If you set this option to Y for a UNIX system, you must also set the DB2_SOURCECLIENT_MODE wrapper option.</td>
</tr>
</tbody>
</table>
### Table 126. Wrapper options for ODBC (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_SOURCE_CLIENT_MODE</td>
<td>Specifies that the client for the data source is 32-bit and that the database instance on the federated server is 64-bit. The only valid value is 32-bit. This option is valid only for UNIX. <strong>Important:</strong> If you set this option, you must also set the DB2_FENCED wrapper option to Y.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in written in Java, specifies a case-sensitive string for the class name that corresponds to the user mapping repository class. For example, “UserMappingRepositoryLDAP”. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN_LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
<tr>
<td>MODULE</td>
<td>Required for federated servers that run on a UNIX system. Specifies the full path of the library that contains the ODBC Driver Manager implementation or the SQL/CLI implementation. There is no default for UNIX. On a Microsoft Windows system, the default is odbc32.dll.</td>
</tr>
</tbody>
</table>

### Server options

**Table 127. Server options for ODBC**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODEPAGE</td>
<td>Specifies the code page that corresponds to the coded character set of the client configuration for the data source. On UNIX and Windows systems that use a non-Unicode federated database, the default is the code page that the federated database uses. On UNIX systems that use a Unicode federated database, the default is 1208. On Windows systems that use a Unicode federated database, the default is 1202.</td>
</tr>
<tr>
<td>COLLATING_SEQUENCE</td>
<td>Specifies whether the data source uses the same default collating sequence as the federated database. Valid values are Y, N, and I. I specifies a case-insensitive. The default is Y. The collating sequence specified for the federated server must match the collating sequence on the remote data source.</td>
</tr>
</tbody>
</table>
Table 127. Server options for ODBC (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMM_RATE</td>
<td>Specifies the communication rate, in megabytes per second, between the federated server and the data source server. Valid values are whole numbers the are greater than 0 and less than 2147483648. The default is 2.</td>
</tr>
<tr>
<td>CPU_RATIO</td>
<td>Specifies how much faster or slower the data source CPU is when compared to federated server CPU. Valid values are greater than 0 and less than 1x10^23. The default is 1.0. Values can be expressed in any valid double notation, for example, 123E10, 123, or 1.21E4. A setting of 1 indicates that the federated server and the data source server have the same CPU speed; a 1:1 ratio. A setting of 0.5 indicates that the federated server CPU speed is 50% slower than the data source CPU. A setting of 2 indicates that the federated CPU is twice as fast as the data source CPU.</td>
</tr>
<tr>
<td>DATEFORMAT</td>
<td>Specifies the date format that the data source uses. Use ‘DD’, ‘MM’, and ‘YY’ or ‘YYYY’ to specify the date format. You can specify a delimiter such as a space, a hyphen, or a comma. For example, the format ‘YYYY-MM-DD’ specifies a date such as 1958-10-01. The value can contain null values.</td>
</tr>
<tr>
<td>DBNAME</td>
<td>Specifies the name of the data source database that you want to access.</td>
</tr>
<tr>
<td>DB2_MAXIMAL_PUSHDOWN</td>
<td>Specifies the primary criteria that the query optimizer uses to choose an access plan. Valid values are Y and N. The default is N; the query optimizer chooses the plan that has the lowest estimated cost. Y specifies that the query optimizer choose the access plan that pushes down the most query operations to the data source.</td>
</tr>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_QUERY</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. The default is 0. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in written in Java, specifies a case-sensitive string for the class name that corresponds to the user mapping repository class. For example, “UserMappingRepositoryLDAP”. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN_LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
<tr>
<td>FOLD_ID</td>
<td>Specifies the case for the user ID that is sent to the data source. There is no default value; the federated server sends the user ID in uppercase; then if the uppercase user ID fails, the server sends the user ID in lowercase. Valid values are U (uppercase), L (lowercase), and N (null). Avoid using the null setting, which might result in poor performance.</td>
</tr>
<tr>
<td>FOLD_PW</td>
<td>Specifies the case for the password that is sent to the data source. There is no default value; the federated server sends the password in uppercase; then if the uppercase password fails, the server sends the password in lowercase. Valid values are U (uppercase), L (lowercase), and N (null). Avoid using the null setting, which might result in poor performance.</td>
</tr>
<tr>
<td>IO_RATIO</td>
<td>Specifies how much faster or slower the data source I/O system runs when compared to the federated server I/O system. Valid values are greater than 0 and less than $1 \times 10^{23}$. The default is 1.0. Values can be expressed in any valid double notation, for example, 123E10, 123, or 1.21E4. A setting of 1 indicates that the federated server and the data source server have the same I/O speed; a 1:1 ratio. A setting of 0.5 indicates that the federated server speed is 50% slower than the data source speed. A setting of 2 indicates that the federated speed is twice as fast as the data source speed.</td>
</tr>
<tr>
<td>NODE</td>
<td>Required. Specifies the name of the node or the system DSN name that is assigned to the ODBC data source when the DSN is defined. The value is case-sensitive.</td>
</tr>
<tr>
<td>OLD_NAME_GEN</td>
<td>Specifies how to convert the column names and index names that are in the data source into nickname column names and local index names for the federated server. Valid values are Y and N. The default is N; the generated names closely match the names in the data source. Y specifies that the generated names are the same as the names that were created in IBM WebSphere Federation Server Version 9 and earlier. Thus, the names might not closely match the data source names.</td>
</tr>
</tbody>
</table>
Table 127. Server options for ODBC  (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUSHDOWN</td>
<td>Specifies whether the federated server allows the data source to evaluate operations. Valid values are Y and N. The default is Y; the data source evaluates operations. N specifies that the federated server send 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 are not included in any SQL that the federated server sends to the data source.</td>
</tr>
<tr>
<td>TIMEFORMAT</td>
<td>Specifies the time format that the data source uses. Use ‘hh12’, ‘hh24’, ‘mm’, ‘ss’, ‘AM’, and ‘A.M.’ to specify the time format. For example, the format ‘hh24:mm:22’ specifies a time such as 16:00:00. The format ‘hh12:mm:ss AM’ specifies a time such as 8:00:00 AM. The value can contain null values.</td>
</tr>
<tr>
<td>TIMESTAMPFORMAT</td>
<td>Specifies the timestamp format that the data source uses. Valid values are in the format that the DATEFORMAT option and the TIMEFORMAT option use. Specify ‘n’ for a tenth of a second, ‘nn’ for a hundreth of a second, ‘nnn’ for milliseconds, and so on, up to ‘nnnnnn’ for microseconds. For example, the format ‘YY-MM-DD-hh24:mm:ss.nnnnnn’ specifies a timestamp such as 1994-01-01-24:00:00.000000. The value can contain null values.</td>
</tr>
<tr>
<td>VARCHAR_NO_TRAILING_BLANKS</td>
<td>Specifies whether the data source contains VARCHAR columns that contain at least one trailing blank character. The default is N; VARCHAR columns contain at least one trailing blank character.</td>
</tr>
</tbody>
</table>

User mapping options

Table 128. User mapping options for ODBC

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_AUTHID</td>
<td>Specifies the remote user ID to which the local user ID is mapped. If you do not specify this option, the ID that is used to connect to the federated database is used.</td>
</tr>
<tr>
<td>REMOTE_PASSWORD</td>
<td>Specifies the remote password for the remote user ID. If you do not specify this option, the password that is used to connect to the federated database is used.</td>
</tr>
</tbody>
</table>
Column options

Table 129. Column options for ODBC

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMERIC_STRING</td>
<td>Specifies whether the column contains strings of numeric characters that include blanks. Valid values are Y and N. The default is N; the column does not contain numeric strings that include blanks. If the column contains only numeric strings followed by trailing blanks, do not specify Y. When NUMERIC_STRING is set to Y for a column, the query optimizer recognizes that the column contains no blanks that could interfere with the sorting of the data in the column. 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.</td>
</tr>
<tr>
<td>VARCHAR_NO_TRAILING_BLANKS</td>
<td>Specifies whether there is at least one trailing blank in the VARCHAR column.</td>
</tr>
</tbody>
</table>

Oracle options reference

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, user mapping, and column options.

Wrapper options

The following tables list the options that apply to this data source and identify the required options that you must specify.

Table 130. Wrapper options for Oracle

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Required. Specifies whether the wrapper runs in fenced mode or in trusted mode. Valid values are Y and N. The default is N; the wrapper runs in trusted mode.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in written in Java, specifies a case-sensitive string for the class name that corresponds to the user mapping repository class. For example, &quot;UserMappingRepositoryLDAP&quot;. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN_LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
</tbody>
</table>
## Server options

*Table 131. Server options for Oracle*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLLATING_SEQUENCE</td>
<td>Specifies whether the data source uses the same default collating sequence as the federated database. Valid values are Y, N, and I. I specifies a case-insensitive. The default is Y. The collating sequence specified for the federated server must match the collating sequence on the remote data source.</td>
</tr>
<tr>
<td>COMM_RATE</td>
<td>Specifies the communication rate, in megabytes per second, between the federated server and the data source server. Valid values are whole numbers that are greater than 0 and less than 2147483648. The default is 2.</td>
</tr>
<tr>
<td>CPU_RATIO</td>
<td>Specifies how much faster or slower the data source CPU is when compared to federated server CPU. Valid values are greater than 0 and less than 1x10^{23}. The default is 1.0. Values can be expressed in any valid double notation, for example, 123E10, 123, or 1.21E4. A setting of 1 indicates that the federated server and the data source server have the same CPU speed; a 1:1 ratio. A setting of 0.5 indicates that the federated server CPU speed is 50% slower than the data source CPU. A setting of 2 indicates that the federated CPU is twice as fast as the data source CPU.</td>
</tr>
<tr>
<td>DB2_MAXIMAL_PUSHDOWN</td>
<td>Specifies the primary criteria that the query optimizer uses to choose an access plan. Valid values are Y and N. The default is N; the query optimizer chooses the plan that has the lowest estimated cost. Y specifies that the query optimizer choose the access plan that pushes down the most query operations to the data source.</td>
</tr>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_QUERY</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. The default is 1. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests.</td>
</tr>
<tr>
<td>DB2_TWO_PHASE_COMMIT</td>
<td>Specifies whether the federated server connects to the data source in two-phase commit protocol or one-phase commit protocol. Valid values are Y and N. The default is N; the federated server uses the one-phase commit protocol to connect. Y specifies that the federated server uses two-phase commit protocol to connect.</td>
</tr>
</tbody>
</table>
Table 131. Server options for Oracle (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_UM_PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in written in Java, specifies a case-sensitive string for the class name that corresponds to the user mapping repository class. For example, &quot;UserMappingRepositoryLDAP&quot;. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN_LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
<tr>
<td>FED_PROXY_USER</td>
<td>Specifies the authorization ID to use to establish all outbound trusted connections when the inbound connection is non-trusted. <strong>Important:</strong> The user whose ID is specified in this option must have a user mapping that specifies both a REMOTE_AUTHID and a REMOTE_PASSWORD.</td>
</tr>
<tr>
<td>FOLD_ID</td>
<td>Specifies the case for the user ID that is sent to the data source. There is no default value; the federated server sends the user ID in uppercase; then if the uppercase user ID fails, the server sends the user ID in lowercase. Valid values are U (uppercase), L (lowercase), and N (null). Avoid using the null setting, which might result in poor performance.</td>
</tr>
<tr>
<td>FOLD_PW</td>
<td>Specifies the case for the password that is sent to the data source. There is no default value; the federated server sends the password in uppercase; then if the uppercase password fails, the server sends the password in lowercase. Valid values are U (uppercase), L (lowercase), and N (null). Avoid using the null setting, which might result in poor performance.</td>
</tr>
<tr>
<td>IO_RATIO</td>
<td>Specifies how much faster or slower the data source I/O system runs when compared to the federated server I/O system. Valid values are greater than 0 and less than 1x10^23. The default is 1.0. Values can be expressed in any valid double notation, for example, 123E10, 123, or 1.21E4. A setting of 1 indicates that the federated server and the data source server have the same I/O speed; a 1:1 ratio. A setting of 0.5 indicates that the federated server speed is 50% slower than the data source speed. A setting of 2 indicates that the federated speed is twice as fast as the data source speed.</td>
</tr>
<tr>
<td>NODE</td>
<td>Required. Specifies the name of the node where the Oracle database server resides. Obtain the name of the node from the tnsnames.ora file.</td>
</tr>
</tbody>
</table>
Table 131. Server options for Oracle (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLD_NAME_GEN</td>
<td>Specifies how to convert the column names and index names that are in the data source into nickname column names and local index names for the federated server. Valid values are Y and N. The default is N; the generated names closely match the names in the data source. Y specifies that the generated names are the same as the names that were created in IBM WebSphere Federation Server Version 9 and earlier. Thus, the names might not closely match the data source names.</td>
</tr>
<tr>
<td>PLAN_HINTS</td>
<td>Specifies whether or not plan hints are to be enabled. Plan hints are statement fragments that provide additional information that the data source optimizer uses to improve query performance. The data source optimizer uses the plan hints to decide whether or not to use an index and which index or which table join sequence to use. Valid values are Y and N. The default is N; plan hints are not to be enabled. Y specifies that plan hints are to be enabled at the data source if it supports plan hints.</td>
</tr>
<tr>
<td>PUSHDOWN</td>
<td>Specifies whether the federated server allows the data source to evaluate operations. Valid values are Y and N. The default is Y; the federated server allows the data source to evaluate operations. N specifies that the federated server retrieves columns from the remote data source.</td>
</tr>
<tr>
<td>VARCHAR_NO_TRAILING_BLANKS</td>
<td>Specifies for a particular server if VARCHAR columns contain any trailing blanks. To apply the option to a single column, use the VARCHAR_NO_TRAILING_BLANKS column option.</td>
</tr>
<tr>
<td>XA_OPEN_STRING_OPTIONS</td>
<td>Specifies additional information to append to the string. For example, the information might be the directory for trace files.</td>
</tr>
</tbody>
</table>

User mapping options

Table 132. User mapping options for Oracle

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FED_PROXY_USER</td>
<td>Specifies the authorization ID to use to establish all outbound trusted connections when the inbound connection is not trusted. <strong>Important:</strong> The user whose ID is specified in this option must have a user mapping that specifies both a REMOTE_AUTHID and a REMOTE_PASSWORD. If you specify the FED_PROXY_USER user mapping option, you must also specify the FED_PROXY_USER server option.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>REMOTE_AUTHID</td>
<td>Specifies the remote user ID to which the local user ID is mapped. If you do not specify this option, the ID that is used to connect to the federated database is used.</td>
</tr>
<tr>
<td>REMOTE_PASSWORD</td>
<td>Specifies the remote password for the remote user ID. If you do not specify this option, the password that is used to connect to the federated database is used.</td>
</tr>
<tr>
<td>USE_TRUSTED_CONTEXT</td>
<td>Specifies whether the user mapping is trusted. Valid values are Y and N. The default is N; the user mapping is not trusted and can be used only in non-trusted outbound connections. Y specifies that the user mapping is trusted and can be used in both trusted and non-trusted outbound connections.</td>
</tr>
</tbody>
</table>

## Column options

### Table 133. Column options for Oracle

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMERIC_STRING</td>
<td>Specifies whether the column contains strings of numeric characters that include blanks. Valid values are Y and N. The default is N; the column does not contain numeric strings that include blanks. If the column contains only numeric strings followed by trailing blanks, do not specify Y. When NUMERIC_STRING is set to Y for a column, the query optimizer recognizes that the column contains no blanks that could interfere with the sorting of the data in the column. 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.</td>
</tr>
<tr>
<td>VARCHAR_NO_TRAILING_BLANKS</td>
<td>Specifies if the column contains any trailing blanks.</td>
</tr>
</tbody>
</table>

## Script options reference

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, user mapping, nickname, and column options.

### Wrapper options

The following tables list the options that apply to this data source and identify the required options that you must specify.
### Wrapper options for scripts

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Required. Specifies whether the wrapper runs in fenced mode or in trusted mode. Valid values are Y and N. The default is N; the wrapper runs in trusted mode.</td>
</tr>
<tr>
<td>PROXY_TYPE</td>
<td>Specifies the proxy type to use to access the Internet when the federated server is behind a firewall. Valid values are NONE and SOCKS. The default value is NONE.</td>
</tr>
<tr>
<td>PROXY_SERVER_NAME</td>
<td>Specifies the name or IP address of the proxy server. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
<tr>
<td>PROXY_SERVER_PORT</td>
<td>Specifies the name or IP address of the proxy server. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
</tbody>
</table>

### Server options

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAEMON_PORT</td>
<td>Specifies the port number on which the Script daemon listens for Script job requests. The default is 4099. The port number must be the same as the DAEMON_PORT option in the daemon configuration file. If a service name is configured for the Script daemon, use a TCP/IP service name.</td>
</tr>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_QUERY</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. The default is 1. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests.</td>
</tr>
<tr>
<td>NODE</td>
<td>Required. Specifies the DNS host name or IP address of the system on which the Script daemon runs. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
<tr>
<td>PROXY_AUTHID</td>
<td>Specifies the user name for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY_PASSWORD</td>
<td>Specifies the password for proxy server authentication.</td>
</tr>
</tbody>
</table>
Table 135. Server options for scripts (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROXY_SERVER_NAME</td>
<td>Specifies the name or IP address of the proxy server. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
<tr>
<td>PROXY_SERVER_PORT</td>
<td>Specifies the port or service name for the proxy service on the proxy server. Valid values are a decimal port number from 1 to 32760 or a service name.</td>
</tr>
<tr>
<td>PROXY_TYPE</td>
<td>Specifies the proxy type to use to access the Internet when the federated server is behind a firewall. Valid values are NONE and SOCKS. The default value is NONE.</td>
</tr>
</tbody>
</table>

User mapping options

Table 136. User mapping options for scripts

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROXY_AUTHID</td>
<td>Specifies the user name for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY_PASSWORD</td>
<td>Specifies the password for proxy server authentication. The password is encrypted when it is stored in the federated database catalog.</td>
</tr>
</tbody>
</table>

Nickname options

Table 137. Nickname options for scripts

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATASOURCE</td>
<td>Required for the root nickname. Specifies the name of the script to invoke. The script that you specify as the value of this option must also be specified in the Script daemon configuration file. <strong>Important:</strong> This option valid only for the root nickname.</td>
</tr>
<tr>
<td>NAMESPACES</td>
<td>Specifies the namespaces that are associated with the namespace prefixes that are used in the XPATH and TEMPLATE options for each column. Use this syntax: NAMESPACES='prefix1=&quot;actual_namespace1&quot;, prefix2=&quot;actual_namespace2&quot;' Use a comma to separate multiple namespaces. For example: NAMESPACES='<a href="http://www.myweb.com/cust">http://www.myweb.com/cust</a>&quot;, i='<a href="http://www.myweb.com/cust/id">http://www.myweb.com/cust/id</a>&quot;, n='<a href="http://www.myweb.com/cust/name">http://www.myweb.com/cust/name</a>&quot;'</td>
</tr>
</tbody>
</table>
### Table 137. Nickname options for scripts (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STREAMING</td>
<td>Specifies whether the 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 source data fragment-by-fragment. This type of parsing minimizes memory use. Valid values are Y and N. The default is N; the documents are not parsed. This option is valid only on the root nickname. Do not set both the STREAMING and VALIDATE options to Y.</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>Specifies the maximum time, in minutes, to wait for results from the daemon. The default is 60. This option is valid only for the root nickname.</td>
</tr>
<tr>
<td>VALIDATE</td>
<td>Specifies whether the source document is validated to ensure that it conforms to an XML schema or document type definition (DTD) before data is extracted from it. The default is N; validation does not occur. Before you set the value to Y, the schema file or DTD file is in the location that the source document specifies. This option is valid only for the root nickname. Do not set both the STREAMING and VALIDATE options to Y.</td>
</tr>
<tr>
<td>XPATH</td>
<td>Specifies the XPath expression that identifies the XML elements that represent individual tuples. The XPATH nickname option for a child nickname is evaluated in the context of the path that is specified by the XPATH nickname option of its parent. This XPath expression is used as a context for evaluating column values that are identified by the presence of the XPATH column option.</td>
</tr>
</tbody>
</table>

### Column options

*Table 138. Column options for scripts*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>Specifies a default value for a script input column. If the SQL query does not provide a value, this default value is used.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FOREIGN_KEY</td>
<td>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 the option is case-sensitive. Do not specify the XPATH option for this column. The column can be used only to join a parent nickname and a child nickname. A CREATE NICKNAME statement that includes a FOREIGN_KEY option fails if the parent nickname has a different schema name. Unless the nickname that is referred to in a FOREIGN_KEY clause was explicitly defined as lowercase or mixed case in the CREATE NICKNAME statement, you must specify the nickname in uppercase when you refer to this nickname in the FOREIGN_KEY clause. When this option is set on a column, no other option can be set on the column.</td>
</tr>
<tr>
<td>INPUT_MODE</td>
<td>Specifies the input mode for the column. Valid values are CONFIG and FILE_INPUT. CONFIG treats the value as the input mode for a column. FILE_INPUT specifies a file that stores the value. The wrapper passes the specified value to the Script daemon.</td>
</tr>
<tr>
<td>POSITION</td>
<td>Specifies an integer value for positional parameters. If the positional value is set to an integer, then this input must be in this position in the command line. If this option is set, the switch is inserted into the appropriate location when the query runs. If POSITION is set to -1, the option is added as the last command line option. A POSITION integer value cannot be used twice in the same nickname. This option applies only to input columns.</td>
</tr>
<tr>
<td>PRIMARY_KEY</td>
<td>Required for a parent nickname that has one or more child nicknames. Specifies that this nickname is a parent nickname. The column data type must be VARCHAR(16). A nickname can have only one PRIMARY_KEY column option. Yes is the only valid value. Do not specify the XPATH option 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.</td>
</tr>
</tbody>
</table>
Table 138. Column options for scripts (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWITCH</td>
<td>Specifies a flag for the script on the command line. The value of this option preceded the column value that is supplied by WSSCRIPT.ARGS or by the default value, if any. If you do not specify a value for this option and a default value exists for the column, the default value is added without any switch information. This option is required for input columns.</td>
</tr>
<tr>
<td>SWITCH_ONLY</td>
<td>Enables the use of switches without a command line argument. Valid values are Y and N. Set this option to Y and the valid input values are Y and N. For an input value of Y, only the switch is added to the command line. For an input value of N, no value is added to the command line.</td>
</tr>
<tr>
<td>VALID_VALUES</td>
<td>Specifies a set of valid values for a column. Use a semicolon to separate multiple values.</td>
</tr>
<tr>
<td>XPATH</td>
<td>Specifies the Xpath expression in the XML document that contains the data that corresponds to this column. The wrapper evaluates this XPath expression after the CREATE NICKNAME statement applies the XPath expression from the XPATH nickname option.</td>
</tr>
</tbody>
</table>

**Sybase options reference**

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, user mapping, and column options.

**Wrapper options**

The following tables list the options that apply to this data source and identify the required options that you must specify.

Table 139. Wrapper options for Sybase

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Required. Specifies whether the wrapper runs in fenced mode or in trusted mode. On Microsoft Windows, valid values are Y and N. The default is N; the wrapper runs in trusted mode. On UNIX, the default and the only valid value is Y; the wrapper must run in fenced mode.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in written in Java, specifies a case-sensitive string for the class name that corresponds to the user mapping repository class. For example, &quot;UserMappingRepositoryLDAP&quot;. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
</tbody>
</table>
### Table 139. Wrapper options for Sybase (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_UM_PLUGIN_LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
</tbody>
</table>

### Server options

#### Table 140. Server options for Sybase

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLLATING_SEQUENCE</td>
<td>Specifies whether the data source uses the same default collating sequence as the federated database. Valid values are Y, N, and I. I specifies a case-insensitive. The default is Y. The collating sequence specified for the federated server must match the collating sequence on the remote data source.</td>
</tr>
<tr>
<td>COMM_RATE</td>
<td>Specifies the communication rate, in megabytes per second, between the federated server and the data source server. Valid values are whole numbers that are greater than 0 and less than 2147483648. The default is 2.</td>
</tr>
<tr>
<td>CPU_RATIO</td>
<td>Specifies how much faster or slower the data source CPU is when compared to federated server CPU. Valid values are greater than 0 and less than 1x10^34. The default is 1.0. Values can be expressed in any valid double notation, for example, 123E10, 123, or 1.21E4. A setting of 1 indicates that the federated server and the data source server have the same CPU speed; a 1:1 ratio. A setting of 0.5 indicates that the federated server CPU speed is 50% slower than the data source CPU. A setting of 2 indicates that the federated CPU is twice as fast as the data source CPU.</td>
</tr>
<tr>
<td>CONV_EMPTY_STRING</td>
<td>Specifies whether the federated server converts an empty string into a space during replication tasks. Valid values are Y and N. The default is N; the federated server does not convert empty strings. Set this option to Y when the data source has a non-nullable character column that stores an empty string.</td>
</tr>
<tr>
<td>DBNAME</td>
<td>Required. Specifies the name of the database that you want to access. Obtain the name of the database from the Sybase server.</td>
</tr>
</tbody>
</table>
Table 140. Server options for Sybase (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_MAXIMAL_PUSHDOWN</td>
<td>Specifies the primary criteria that the query optimizer uses to choose an access plan. Valid values are Y and N. The default is N; the query optimizer chooses the plan that has the lowest estimated cost. Y specifies that the query optimizer choose the access plan that pushes down the most query operations to the data source.</td>
</tr>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_QUERY</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. The default is 1. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests.</td>
</tr>
<tr>
<td>DB2_TWO_PHASE_COMMIT</td>
<td>Specifies whether the federated server connects to the data source in two-phase commit protocol or one-phase commit protocol. Valid values are Y and N. The default is N; the federated server uses the one-phase commit protocol to connect. Y specifies that the federated server uses two-phase commit protocol to connect.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in written in Java, specifies a case-sensitive string for the class name that corresponds to the user mapping repository class. For example, &quot;UserMappingRepositoryLDAP&quot;. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN_LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
<tr>
<td>FOLD_ID</td>
<td>Specifies the case for the user ID that is sent to the data source. There is no default value; the federated server sends the user ID in uppercase; then if the uppercase user ID fails, the server sends the user ID in lowercase. Valid values are U (uppercase), L (lowercase), and N (null). Avoid using the null setting, which might result in poor performance.</td>
</tr>
<tr>
<td>FOLD_PW</td>
<td>Specifies the case for the password that is sent to the data source. There is no default value; the federated server sends the password in uppercase; then if the uppercase password fails, the server sends the password in lowercase. Valid values are U (uppercase), L (lowercase), and N (null). Avoid using the null setting, which might result in poor performance.</td>
</tr>
</tbody>
</table>
Table 140. Server options for Sybase (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFILE</td>
<td>Specifies the path and name of the Sybase interface file to use instead of the default interface file. The Sybase wrapper searches for the interface file in the following places, in the order specified: On Microsoft Windows, in the IFILE server option, then in the %DB2PATH%\interfaces directory, and finally in the %SYBASE%\ini\sql.ini directory. On UNIX, in the IFILE server option, then in the sqllib/interfaces directory, and finally in the $SYBASE/interfaces directory.</td>
</tr>
<tr>
<td>IO_RATIO</td>
<td>Specifies how much faster or slower the data source I/O system runs when compared to the federated server I/O system. Valid values are greater than 0 and less than 1x10^25. The default is 1.0. Values can be expressed in any valid double notation, for example, 123E10, 123, or 1.21E4. A setting of 1 indicates that the federated server and the data source server have the same I/O speed; a 1:1 ratio. A setting of 0.5 indicates that the federated server speed is 50% slower than the data source speed. A setting of 2 indicates that the federated speed is twice as fast as the data source speed.</td>
</tr>
<tr>
<td>LOGIN_TIMEOUT</td>
<td>Specifies the amount of time, in seconds, that the federated server waits before abandoning a login request. The default is 0; the federated server waits an unlimited amount of time.</td>
</tr>
<tr>
<td>NODE</td>
<td>Required. Specifies the name of the node where the Sybase server resides. The node name is in the Sybase interfaces file.</td>
</tr>
<tr>
<td>OLD_NAME_GEN</td>
<td>Specifies how to convert the column names and index names that are in the data source into nickname column names and local index names for the federated server. Valid values are Y and N. The default is N; the generated names closely match the names in the data source. Y specifies that the generated names are the same as the names that were created in IBM WebSphere Federation Server Version 9 and earlier. Thus, the names might not closely match the data source names.</td>
</tr>
<tr>
<td>PACKET_SIZE</td>
<td>Specifies the packet size, in bytes, that the client library uses to send tabular data stream (TDS) packets. If the Sybase wrapper needs to send or receive large amounts of text and image data, increase the PACKET_SIZE.</td>
</tr>
</tbody>
</table>
Table 140. Server options for Sybase (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLAN_HINTS</td>
<td>Specifies whether or not plan hints are to be enabled. Plan hints are statement fragments that provide additional information that the data source optimizer uses to improve query performance. The data source optimizer uses the plan hints to decide whether or not to use an index and which index or which table join sequence to use. Valid values are Y and N. The default is N; plan hints are not to be enabled. Y specifies that plan hints are to be enabled at the data source if it supports plan hints.</td>
</tr>
<tr>
<td>PUSHDOWN</td>
<td>Specifies whether the federated server allows the data source to evaluate operations. Valid values are Y and N. The default is Y; the data source evaluates operations. N specifies that the federated server send 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 are not included in any SQL that the federated server sends to the data source.</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>Specifies the maximum time, in seconds, that the federated server waits for the remote server to respond to a command. The default is 0, which specifies an unlimited amount of time.</td>
</tr>
<tr>
<td>XA_OPEN_STRING_OPTIONS</td>
<td>Specifies open strings for the Sybase DTM XA interface. These strings are in addition to the LRM name, user name, and password.</td>
</tr>
</tbody>
</table>

User mapping options

Table 141. User mapping options for Sybase

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_AUTHID</td>
<td>Specifies the remote user ID to which the local user ID is mapped. If you do not specify this option, the ID that is used to connect to the federated database is used.</td>
</tr>
<tr>
<td>REMOTE_PASSWORD</td>
<td>Specifies the remote password for the remote user ID. If you do not specify this option, the password that is used to connect to the federated database is used.</td>
</tr>
</tbody>
</table>
### Column options

*Table 142. Column options for Sybase*

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMERIC_STRING</td>
<td>Specifies how to treat numeric strings. The default is N. If the data source string column contains only numeric strings and no other characters, including blanks, set the NUMERIC_STRING option to Y. When NUMERIC_STRING is set to Y for a column, the query optimizer recognizes that the column contains no blanks that could interfere with the sorting of the data in the column. 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.</td>
</tr>
</tbody>
</table>

### Teradata options reference

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, user mapping, and column options.

#### Wrapper options

The following tables list the options that apply to this data source and identify the required options that you must specify.

*Table 143. Wrapper options for Teradata*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Required. Specifies whether the wrapper runs in fenced mode or in trusted mode. On Microsoft Windows, valid values are Y and N. The default is N; the wrapper runs in trusted mode. On UNIX, the default value is Y; the wrapper must run in fenced mode.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in written in Java, specifies a case-sensitive string for the class name that corresponds to the user mapping repository class. For example, &quot;UserMappingRepositoryLDAP&quot;. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN_LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
</tbody>
</table>
## Server options

*Table 144. Server options for Teradata*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLLATING_SEQUENCE</td>
<td>Specifies whether the data source uses the same default collating sequence as the federated database. Valid values are Y, N, and I. I specifies a case-insensitive. The default is Y. The collating sequence specified for the federated server must match the collating sequence on the remote data source.</td>
</tr>
<tr>
<td>COMM_RATE</td>
<td>Specifies the communication rate, in megabytes per second, between the federated server and the data source server. Valid values are whole numbers that are greater than 0 and less than 2147483648. The default is 2.</td>
</tr>
<tr>
<td>CPU_RATIO</td>
<td>Specifies how much faster or slower the data source CPU is when compared to federated server CPU. Valid values are greater than 0 and less than 1x10^2. The default is 1.0. Values can be expressed in any valid double notation, for example, 123E10, 123, or 1.21E4. A setting of 1 indicates that the federated server and the data source server have the same CPU speed; a 1:1 ratio. A setting of 0.5 indicates that the federated server CPU speed is 50% slower than the data source CPU. A setting of 2 indicates that the federated CPU is twice as fast as the data source CPU.</td>
</tr>
<tr>
<td>DB2_MAXIMAL_PUSHDOWN</td>
<td>Specifies the primary criteria that the query optimizer uses to choose an access plan. Valid values are Y and N. The default is N; the query optimizer chooses the plan that has the lowest estimated cost. Y specifies that the query optimizer choose the access plan that pushes down the most query operations to the data source.</td>
</tr>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. The default is 1. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests.</td>
</tr>
<tr>
<td>QUERY</td>
<td>DB2_UUM_PLUGIN</td>
</tr>
</tbody>
</table>
### Server options for Teradata (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_UM_PLUGIN_LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
<tr>
<td>IO_RATIO</td>
<td>Specifies how much faster or slower the data source I/O system runs when compared to the federated server I/O system. Valid values are greater than 0 and less than 1x10^23. The default is 1.0. Values can be expressed in any valid double notation, for example, 123E10, 123, or 1.21E4. A setting of 1 indicates that the federated server and the data source server have the same I/O speed; a 1:1 ratio. A setting of 0.5 indicates that the federated server speed is 50% slower than the data source speed. A setting of 2 indicates that the federated speed is twice as fast as the data source speed.</td>
</tr>
<tr>
<td>NODE</td>
<td>Required. Specifies the alias name or IP address of the Teradata server.</td>
</tr>
<tr>
<td>OLD_NAME_GEN</td>
<td>Specifies how to convert the column names and index names that are in the data source into nickname column names and local index names for the federated server. Valid values are Y and N. The default is N; the generated names closely match the names in the data source. Y specifies that the generated names are the same as the names that were created in IBM WebSphere Federation Server Version 9 and earlier. Thus, the names might not closely match the data source names.</td>
</tr>
<tr>
<td>PUSHDOWN</td>
<td>Specifies whether the federated server allows the data source to evaluate operations. Valid values are Y and N. The default is Y; the data source evaluates operations. N specifies that the federated server send 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 are not included in any SQL that the federated server sends to the data source.</td>
</tr>
</tbody>
</table>

### User mapping options

**Table 145. User mapping options for Teradata**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE.AUTHID</td>
<td>Specifies the remote user ID to which the local user ID is mapped. If you do not specify this option, the ID that is used to connect to the federated database is used.</td>
</tr>
</tbody>
</table>
Table 145. User mapping options for Teradata (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_PASSWORD</td>
<td>Specifies the remote password for the remote user ID. If you do not specify this option, the password that is used to connect to the federated database is used.</td>
</tr>
</tbody>
</table>

Column options

Table 146. Column options for Teradata

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMERIC_STRING</td>
<td>Specifies how to treat numeric strings. The default is N. If the data source string column contains only numeric strings and no other characters, including blanks, set the NUMERIC_STRING option to Y. When NUMERIC_STRING is set to Y for a column, the query optimizer recognizes that the column contains no blanks that could interfere with the sorting of the data in the column. 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.</td>
</tr>
</tbody>
</table>

Table-structured file options reference

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, nickname, and column options.

The following tables list the options that apply to this data source and identify the required options that you must specify.

Wrapper options

Table 147. Wrapper options for table-structured files

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Required. Specifies whether the wrapper runs in fenced mode or in trusted mode. Valid values are Y and N. The default is N; the wrapper runs in trusted mode.</td>
</tr>
</tbody>
</table>
## Server options

*Table 148. Server options for table-structured files*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_QUERY</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. The default is 1. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests.</td>
</tr>
</tbody>
</table>

## Nickname options

*Table 149. Nickname options for table-structured files*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLUMN_DELIMITER</td>
<td>Specifies a single character to use as the delimiter that separates columns in the table-structured file. The default 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 by 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.</td>
</tr>
<tr>
<td>CODEPAGE</td>
<td>Specifies the code page of the file at the data source. This option is valid only for federated databases that use Unicode. The source data is converted from the specified code page to Unicode.</td>
</tr>
<tr>
<td>FILE_PATH</td>
<td>Specifies the fully qualified path to the table-structured file. Enclose the file name in single quotation marks. The data file must be a standard file or a symbolic link, rather than a pipe or another non-standard file type. <strong>Important</strong>: If you specify the FILE_PATH option, do not specify a DOCUMENT column.</td>
</tr>
<tr>
<td>KEY_COLUMN</td>
<td>Specifies the name of the column on which the file is sorted. A column that has the DOCUMENT column option cannot be the key column. Only single-column keys are supported. 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 as non-nullable by adding the NOT NULL option to its definition in the nickname statement. This value is case-sensitive.</td>
</tr>
</tbody>
</table>
Table 149. Nickname options for table-structured files (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SORTED</td>
<td>Specifies whether the file at the data source is or is not sorted in ascending order. Valid values are Y and N. The default is N; the file at the data source is not sorted in ascending order. 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.</td>
</tr>
<tr>
<td>VALIDATE_DATA_FILE</td>
<td>For sorted files, this option specifies whether the wrapper verifies that the key column is sorted in ascending order and checks for null keys. This validation occurs only once when the nickname is first created. The default is N; sort order is not verified. This option is valid only when the SORTED option is set to Y and the DOCUMENT option is not specified.</td>
</tr>
</tbody>
</table>

Column options

Table 150. Column options for table-structured files

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOCUMENT</td>
<td>Allows you to specify the file path when the query runs, instead of when you create the nickname. The only valid value is FILE. Only one column of each nickname can be specified with the DOCUMENT option. 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.</td>
</tr>
</tbody>
</table>

WebSphere Business Integration options reference

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, nickname, and column options.

Wrapper options

The following tables list the options that apply to this data source and identify the required options that you must specify.
### Wrapper options for WebSphere Business Integration

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Required. Specifies whether the wrapper runs in fenced mode or in trusted mode. Valid values are Y and N. The default is N; the wrapper runs in trusted mode.</td>
</tr>
</tbody>
</table>

### Server options for WebSphere Business Integration

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP_TYPE</td>
<td>Required. Specifies the type of remote application. Valid values are PSOFT, SAP, and SIEBEL. The default is SAP.</td>
</tr>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_QUERY</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. The default is 1. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests.</td>
</tr>
<tr>
<td>FAULT_QUEUE</td>
<td>Required. Specifies the name of the fault queue that delivers error messages from the adapter to the wrapper. The specified name must conform to the specifications for queue names for WebSphere MQ.</td>
</tr>
<tr>
<td>MQCONN_NAME</td>
<td>Specifies the host name or network address of the computer that runs WebSphere MQ server. For example, a valid value is 9.30.76.151 (1420), where 1420 is the port number. If no port number is specified, the default port 1414 is used. If this option is not set, the MQSERVER environment variable that is specified in the db2dj.ini file is used to select the channel definition. If this environment variable is not set, the client channel table is used.</td>
</tr>
<tr>
<td>MQ_MANAGER</td>
<td>Required. Specifies the valid name of any valid WebSphere MQ manager.</td>
</tr>
<tr>
<td>MQ_RESPONSE_TIMEOUT</td>
<td>Required. Specifies the amount of time, in milliseconds, that the wrapper waits for a response message from the response queue. The default is 50000. Specify -1 to indicate that there is no timeout period.</td>
</tr>
<tr>
<td>MQ_SVRCONN_CHANNELNAME</td>
<td>Specifies the name of the server-connection channel on the WebSphere MQ Manager to connect to. This option is valid only if the MQCONN_NAME server option is specified. The default is SYSTEM.DEF.SVRCONN.</td>
</tr>
</tbody>
</table>
Table 152. Server options for WebSphere Business Integration (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESPONSE_QUEUE</td>
<td>Required. Specifies 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.</td>
</tr>
<tr>
<td>RESULTSET_ENABLED</td>
<td>Required. Specifies whether the adapter is enabled to support result sets. Valid values are Y and N. The default is N; the adapter is not enabled to support result sets. Before you set the option to Y, ensure that you are using MQ Server Version 2.6 or later and that the ResultSetEnabled property in the adapter configuration file is set to True.</td>
</tr>
<tr>
<td>REQUEST_QUEUE</td>
<td>Required. Specifies the name of the request queue that delivers query results from the wrapper to the adapter. The name must conform to the specifications for queue names for WebSphere MQ.</td>
</tr>
</tbody>
</table>

Nickname options

Table 153. Nickname options for WebSphere Business Integration

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUSOBJ_NAME</td>
<td>Specifies the name of the XML schema definition file (.xsd) that represents the business object. Specify this option only for the root nickname. The value of this option is provided to the adapter when queries are sent. For example, 'sap_bapi_customer_get_detail2'.</td>
</tr>
</tbody>
</table>
| NAMESPACES        | Specifies the namespaces that are associated with the namespace prefixes that are used in the XPATH and TEMPLATE options for each column. Use this syntax:<br/>NAMESPACES='prefix1="actual_namespace1", prefix2="actual_namespace2"'<br/>Use a comma to separate multiple namespaces. For example:<br/>NAMESPACES='http://www.myweb.com/cust", i="http://www.myweb.com/cust/id", n="http://www.myweb.com/cust/name"'
| RESULTSET_SIZE    | Specifies how many business objects the adapter returns. The default is 0; all business objects are returned. Specify any other value to have that only number of objects returned, but keep in mind that the result set might be incomplete. Specify this option only for the root nickname. This option is valid only when the RESULTSET_ENABLED server option is set to Y. |
### Table 153. Nickname options for WebSphere Business Integration (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQUIRE_PREDICATE</td>
<td>Specifies whether or not to require an equality predicate on at least one input column in all queries on the nickname hierarchy. Requiring an equality predicate can limit the size of the result set. Valid values are Y and N. The default is Y; an equality predicate is required. If you know that the size of the result set will not exceed the Java virtual machine (JVM) memory limit, set the value to N. Specify this option only for the root nickname. This option is valid only when the RESULTSET_ENABLED server option is set to Y.</td>
</tr>
<tr>
<td>STREAMING</td>
<td>Specifies whether the 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 source data fragment-by-fragment. This type of parsing minimizes memory use. Valid values are Y and N. The default is N; the documents are not parsed. This option is valid only on the root nickname.</td>
</tr>
<tr>
<td>TEMPLATE</td>
<td>Specifies the nickname template fragment to use to construct an XML input document. The name must conform to the specified template syntax. Specify this option only on the root nickname.</td>
</tr>
<tr>
<td>XPATH</td>
<td>Required. Specifies the XPath expression that identifies the elements that represent the individual tuples. The XPath expression is used as a context for evaluating column values that the XPATH nickname column option identifies.</td>
</tr>
</tbody>
</table>

### Column options

**Table 154. Column options for WebSphere Business Integration**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESCAPE_INPUT</td>
<td>Specifies whether XML special characters are replaced in XML input values or not. Use this option to include XML fragments, for example XML fragments with repeating elements, as input. Valid values are Y and N. The default is N; the XML input values are replaced with special characters that XML uses to represent input characters. Y specifies that input characters are preserved exactly as they appear. The column data type must be VARCHAR or CHAR. To use the ESCAPE_INPUT column option, you must also specify the TEMPLATE column option.</td>
</tr>
</tbody>
</table>
Table 154. Column options for WebSphere Business Integration  (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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 the option is case-sensitive. Do not specify the XPATH option for this column. The column can be used only to join a parent nicknames and a child nickname. A CREATE NICKNAME statement that includes a FOREIGN_KEY option fails if the parent nickname has a different schema name. Unless the nickname that is referred to in a FOREIGN_KEY clause was explicitly defined as lowercase or mixed case in the CREATE NICKNAME statement, you must specify the nickname in uppercase when you refer to this nickname in the FOREIGN_KEY clause.  
**Note:**  
- When this option is set on a column, no other option can be set on the column.  
- If you set this column option, you cannot later use the ALTER NICKNAME statement to drop the option. Instead, you must drop the nickname and then create the nickname again without this column option. |
| PRIMARY_KEY  | Required for a parent nickname that has one or more child nicknames. Specifies that this nickname is a parent nickname. The column data type must be VARCHAR(16). A nickname can have only one PRIMARY_KEY column option. Yes is the only valid value. Do not specify the XPATH option for this column. The column can be used only to join parent nicknames and child nicknames.  
**Note:**  
- When this option is set on a column, no other option can be set on the column.  
- If you set this column option, you cannot later use the ALTER NICKNAME statement to drop the option. Instead, you must drop the nickname and then create the nickname again without this column option. |
| TEMPLATE     | Specifies the nickname template fragment to use to construct an XML input document. The name must conform to the specified template syntax. Specify this option only for the root nickname.  
**Note:** If you set this column option, you cannot later use the ALTER NICKNAME statement to drop the option. Instead, you must drop the nickname and then create the nickname again without this column option. |
Table 154. Column options for WebSphere Business Integration (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPATH</td>
<td>Specifies the XPath expression in the XML document that contains the data that corresponds to this column. The wrapper evaluates this XPath expression after the CREATE NICKNAME statement applies the XPath expression from the XPATH nickname option. <strong>Note:</strong> If you set this column option, you cannot later use the ALTER NICKNAME statement to drop the option. Instead, you must drop the nickname and then create the nickname again without this column option.</td>
</tr>
</tbody>
</table>

Web services options reference

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, user mapping, nickname, and column options.

Wrapper options

The following tables list the options that apply to this data source and identify the required options that you must specify.

Table 155. Wrapper options for Web services

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Required. Specifies whether the wrapper runs in fenced mode or in trusted mode. Valid values are Y and N. The default is N; the wrapper runs in trusted mode.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in written in Java, specifies a case-sensitive string for the class name that corresponds to the user mapping repository class. For example, &quot;UserMappingRepositoryLDAP&quot;. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
<tr>
<td>DB2_UM_PLUGIN_LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
<tr>
<td>PROXY_TYPE</td>
<td>Specifies the proxy type to use to access the Internet when the federated server is behind a firewall. Valid values are NONE, HTTP, and SOCKS. The default value is NONE.</td>
</tr>
<tr>
<td>PROXY_SERVER_NAME</td>
<td>Specifies the name or IP address of the proxy server. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
<tr>
<td>PROXY_SERVER_PORT</td>
<td>Specifies the port or service name for the proxy service on the proxy server. Valid values are a decimal port number from 1 to 32760 or a service name.</td>
</tr>
</tbody>
</table>
Table 155. Wrapper options for Web services (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL_KEYSTORE_FILE</td>
<td>Specifies the certificate storage file for communications that use SSL or TSL. A valid value is a fully qualified path name that is accessible by the federated database agent or by a fenced-mode process. The default is install path/cfg/WSWrapperKeystore.kdb.</td>
</tr>
<tr>
<td>SSL_KEYSTORE_PASSWORD</td>
<td>Specifies the password to use to access the file in the SSL_KEYSTORE_FILE option. Valid values are a password, which is encrypted when it is stored in the federated database catalog, and file:file_name, where file_name is the fully qualified path to a stash file.</td>
</tr>
<tr>
<td>SSL_VERIFY_SERVER_CERTIFICATE</td>
<td>Specifies whether the server certificate is verified during SSL authentication. Valid values are Y and N. The default is N; the certificate is not verified.</td>
</tr>
</tbody>
</table>

Server options

Table 156. Server options for Web services

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_QUERY</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. The default is 1. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests.</td>
</tr>
<tr>
<td>DB2 UM PLUGIN</td>
<td>Specifies the implementation of the user mapping plug-in. For a plug-in written in Java, specifies a case-sensitive string for the class name that corresponds to the user mapping repository class. For example, &quot;UserMappingRepositoryLDAP&quot;. For a plug-in written in C, specifies any valid C library name.</td>
</tr>
<tr>
<td>DB2 UM PLUGIN LANG</td>
<td>Specifies the language of the user mapping plug-in. Valid values are Java and C. The default is Java.</td>
</tr>
<tr>
<td>PROXY AUTHID</td>
<td>Specifies the user name for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY PASSWORD</td>
<td>Specifies the password for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY SERVER NAME</td>
<td>Specifies the name or IP address of the proxy server. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
</tbody>
</table>
Table 156. Server options for Web services  (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROXY_SERVER_PORT</td>
<td>Specifies the port or service name for the proxy service on the proxy server. Valid values are a decimal port number from 1 to 32760 or a service name.</td>
</tr>
<tr>
<td>PROXY_TYPE</td>
<td>Specifies the proxy type to use to access the Internet when the federated server is behind a firewall. Valid values are NONE, HTTP, and SOCKS. The default value is NONE.</td>
</tr>
<tr>
<td>SSL_CLIENT_CERTIFICATE_LABEL</td>
<td>Specifies the client certificate to send during SSL authentication. If you do not specify a value, the current authorization ID for the federated database is used to locate the certificate.</td>
</tr>
<tr>
<td>SSL_KEYSTORE_FILE</td>
<td>Specifies the certificate storage file for communications that use SSL or TSL. A valid value is a fully qualified path name that is accessible by the federated database agent or by a fenced-mode process. The default is install path/cfg/WSWrapperKeystore.kdb.</td>
</tr>
<tr>
<td>SSL_KEYSTORE_PASSWORD</td>
<td>Specifies the password to use to access the file in the SSL_KEYSTORE_FILE option. Valid values are a password, which is encrypted when it is stored in the federated database catalog, and file:file_name, where file_name is the fully qualified path to a stash file.</td>
</tr>
<tr>
<td>SSL_VERIFY_SERVER_CERTIFICATE</td>
<td>Specifies whether to verify the server certificate during SSL authentication. Valid values are Y and N. The default is N; the certificate is not verified.</td>
</tr>
</tbody>
</table>

User mapping options

Table 157. User mapping options for Web services

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROXY_AUTHID</td>
<td>Specifies the user name for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY_PASSWORD</td>
<td>Specifies the password for proxy server authentication. The password is encrypted when it is stored in the federated database catalog.</td>
</tr>
<tr>
<td>REMOTE_AUTHID</td>
<td>Specifies the remote user ID to which the local user ID is mapped. If you do not specify this option, the ID that is used to connect to the federated database is used.</td>
</tr>
<tr>
<td>REMOTE_PASSWORD</td>
<td>Specifies the remote password for the remote user ID. If you do not specify this option, the password that is used to connect to the federated database is used.</td>
</tr>
</tbody>
</table>
Table 157. User mapping options for Web services (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL_CLIENT_CERTIFICATE_LABEL</td>
<td>Specifies the client certificate to send during SSL authentication. If you do not specify a value, the current federated database authorization ID is used to locate the certificate.</td>
</tr>
</tbody>
</table>

Nickname options

Table 158. Nickname options for Web services

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| NAMESPACES | Specifies the namespaces that are associated with the namespace prefixes that are used in the XPath and TEMPLATE options for each column. Use this syntax:

```
NAMESPACES 'prefix1'="actual_namespace1", prefix2="actual_namespace2"
```

Use a comma to separate multiple namespaces. For example:

```
```

| SOAPACTION | Required for the root nickname. Specifies the URI SOAPACTION attribute from the Web Services Description Language (WSDL) format. The URL can contain a colon-separated IPv6 address if it is enclosed in square brackets. For example:

```
http://[1080:0:0:8:800:200C:417A]
```

**Note:** This option is not valid for nonroot nicknames. |

| STREAMING | Specifies whether the 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 source data fragment-by-fragment. This type of parsing minimizes memory use. Valid values are Y and N. The default is N; the documents are not parsed. This option is valid only on the root nickname. |

| TEMPLATE | Specifies the nickname template fragment to use to construct a SOAP request. The fragment must conform to the specified template syntax. This option is valid only for the root nickname. |
Table 158. Nickname options for Web services  (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>Required for the root nickname. Specifies the URL for the Web service endpoint. Supported protocols are HTTP and HTTPS. The URL can contain a colon-separated IPv6 address if it is enclosed in square brackets. For example:http://[1080:0:0:8:800:200C:417A]</td>
</tr>
<tr>
<td>XML_CODESET</td>
<td>Specifies the encoding to use to send and receive XML data. This option overrides the internal encoding.</td>
</tr>
<tr>
<td>XPATH</td>
<td>Required. Specifies the Xpath expression that identifies the SOAP response elements that represent individual tuples. The Xpath expression is used as a context for evaluating column values that the XPATH nickname column options identify.</td>
</tr>
</tbody>
</table>

Column options

Table 159. Column options for Web services

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESCAPE_INPUT</td>
<td>Specifies whether XML special characters are replaced in XML input values or not. Use this option to include XML fragments as input, for example, to include XML fragments that have repeating elements. Valid values are Y and N. N is the default; XML input values are retained. The column data type must be VARCHAR or CHAR. If ESCAPE_INPUT is set to Y, you must also specify the TEMPLATE column option.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 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 the option is case-sensitive. Do not specify the XPATH option for this column. The column can be used only to join a parent nicknames and a child nickname. A CREATE NICKNAME statement that includes a FOREIGN_KEY option fails if the parent nickname has a different schema name. Unless the nickname that is referred to in a FOREIGN_KEY clause was explicitly defined as lowercase or mixed case in the CREATE NICKNAME statement, you must specify the nickname in uppercase when you refer to this nickname in the FOREIGN_KEY clause. **Note:**  
  • When this option is set on a column, no other option can be set on the column.  
  • If you set this column option, you cannot later use the ALTER NICKNAME statement to drop the option. Instead, you must drop the nickname and then create the nickname again without this column option. |
| PRIMARY_KEY        | Required for a parent nickname that has one or more child nicknames. Specifies that this nickname is a parent nickname. The column data type must be VARCHAR(16). A nickname can have only one PRIMARY_KEY column option. Yes is the only valid value. Do not specify the XPATH option for this column. The column can be used only to join parent nicknames and child nicknames. **Note:**  
  • When this option is set on a column, no other option can be set on the column.  
  • If you set this column option, you cannot later use the ALTER NICKNAME statement to drop the option. Instead, you must drop the nickname and then create the nickname again without this column option. |
| SOAPACTIONCOLUMN   | Specifies the column that dynamically specifies the SOAP action for the Web Services endpoint when you run a query. This option is valid only for the root nickname. If the host name is an IPv6 (colon-separated) address, enclose the host name in square brackets. For example: ’http://[1080:0:0:8:800:200C:417A]:99/soap’ When this option is set on a column, no other option can be set on the column. |
Table 159. Column options for Web services  (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMPLATE</td>
<td>Specifies the column template fragment to use to construct the XML input document. The fragment must conform to the specified template syntax. Note: If you set this column option, you cannot later use the ALTER NICKNAME statement to drop the option. Instead, you must drop the nickname and then create the nickname again without this column option.</td>
</tr>
<tr>
<td>URLCOLUMN</td>
<td>Specifies the column that dynamically specifies the SOAP action for the Web Services endpoint when you run a query. This option is valid only for the root nickname. If the host name is an IPv6 (colon-separated) address, enclose the host name in square brackets. For example: <code>http://[1080:0:0:0:8:800:200C:417A]:99/soap</code> When this option is set on a column, no other option can be set on the column.</td>
</tr>
<tr>
<td>XPATH</td>
<td>Specifies the Xpath expression in the XML document that contains the data that corresponds to this column. The wrapper evaluates this XPath expression after the CREATE NICKNAME statement applies the XPath expression from the XPATH nickname option. Note: If you set this column option, you cannot later use the ALTER NICKNAME statement to drop the option. Instead, you must drop the nickname and then create the nickname again without this column option.</td>
</tr>
</tbody>
</table>

XML options reference

To configure how the federated server and its users interact with a data source, set and modify wrapper, server, user mapping, nickname, and column options.

Wrapper options

The following tables list the options that apply to this data source and identify the required options that you must specify.

Table 160. Wrapper options for XML

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_FENCED</td>
<td>Required. Specifies whether the wrapper runs in fenced mode or in trusted mode. Valid values are Y and N. N is the default; the wrapper runs in trusted mode.</td>
</tr>
<tr>
<td>PROXY_TYPE</td>
<td>Specifies the proxy type to use to access the Internet when the federated server is behind a firewall. Valid values are NONE, HTTP, and SOCKS. The default value is NONE.</td>
</tr>
</tbody>
</table>
## Table 160. Wrapper options for XML (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROXY_SERVER_NAME</td>
<td>Specifies the name or IP address of the proxy server. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
<tr>
<td>PROXY_SERVER_PORT</td>
<td>Specifies the port or service name for the proxy service on the proxy server. Valid values are a decimal port number from 1 to 32760 or a service name.</td>
</tr>
<tr>
<td>SSL_KEYSTORE_FILE</td>
<td>Specifies the certificate storage file for communications that use SSL or TSL. A valid value is a fully qualified path name that is accessible by the federated database agent or by a fenced-mode process.</td>
</tr>
<tr>
<td>SSL_KEYSTORE_PASSWORD</td>
<td>Specifies the password to use to access the file in the SSL_KEYSTORE_FILE option. Valid values are a password, which is encrypted when it is stored in the federated database catalog, and file:file_name, where file_name is the fully qualified path to a stash file.</td>
</tr>
<tr>
<td>SSL_VERIFY_SERVER_CERTIFICATE</td>
<td>Specifies whether the server certificate is verified during SSL authentication. Valid values are Y and N. The default is N; the certificate is not verified.</td>
</tr>
</tbody>
</table>

## Server options

### Table 161. Server options for XML

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2_MAX_ASYNC_REQUESTS_PER_QUERY</td>
<td>Specifies the maximum number of concurrent asynchronous requests from a query. Valid values are from -1 to 64000. The default is 1. -1 specifies that the federated query optimizer determines the number of requests. 0 specifies that the data source cannot accommodate additional asynchronous requests.</td>
</tr>
<tr>
<td>PROXY_AUTHID</td>
<td>Specifies the user name for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY_PASSWORD</td>
<td>Specifies the password for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY_SERVER_NAME</td>
<td>Specifies the name or IP address of the proxy server. Valid IP addresses are in IPv4 (dot-separated) format or in IPv6 (colon-separated) format. Use IPv6 format only if IPv6 is configured.</td>
</tr>
<tr>
<td>PROXY_SERVER_PORT</td>
<td>Specifies the port or service name for the proxy service on the proxy server. Valid values are a decimal port number from 1 to 32760 or a service name.</td>
</tr>
</tbody>
</table>
### Table 161. Server options for XML (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROXY_TYPE</td>
<td>Specifies the proxy type to use to access the Internet when the federated server is behind a firewall. Valid values are NONE, HTTP, and SOCKS. The default value is NONE.</td>
</tr>
<tr>
<td>SOCKET_TIMEOUT</td>
<td>Specifies the maximum time, in minutes, that the federated server waits for results from the proxy server. A valid value is any number that is greater than or equal to 0. The default is 0; the server waits an unlimited amount of time.</td>
</tr>
<tr>
<td>SSL_CLIENT_CERTIFICATE_LABEL</td>
<td>Specifies the client certificate to send during SSL authentication. If you do not specify a value, the current authorization ID for the federated database is used to locate the certificate.</td>
</tr>
<tr>
<td>SSL_KEYSTORE_FILE</td>
<td>Specifies the certificate storage file for communications that use SSL or TSL. A valid value is a fully qualified path name that is accessible by the federated database agent or by a fenced-mode process.</td>
</tr>
<tr>
<td>SSL_KEYSTORE_PASSWORD</td>
<td>Specifies the password to use to access the file in the SSL_KEYSTORE_FILE option. Valid values are a password, which is encrypted when it is stored in the federated database catalog, and file:file_name, where file_name is the fully qualified path to a stash file.</td>
</tr>
<tr>
<td>SSL_VERIFY_SERVER_CERTIFICATE</td>
<td>Specifies whether to verify the server certificate during SSL authentication. The default is N; the certificate is not verified.</td>
</tr>
</tbody>
</table>

### User mapping options

#### Table 162. User mapping options for XML

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROXY_AUTHID</td>
<td>Specifies the user name for proxy server authentication.</td>
</tr>
<tr>
<td>PROXY_PASSWORD</td>
<td>Specifies the password for proxy server authentication.</td>
</tr>
<tr>
<td>SSL_CLIENT_CERTIFICATE_LABEL</td>
<td>Specifies the client certificate to send during SSL authentication. If you do not specify a value, the current authorization ID for the federated database is used to locate the certificate.</td>
</tr>
</tbody>
</table>
# Nickname options

*Table 163. Nickname options for XML*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIRECTORY_PATH</td>
<td>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 valid only for the root nickname.</td>
</tr>
<tr>
<td>FILE_PATH</td>
<td>Specified the file path of the XML document. If you specify FILE_PATH, do not specify a DOCUMENT column. This option is valid only for the root nickname.</td>
</tr>
<tr>
<td>INSTANCE_PARSE_TIME</td>
<td>Specifies the time, in milliseconds, that is required to parse one row of the XML source document. The valid value can be an integer or decimal value. The default is 7. This option is valid only for columns of the root nickname. To optimize queries of large or complex XML source structures, modify the INSTANCE_PARSE_TIME, XPATH_EVAL_TIME, and NEXT_TIME options.</td>
</tr>
<tr>
<td>NAMESPACES</td>
<td>Specifies the namespaces that are associated with the namespace prefixes that are used in the XPATH and TEMPLATE options for each column. Use this syntax: NAMESPACES='prefix1=actual_namespace1', 'prefix2=actual_namespace2' Use a comma to separate multiple namespaces. For example: NAMESPACES='<a href="http://www.myweb.com/cust">http://www.myweb.com/cust</a>', i='<a href="http://www.myweb.com/cust/id">http://www.myweb.com/cust/id</a>', n='<a href="http://www.myweb.com/cust/name">http://www.myweb.com/cust/name</a>'</td>
</tr>
<tr>
<td>NEXT_TIME</td>
<td>Specified the time, in milliseconds, that is required to locate subsequent source elements from the Xpath expression. The default is 1. This option is valid for root nicknames and nonroot nicknames. To optimize queries of large or complex XML source structures, modify the INSTANCE_PARSE_TIME, XPATH_EVAL_TIME, and NEXT_TIME options.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>STREAMING</td>
<td>Specifies whether the source document should be separated into logical</td>
</tr>
<tr>
<td></td>
<td>fragments for processing. The fragments correspond to the node that matches</td>
</tr>
<tr>
<td></td>
<td>the XPath expression of the nickname. The wrapper then parses and processes</td>
</tr>
<tr>
<td></td>
<td>source data fragment-by-fragment. This type of parsing minimizes memory use.</td>
</tr>
<tr>
<td></td>
<td>Valid values are Y and N. The default is N; the documents are not parsed.</td>
</tr>
<tr>
<td></td>
<td>This option is valid only on the root nickname.</td>
</tr>
<tr>
<td>VALIDATE</td>
<td>Specifies whether the source document is validated to ensure that it</td>
</tr>
<tr>
<td></td>
<td>conforms to an XML schema or document type definition (DTD) before data is</td>
</tr>
<tr>
<td></td>
<td>extracted from it. The default is N; validation does not occur. Before you</td>
</tr>
<tr>
<td></td>
<td>set the value to Y, the schema file or DTD file is in the location that the</td>
</tr>
<tr>
<td></td>
<td>source document specifies. This option is valid only for the root nickname.</td>
</tr>
<tr>
<td></td>
<td>Do not set both the STREAMING and VALIDATE options to Y.</td>
</tr>
<tr>
<td>XPATH</td>
<td>Required. Specifies the XPath expression that identifies the elements that</td>
</tr>
<tr>
<td></td>
<td>represent the individual tuples. The XPath expression is used as a context</td>
</tr>
<tr>
<td></td>
<td>for evaluating column values that are identified by the XPATH column option.</td>
</tr>
<tr>
<td>XPATH_EVAL_TIME</td>
<td>Specifies the time required, in milliseconds, to evaluate the XPath</td>
</tr>
<tr>
<td></td>
<td>expression of the nickname and to locate the first element. The value can</td>
</tr>
<tr>
<td></td>
<td>be an integer or decimal value. The default is 1. This option is valid for</td>
</tr>
<tr>
<td></td>
<td>root nicknames and nonroot nicknames. To optimize queries of large or</td>
</tr>
<tr>
<td></td>
<td>complex XML source structures, modify the INSTANCE_PARSE_TIME, XPATH_EVAL_</td>
</tr>
<tr>
<td></td>
<td>TIME, and NEXT_TIME options.</td>
</tr>
</tbody>
</table>
## Column options

*Table 164. Column options for XML*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOCUMENT</td>
<td>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 valid only for columns of the root nickname (the nickname that identifies the elements at the top level of the XML document). Only one column for each nickname can be specified with the DOCUMENT option. If you use a DOCUMENT column option instead of a FILE_PATH or DIRECTORY_PATH nickname option, the document that corresponds to this nickname is supplied when the query runs. Valid values are FILE, DIRECTORY, URI, and COLUMN. The default is FILE. The column must be of VARCHAR data type.</td>
</tr>
</tbody>
</table>
| 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 the option is case-sensitive. Do not specify the XPATH option for this column. The column can be used only to join a parent nicknames and a child nickname. A CREATE NICKNAME statement that includes a FOREIGN_KEY option fails if the parent nickname has a different schema name. Unless the nickname that is referred to in a FOREIGN_KEY clause was explicitly defined as lowercase or mixed case in the CREATE NICKNAME statement, you must specify the nickname in uppercase when you refer to this nickname in the FOREIGN_KEY clause. **Note:**  
  * When this option is set on a column, no other option can be set on the column.  
  * If you set this column option, you cannot later use the ALTER NICKNAME statement to drop the option. Instead, you must drop the nickname and then create the nickname again without this column option. |
### Table 164. Column options for XML (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIMARY_KEY</td>
<td>Required for a parent nickname that has one or more child nicknames. Specifies that this nickname is a parent nickname. The column data type must be VARCHAR(16). A nickname can have only one PRIMARY_KEY column option. Yes is the only valid value. Do not specify the XPATH option for this column. The column can be used only to join parent nicknames and child nicknames. <strong>Note:</strong>&lt;br&gt;• When this option is set on a column, no other option can be set on the column.&lt;br&gt;• If you set this column option, you cannot later use the ALTER NICKNAME statement to drop the option. Instead, you must drop the nickname and then create the nickname again without this column option.</td>
</tr>
<tr>
<td>XPATH</td>
<td>Specifies the Xpath expression in the XML document that contains the data that corresponds to this column. The wrapper evaluates this XPath expression after the CREATE NICKNAME statement applies the XPath expression from the XPATH nickname option. <strong>Note:</strong> If you set this column option, you cannot later use the ALTER NICKNAME statement to drop the option. Instead, you must drop the nickname and then create the nickname again without this column option.</td>
</tr>
</tbody>
</table>
Chapter 23. 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 that contain information pertinent to a federated system, such as the SYSCAT.WRAPPERS view.

The SYSCAT views are read-only. You cannot issue an update or insert operation on a view in the SYSCAT schema. 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>
<thead>
<tr>
<th>Catalog views</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSCAT.CHECKS</td>
<td>Contains check constraint information that you defined.</td>
</tr>
<tr>
<td>SYSCAT.COLCHECKS</td>
<td>Contains columns referenced by a check constraint.</td>
</tr>
<tr>
<td>SYSCAT.COLUMNS</td>
<td>Contains column information about the data source objects (tables and views) that you created nicknames for.</td>
</tr>
<tr>
<td>SYSCAT.COLOPTIONS</td>
<td>Contains information about column option values that you set for a nickname.</td>
</tr>
<tr>
<td>SYSCAT.CONSTDEP</td>
<td>Contains the dependency of an informational constraint that you defined.</td>
</tr>
<tr>
<td>SYSCAT.DATATYPES</td>
<td>Contains data type information about local built-in and user-defined DB2 data types.</td>
</tr>
<tr>
<td>SYSCAT.DBAUTH</td>
<td>Contains the database authorities held by individual users and groups.</td>
</tr>
<tr>
<td>SYSCAT.FUNCMAPOPTIONS</td>
<td>Contains information about option values that you have set for a function mapping.</td>
</tr>
<tr>
<td>SYSCAT.FUNCMAPPINGS</td>
<td>Contains the function mappings between the federated database and the data source objects.</td>
</tr>
<tr>
<td>SYSCAT.INDEXCOLUSE</td>
<td>Contains columns that participate in an index.</td>
</tr>
<tr>
<td>SYSCAT.INDEXES</td>
<td>Contains index specifications for data source objects.</td>
</tr>
<tr>
<td>SYSCAT.INDEXOPTIONS</td>
<td>Contains information about index options.</td>
</tr>
<tr>
<td>SYSCAT.KEYCOLUSE</td>
<td>Contains columns that participate in a key defined by a unique key, primary key, or foreign key constraint.</td>
</tr>
<tr>
<td>SYSCAT.NICKNAMES</td>
<td>Contains information about nicknames that you created.</td>
</tr>
</tbody>
</table>
### Catalog views typically used with a federated system (continued)

<table>
<thead>
<tr>
<th>Catalog views</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSCAT.REFERENCES</td>
<td>Contains information about referential constraints that you defined.</td>
</tr>
<tr>
<td>SYSCAT.ROUTINES</td>
<td>Contains local DB2 user-defined functions, or function templates. Function templates are used to map to a data source function.</td>
</tr>
<tr>
<td>SYSCAT.REVTYPEMAPPINGS</td>
<td>This view is not used. All data type mappings are recorded in the SYSCAT.TYPEMAPPINGS view.</td>
</tr>
<tr>
<td>SYSCAT.ROUTINEOPTIONS</td>
<td>Contains information about federated routine option values.</td>
</tr>
<tr>
<td>SYSCAT.ROUTINEPARMOPTIONS</td>
<td>Contains information about federated routine parameter option values.</td>
</tr>
<tr>
<td>SYSCAT.ROUTINEPARMS</td>
<td>Contains a parameter or the result of a routine defined in SYSCAT.ROUTINES.</td>
</tr>
<tr>
<td>SYSCAT.ROUTINESFEDERATED</td>
<td>Contains information about federated routines that you defined.</td>
</tr>
<tr>
<td>SYSCAT.SERVERS</td>
<td>Contains server definitions that you create for data source servers.</td>
</tr>
<tr>
<td>SYSCAT.TABCONST</td>
<td>Each row represents a table and nickname constraints of type CHECK, UNIQUE, PRIMARY KEY, or FOREIGN KEY.</td>
</tr>
<tr>
<td>SYSCAT.TABLES</td>
<td>Contains information about each local DB2 table, federated view, and nickname that you create.</td>
</tr>
<tr>
<td>SYSCAT.TYPEMAPPINGS</td>
<td>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.</td>
</tr>
<tr>
<td>SYSCAT.USEROPTIONS</td>
<td>Contains user authorization information that you set when you create user mappings between the federated database and the data source servers.</td>
</tr>
<tr>
<td>SYSCAT.VIEWS</td>
<td>Contains information about local federated views that you create.</td>
</tr>
<tr>
<td>SYSCAT.WRAPOPTIONS</td>
<td>Contains information about option values that you have set for a wrapper.</td>
</tr>
<tr>
<td>SYSCAT.WRAPERS</td>
<td>Contains the name of the wrapper and library file for each data source that you create a wrapper for.</td>
</tr>
</tbody>
</table>

The following table lists the SYSSTAT views which contain federated information. These are read-write views that contain statistics you can update.
### Table 166. Federated updatable global catalog views

<table>
<thead>
<tr>
<th>Catalog views</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSSTAT.COLUMNS</td>
<td>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.</td>
</tr>
<tr>
<td>SYSSTAT.INDEXES</td>
<td>Contains statistical information about each index specification for data source objects.</td>
</tr>
<tr>
<td>SYSSTAT.ROUTINES</td>
<td>Contains statistical information about each user-defined function. Does not include built-in functions. Statistics are not recorded for inherited columns of typed tables.</td>
</tr>
<tr>
<td>SYSSTAT.TABLES</td>
<td>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.</td>
</tr>
</tbody>
</table>
Chapter 24. Function mapping options for federated systems

The federated server provides default mappings between DB2 functions and data source 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 or to change the default mapping, you create a function mapping.

When you create a function mapping, you specify the name of the data source function and must enable the mapped function. Then when you use the mapped function, the query optimizer compares the cost of running the function at the data source with the cost of running the function at the federated server.

Table 167. Options for function mappings

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISABLE</td>
<td>Enable or disable a default function mapping. Valid values are Y and N. The default is N.</td>
</tr>
<tr>
<td>REMOTE_NAME</td>
<td>The name of the data source function. The default is the local name.</td>
</tr>
</tbody>
</table>
Chapter 25. Valid server types in SQL statements

Server types indicate the kind of data source that the server definition represents.

Server types vary by vendor, purpose, and operating system. Supported values depend on the data source.

For most data sources, you must specify a valid server type in the CREATE SERVER statement.

*Table 168. Data sources and server types*

<table>
<thead>
<tr>
<th>Data source</th>
<th>Server type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BioRS</td>
<td>A server type is not required in the CREATE SERVER statement.</td>
</tr>
<tr>
<td>Blast</td>
<td>BLASTN, BLASTP, BLASTX, TBLASTN, TBLASTX</td>
</tr>
<tr>
<td>Entrez</td>
<td>NUCLEOTIDE, OMIM, PUBMED</td>
</tr>
<tr>
<td>Excel</td>
<td>A server type is not required in the CREATE SERVER statement.</td>
</tr>
<tr>
<td>HMMER</td>
<td>PFAM, SEARCH</td>
</tr>
<tr>
<td>IBM DB2 Universal Database™ for Linux, UNIX, and Windows</td>
<td>DB2/UDB</td>
</tr>
<tr>
<td>IBM DB2 Universal Database for System i and AS/400®</td>
<td>DB2/ISERIES</td>
</tr>
<tr>
<td>IBM DB2 Universal Database for z/OS</td>
<td>DB2/ZOS</td>
</tr>
<tr>
<td>IBM DB2 for VM</td>
<td>DB2/VM</td>
</tr>
<tr>
<td>Informix</td>
<td>INFORMIX</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>MSSQLSERVER (Required for data sources supported by the DataDirect Connect ODBC 4.2 (or later) driver or the Microsoft SQL Server ODBC 3.0 (or later) driver.)</td>
</tr>
<tr>
<td>ODBC</td>
<td>ODBC (Required for ODBC data sources that are supported by the ODBC 3.x driver.)</td>
</tr>
<tr>
<td>OLE DB</td>
<td>A server type is not required in the CREATE SERVER statement.</td>
</tr>
<tr>
<td>Oracle</td>
<td>ORACLE (Required for Oracle data sources supported by Oracle NET8 client software.)</td>
</tr>
<tr>
<td>Sybase (CTLIB)</td>
<td>SYBASE</td>
</tr>
<tr>
<td>Table-structured files</td>
<td>A server type is not required in the CREATE SERVER statement.</td>
</tr>
<tr>
<td>Teradata</td>
<td>TERADATA</td>
</tr>
<tr>
<td>Web services</td>
<td>A server type is not required in the CREATE SERVER statement.</td>
</tr>
<tr>
<td>WebSphere Business Integration</td>
<td>WBI</td>
</tr>
<tr>
<td>XML</td>
<td>A server type is not required in the CREATE SERVER statement.</td>
</tr>
</tbody>
</table>
Chapter 26. Data type mappings

Data types mappings for relational data sources include forward type mappings, reverse type mappings, and type mappings that are specific to Unicode. Each of the nonrelational data sources support specific data types.

Default forward data type mappings

The two kinds of mappings between data source data types and federated database data types are forward type mappings and reverse type mappings. In a forward type mapping, the mapping is from a remote type to a comparable local type.

You can override a default type mapping, or create a new type mapping with the CREATE TYPE MAPPING statement.

These mappings are valid with all the supported versions, unless otherwise noted.

For all default forward data types mapping from a data source to the federated database, the federated schema is SYSIBM.

The following tables show the default forward mappings between federated database data types and data source data types.

DB2 Database for Linux, UNIX, and Windows data sources

The following table lists the forward default data type mappings for DB2 Database for Linux, UNIX, and Windows data sources.

Table 169. DB2 Database for Linux, UNIX, and Windows forward default data type mappings (Not all columns shown)

<table>
<thead>
<tr>
<th>REMOTE_TYPE_NAME</th>
<th>REMOTE_LOWER_LEN</th>
<th>REMOTE_UPPER_LEN</th>
<th>REMOTE_LOWER_SCALE</th>
<th>REMOTE_UPPER_SCALE</th>
<th>REMOTE_BUILTIN</th>
<th>FEDERATED_TYPE_NAME</th>
<th>FEDERATED_LENGTH</th>
<th>FEDERATED_SCALE</th>
<th>FEDERATED_BUILTIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BIGINT</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>CHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>CHAR</td>
<td>-</td>
<td>0</td>
<td>Y</td>
</tr>
<tr>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DATE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DATE</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FLOAT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>GRAPHIC</td>
<td>-</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTEGER</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>LONGVAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>INTEGER</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>LONGVAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LONGVARG</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>REAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>REAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 169. DB2 Database for Linux, UNIX, and Windows forward default data type mappings (Not all columns shown) (continued)

<table>
<thead>
<tr>
<th>FEDERATED_TYPE_NAME</th>
<th>FEDERATED_LENGTH</th>
<th>FEDERATED_SCALE</th>
<th>FEDERATED_BIT_DATA</th>
<th>REMOTE_TYPE_NAME</th>
<th>REMOTE_LOWER_LEN</th>
<th>REMOTE_UPPER_LEN</th>
<th>REMOTE_LOWER_SCALE</th>
<th>REMOTE_UPPER_SCALE</th>
<th>REMOTE_BIT_DATA</th>
<th>REMOTE_DATA_OPERATORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIME</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARGRAPH</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARGRAPH</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARGRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

DB2 for System i data sources

The following table lists the forward default data type mappings for DB2 for System i data sources.

Table 170. DB2 for System i forward default data type mappings (Not all columns shown)

<table>
<thead>
<tr>
<th>Remote Type Name</th>
<th>Remote Lower Len</th>
<th>Remote Upper Len</th>
<th>Remote Lower Scale</th>
<th>Remote Upper Scale</th>
<th>Remote Bit Data</th>
<th>Remote Data Operators</th>
<th>Federated Type Name</th>
<th>Federated Length</th>
<th>Federated Scale</th>
<th>Federated Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHAR</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>CHAR</td>
<td>255</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>CHAR</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>0</td>
<td>Y</td>
</tr>
<tr>
<td>CHAR</td>
<td>255</td>
<td>32672</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>0</td>
<td>Y</td>
</tr>
<tr>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DATE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DATE</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FLOAT</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>REAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FLOAT</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>1</td>
<td>127</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>GRAPHIC</td>
<td>-</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>128</td>
<td>16336</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARGRAPHIC</td>
<td>-</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTEGER</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>TIME</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIME</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTMP</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>1</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>1</td>
<td>32672</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>0</td>
<td>Y</td>
</tr>
<tr>
<td>VARG</td>
<td>1</td>
<td>16336</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARGRAPHIC</td>
<td>-</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>1</td>
<td>16336</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARGRAPHIC</td>
<td>-</td>
<td>0</td>
<td>N</td>
</tr>
</tbody>
</table>

DB2 for VM and VSE data sources

The following table lists the forward default data type mappings for DB2 for VM and VSE data sources.
### Table 171. DB2 Server for VM and VSE forward default data type mappings (Not all columns shown)

<table>
<thead>
<tr>
<th>Remote Typename</th>
<th>Remote Lower Len</th>
<th>Remote Upper Len</th>
<th>Remote Lower Scale</th>
<th>Remote Upper Scale</th>
<th>Remote Bit Data</th>
<th>Remote Data Operators</th>
<th>Federated Typename</th>
<th>Federated Length</th>
<th>Federated Scale</th>
<th>Federated Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHAR</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>0</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>CHAR</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>CHAR</td>
<td>0</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DATE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DATE</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DBAHW</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DBAINT</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTEGER</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FLOAT</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>REAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FLOAT</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>1</td>
<td>127</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>GRAPHIC</td>
<td>0</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIME</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIME</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>1</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>0</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>1</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>VARCHAR</td>
<td>0</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>1</td>
<td>16336</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARGRAPHIC</td>
<td>0</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>VARGRAPH</td>
<td>1</td>
<td>16336</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARGRAPHIC</td>
<td>0</td>
<td>N</td>
<td>-</td>
</tr>
</tbody>
</table>

### DB2 for z/OS data sources

The following table lists the forward default data type mappings for DB2 for z/OS data sources.

### Table 172. DB2 for z/OS forward default data type mappings (Not all columns shown)

<table>
<thead>
<tr>
<th>Remote Typename</th>
<th>Remote Lower Len</th>
<th>Remote Upper Len</th>
<th>Remote Lower Scale</th>
<th>Remote Upper Scale</th>
<th>Remote Bit Data</th>
<th>Remote Data Operators</th>
<th>Federated Typename</th>
<th>Federated Length</th>
<th>Federated Scale</th>
<th>Federated Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHAR</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>0</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>CHAR</td>
<td>255</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>0</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>CHAR</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>CHAR</td>
<td>0</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>CHAR</td>
<td>255</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>VARCHAR</td>
<td>0</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DATE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DATE</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FLOAT</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>REAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FLOAT</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>1</td>
<td>127</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>GRAPHIC</td>
<td>0</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTEGER</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ROWID</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>VARCHAR</td>
<td>40</td>
<td>-</td>
<td>Y</td>
</tr>
</tbody>
</table>
### Informix data sources

The following table lists the forward default data type mappings for Informix data sources.

**Table 173. Informix forward default data type mappings (Not all columns shown)**

<table>
<thead>
<tr>
<th>Remote Type</th>
<th>Remote Lower Len</th>
<th>Remote Upper Len</th>
<th>Remote Lower Scale</th>
<th>Remote Upper Scale</th>
<th>Remote Bit Data</th>
<th>Remote Data Operators</th>
<th>Federated Type</th>
<th>Federated Length</th>
<th>Federated Scale</th>
<th>Federated Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOLEAN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BOOLEAN</td>
<td>CHAR</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BYTE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BYTE</td>
<td>CHARACTER</td>
<td>6</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>CHAR</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>CHARACTER</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CLOB</td>
<td>CHARACTER</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>DATE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DATE</td>
<td>DATE</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>DATETIME</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>-</td>
<td>DATETIME</td>
<td>DATE</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>DATETIME</td>
<td>6</td>
<td>10</td>
<td>6</td>
<td>10</td>
<td>-</td>
<td>DATETIME</td>
<td>TIME</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>1</td>
<td>31</td>
<td>0</td>
<td>31</td>
<td>-</td>
<td>DECIMAL</td>
<td>DECIMAL</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>32</td>
<td>130</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>DOUBLE</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>FLOAT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>FLOAT</td>
<td>DOUBLE</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTEGER</td>
<td>INTEGER</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>INTERVAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTERVAL</td>
<td>VARCHAR</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>INT8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INT8</td>
<td>BIGINT</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>NVARCHAR</td>
<td>1</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>NVARCHAR</td>
<td>BIGINT</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>MONEY</td>
<td>1</td>
<td>31</td>
<td>0</td>
<td>31</td>
<td>-</td>
<td>MONEY</td>
<td>DECIMAL</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>MONEY</td>
<td>32</td>
<td>32</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>MONEY</td>
<td>DOUBLE</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>NVARCHAR</td>
<td>1</td>
<td>32</td>
<td>32</td>
<td>-</td>
<td>-</td>
<td>NVARCHAR</td>
<td>DECIMAL</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>SERIAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SERIAL</td>
<td>INTEGER</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>SERIAL8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SERIAL8</td>
<td>BIGINT</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
</tbody>
</table>

**Table 172. DB2 for z/OS forward default data type mappings (Not all columns shown) (continued)**

<table>
<thead>
<tr>
<th>Remote Type</th>
<th>Remote Lower Len</th>
<th>Remote Upper Len</th>
<th>Remote Lower Scale</th>
<th>Remote Upper Scale</th>
<th>Remote Bit Data</th>
<th>Remote Data Operators</th>
<th>Federated Type</th>
<th>Federated Length</th>
<th>Federated Scale</th>
<th>Federated Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>SMALLINT</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIME</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIME</td>
<td>TIME</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>1</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>VARCHAR</td>
<td>1</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>1</td>
<td>32672</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>VARCHAR</td>
<td>VARCHAR</td>
<td>0</td>
<td>0</td>
<td>Y</td>
</tr>
<tr>
<td>VARG</td>
<td>1</td>
<td>16336</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARG</td>
<td>VARG</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>1</td>
<td>16336</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARGRAPHIC</td>
<td>VARGRAPHIC</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
</tbody>
</table>
Table 173. Informix forward default data type mappings (Not all columns shown) (continued)

<table>
<thead>
<tr>
<th>Remote Typename</th>
<th>Remote Lower Len</th>
<th>Remote Upper Len</th>
<th>Remote Lower Scale</th>
<th>Remote Upper Scale</th>
<th>Remote Bit Data</th>
<th>Remote Data Operators</th>
<th>Federated Typename</th>
<th>Federated Length</th>
<th>Federated Scale</th>
<th>Federated Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLFLOAT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>REAL</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>SMALLINT</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TEXT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>CLOB</td>
<td>2147483647</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>3267</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Notes:
- For the Informix DATETIME data type, the DB2 UNIX and Windows federated server uses the Informix high-level qualifier as the REMOTE_LENGTH and the Informix low-level qualifier as the REMOTE_SCALE.
- The Informix qualifiers are the "TU_" constants defined in the Informix Client SDK datatime.h file. The constants are:
  0 = YEAR
  2 = MONTH
  3 = DAY
  4 = HOUR
  5 = YEARHOUR

Microsoft SQL Server data sources

The following table lists the forward default data type mappings for Microsoft SQL Server data sources.

Table 174. Microsoft SQL Server forward default data type mappings

<table>
<thead>
<tr>
<th>Remote Typename</th>
<th>Remote Lower Len</th>
<th>Remote Upper Len</th>
<th>Remote Lower Scale</th>
<th>Remote Upper Scale</th>
<th>Remote Bit Data</th>
<th>Remote Data Operators</th>
<th>Federated Typename</th>
<th>Federated Length</th>
<th>Federated Scale</th>
<th>Federated Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>bigint</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BIGINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>binary</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>binary</td>
<td>255</td>
<td>8000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>bit</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>char</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>char</td>
<td>255</td>
<td>8000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>datetime</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>10</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>decimal</td>
<td>1</td>
<td>31</td>
<td>31</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>decimal</td>
<td>32</td>
<td>38</td>
<td>38</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>float</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>float</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>REAL</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>image</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BLOB</td>
<td>2147483647</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>int</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTEGER</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>money</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>19</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>nchar</td>
<td>1</td>
<td>127</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>nchar</td>
<td>128</td>
<td>4000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>numeric</td>
<td>1</td>
<td>31</td>
<td>31</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>numeric</td>
<td>32</td>
<td>38</td>
<td>38</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ntext</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CLOB</td>
<td>2147483647</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>nvarchar</td>
<td>1</td>
<td>4000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>real</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>REAL</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>smallint</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>smalldatetime</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>10</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
### Table 174. Microsoft SQL Server forward default data type mappings  (continued)

<table>
<thead>
<tr>
<th>Remote Typename</th>
<th>Remote Lower Len</th>
<th>Remote Lower Scale</th>
<th>Remote Upper Len</th>
<th>Remote Upper Scale</th>
<th>Remote Bit Data</th>
<th>Remote Data Operators</th>
<th>Federated Typename</th>
<th>Federated Length</th>
<th>Federated Scale</th>
<th>Federated Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>smallmoney</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>DECIMAL</td>
<td>10</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>SQL_BIGINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>BIGINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_BINARY</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>SQL_BINARY</td>
<td>255</td>
<td>8000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>SQL_BIT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>SMALLINT</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_CHAR</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>CHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>SQL_CHAR</td>
<td>255</td>
<td>8000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>SQL_DATE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>DATE</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_DECIMAL</td>
<td>1</td>
<td>31</td>
<td>0</td>
<td>31</td>
<td>-</td>
<td></td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_DECIMAL</td>
<td>32</td>
<td>38</td>
<td>0</td>
<td>38</td>
<td>-</td>
<td></td>
<td>DOUBLEX</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>DOUBLEX</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_FLOAT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>DOUBLEX</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_GUID</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>SQL_INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>INTEGER</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_LONGVARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>CLOB</td>
<td>2147483647</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>SQL_LONGVARBINARY</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>BLOB</td>
<td>-</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>SQL_NUMERIC</td>
<td>1</td>
<td>31</td>
<td>0</td>
<td>31</td>
<td>-</td>
<td></td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_NUMERIC</td>
<td>32</td>
<td>38</td>
<td>0</td>
<td>38</td>
<td>-</td>
<td></td>
<td>DOUBLEX</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_REAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>REAL</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>SMALLINT</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_TIME</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>TIME</td>
<td>3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>TIMESTAMP</td>
<td>10</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_TINYINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>SMALLINT</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_VARBINARY</td>
<td>1</td>
<td>8000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>SQL_VARCHAR</td>
<td>1</td>
<td>8000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>SQL_WCHAR</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>SQL_WCHAR</td>
<td>255</td>
<td>8800</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>SQL_WLONGVARCHAR</td>
<td>1073741823</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>CLOB</td>
<td>2147483647</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>SQL_WVARCHAR</td>
<td>1</td>
<td>16336</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>text</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>timestamp</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>VARCHAR</td>
<td>8</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>tinyint</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>SMALLINT</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>uniqueidentifier</td>
<td>1</td>
<td>4000</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td></td>
<td>VARCHAR</td>
<td>16</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>varbinary</td>
<td>1</td>
<td>8000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>varchar</td>
<td>1</td>
<td>8000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
</tbody>
</table>

**Note:**
1. This type mapping is valid only with Microsoft SQL Server Version 2000.

### ODBC data sources

The following table lists the forward default data type mappings for ODBC data sources.
Table 175. ODBC forward default data type mappings (Not all columns shown)

<table>
<thead>
<tr>
<th>Remote Typename</th>
<th>Remote Lower Len</th>
<th>Remote Upper Len</th>
<th>Remote Lower Scale</th>
<th>Remote Upper Scale</th>
<th>Remote Bit Data</th>
<th>Remote Data Operators</th>
<th>Federated Typename</th>
<th>Federated Length</th>
<th>Federated Scale</th>
<th>Federated Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL_BIGINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BIGINT</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_BINARY</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>SQL_BINARY</td>
<td>255</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>SQL_BIT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_CHAR</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>SQL_CHAR</td>
<td>255</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>SQL_DECIMAL</td>
<td>1</td>
<td>31</td>
<td>0</td>
<td>31</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_DECIMAL</td>
<td>32</td>
<td>38</td>
<td>0</td>
<td>38</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_FLOAT</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>FLOAT</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_FLOAT</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>FLOAT</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTEGER</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_LONGVARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CLOB</td>
<td>2147483647</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>SQL_LONGVARBINARY</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BLOB</td>
<td>2147483647</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>SQL_NUMERIC</td>
<td>1</td>
<td>31</td>
<td>0</td>
<td>31</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_NUMERIC</td>
<td>32</td>
<td>32</td>
<td>0</td>
<td>32</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_REAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>REAL</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_TYPE_DATE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DATE</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_TYPE_TIME</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIME</td>
<td>3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_TYPE_TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>10</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_TINYINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL_VARBINARY</td>
<td>1</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>SQL_VARCHAR</td>
<td>1</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>SQL_WCHAR</td>
<td>1</td>
<td>127</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>SQL_WCHAR</td>
<td>128</td>
<td>16336</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>SQL_WVARCHAR</td>
<td>1</td>
<td>16336</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>SQL_WLONGVARCHAR</td>
<td>-</td>
<td>1073741823</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CLOB</td>
<td>2147483647</td>
<td>-</td>
<td>N</td>
</tr>
</tbody>
</table>

**Oracle NET8 data sources**

The following table lists the forward default data type mappings for Oracle NET8 data sources.

Table 176. Oracle NET8 forward default data type mappings

<table>
<thead>
<tr>
<th>Remote Typename</th>
<th>Remote Lower Len</th>
<th>Remote Upper Len</th>
<th>Remote Lower Scale</th>
<th>Remote Upper Scale</th>
<th>Remote Bit Data</th>
<th>Remote Data Operators</th>
<th>Federated Typename</th>
<th>Federated Length</th>
<th>Federated Scale</th>
<th>Federated Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOB</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>\0</td>
<td>BLOB</td>
<td>2147483647</td>
<td>0</td>
<td>Y</td>
</tr>
<tr>
<td>CHAR</td>
<td>1</td>
<td>254</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>\0</td>
<td>CHAR</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>CHAR</td>
<td>255</td>
<td>2000</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>\0</td>
<td>VARCHAR</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>CLOB</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>\0</td>
<td>CLOB</td>
<td>2147483647</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>DATE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>\0</td>
<td>TIMESTAMP</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
</tbody>
</table>
Table 176. Oracle NET8 forward default data type mappings (continued)

<table>
<thead>
<tr>
<th>Remote Typename</th>
<th>Remote Lower Len</th>
<th>Remote Upper Len</th>
<th>Remote Lower Scale</th>
<th>Remote Upper Scale</th>
<th>Remote Bit Data</th>
<th>Remote Data Operators</th>
<th>Federated Typename</th>
<th>Federated Length</th>
<th>Federated Scale</th>
<th>Federated Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOAT</td>
<td>1</td>
<td>126</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>\0</td>
<td>DOUBLE</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>LONG</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>\0</td>
<td>CLOB</td>
<td>2147483647</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>LONG RAW</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>\0</td>
<td>BLOB</td>
<td>2147483647</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>NUMBER</td>
<td>10</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>\0</td>
<td>BIGINT</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>NUMBER</td>
<td>1</td>
<td>38</td>
<td>-84</td>
<td>127</td>
<td>-</td>
<td>\0</td>
<td>DOUBLE</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>NUMBER</td>
<td>1</td>
<td>31</td>
<td>0</td>
<td>31</td>
<td>-</td>
<td>\0</td>
<td>DECIMAL</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>NUMBER</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>\0</td>
<td>SMALLINT</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>NUMBER</td>
<td>5</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>\0</td>
<td>INTEGER</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>NUMBER</td>
<td>-</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>\0</td>
<td>DECIMAL</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>RAW</td>
<td>1</td>
<td>2000</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>\0</td>
<td>VARCHAR</td>
<td>0</td>
<td>0</td>
<td>Y</td>
</tr>
<tr>
<td>ROWID</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NULL</td>
<td>-</td>
<td>\0</td>
<td>CHAR</td>
<td>18</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>10</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR2</td>
<td>1</td>
<td>4000</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>\0</td>
<td>VARCHAR</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
</tbody>
</table>

Note:
1. This type mapping is valid only for Oracle 9i (or later) client and server configurations.

Sybase data sources

The following table lists the forward default data type mappings for Sybase data sources.

Table 177. Sybase CTLIB forward default data type mappings

<table>
<thead>
<tr>
<th>Remote Typename</th>
<th>Remote Lower Len</th>
<th>Remote Upper Len</th>
<th>Remote Lower Scale</th>
<th>Remote Upper Scale</th>
<th>Remote Bit Data</th>
<th>Remote Data Operators</th>
<th>Federated Typename</th>
<th>Federated Length</th>
<th>Federated Scale</th>
<th>Federated Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>binary</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>binary</td>
<td>255</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>bit</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>char</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>char</td>
<td>255</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>char null (see varchar)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>datetim</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>datetimn</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>decimal</td>
<td>1</td>
<td>31</td>
<td>0</td>
<td>31</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>decimal</td>
<td>32</td>
<td>38</td>
<td>0</td>
<td>38</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>decimal</td>
<td>1</td>
<td>31</td>
<td>0</td>
<td>31</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>decimal</td>
<td>32</td>
<td>38</td>
<td>0</td>
<td>38</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>float</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>REAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>float</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>floatn</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>REAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>floatn</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>image</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>int</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>intn</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>money</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>19</td>
<td>4</td>
<td>-</td>
</tr>
</tbody>
</table>
### Table 177. Sybase CTLIB forward default data type mappings (continued)

<table>
<thead>
<tr>
<th>Remote Typename</th>
<th>Remote Lower Len</th>
<th>Remote Upper Len</th>
<th>Remote Lower Scale</th>
<th>Remote Upper Scale</th>
<th>Remote Bit Data</th>
<th>Remote Data Operators</th>
<th>Federated Typename</th>
<th>Federated Length</th>
<th>Federated Scale</th>
<th>Federated Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>moneyn</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>19</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>nchar</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>nchar</td>
<td>255</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
</tbody>
</table>

**Note:**
1. Valid for non-Unicode federated databases.

### Teradata data sources

The following table lists the forward default data type mappings for Teradata data sources.

### Table 178. Teradata forward default data type mappings (Not all columns shown)

<table>
<thead>
<tr>
<th>Remote Typename</th>
<th>Remote Lower Len</th>
<th>Remote Upper Len</th>
<th>Remote Lower Scale</th>
<th>Remote Upper Scale</th>
<th>Remote Bit Data</th>
<th>Remote Data Operators</th>
<th>Federated Typename</th>
<th>Federated Length</th>
<th>Federated Scale</th>
<th>Federated Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYTE</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>BYTE</td>
<td>255</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>BYTE</td>
<td>32673</td>
<td>64000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BYTEINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHAR</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHAR</td>
<td>255</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHAR</td>
<td>32673</td>
<td>64000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DATE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DATE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 178. Teradata forward default data type mappings (Not all columns shown)  (continued)

<table>
<thead>
<tr>
<th>Remote Typename</th>
<th>Remote Lower Len</th>
<th>Remote Upper Len</th>
<th>Remote Lower Scale</th>
<th>Remote Upper Scale</th>
<th>Remote Bit Data</th>
<th>Remote Data Operators</th>
<th>Federated Typename</th>
<th>Federated Length</th>
<th>Federated Scale</th>
<th>Federated Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECIMAL</td>
<td>1</td>
<td>18</td>
<td>0</td>
<td>18</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FLOAT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>1</td>
<td>127</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>GRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>128</td>
<td>16336</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>INTERVAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>1</td>
<td>18</td>
<td>0</td>
<td>18</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>REAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIME</td>
<td>0</td>
<td>21</td>
<td>0</td>
<td>21</td>
<td>-</td>
<td>-</td>
<td>TIME</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARBYTE</td>
<td>1</td>
<td>32762</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>VARBYTE</td>
<td>32763</td>
<td>64000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>1</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>32673</td>
<td>64000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>1</td>
<td>16336</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARGRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>16337</td>
<td>32000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Default reverse data type mappings

For most data sources, the default type mappings are in the wrappers.

The two kinds of mappings between data source data types and federated database data types are forward type mappings and reverse type mappings. In a forward type mapping, the mapping is from a remote type to a comparable local type. The other type of mapping is a reverse type mapping, which is used with transparent DDL to create or modify remote tables.

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

When you define a remote table or view to the federated database, the definition includes a reverse type mapping. The mapping is from a local federated database data type for each column, and the corresponding remote data type. For example, there is a default reverse type mapping in which the local type REAL points to the Informix type SMALLFLOAT.

Federated databases do not support mappings for LONG VARCHAR, LONG VARGRAPHIC, and user-defined types.

When you use the CREATE TABLE statement to create a remote table, you specify the local data types you want to include in the remote table. These default reverse type mappings will assign corresponding remote types to these columns. For example, suppose that you use the CREATE TABLE statement to define an...
Informix table with a column C2. You specify BIGINT as the data type for C2 in the statement. The default reverse type mapping of BIGINT depends on which version of Informix you are creating the table on. The mapping for C2 in the Informix table will be to DECIMAL in Informix Version 8 and to INT8 in Informix Version 9.

You can override a default reverse type mapping, or create a new reverse type mapping with the CREATE TYPE MAPPING statement.

The following tables show the default reverse mappings between federated database local data types and remote data source data types.

These mappings are valid with all the supported versions, unless otherwise noted.

**DB2 Database for Linux, UNIX, and Windows data sources**

The following table lists the reverse default data type mappings for DB2 Database for Linux, UNIX, and Windows data sources.

<table>
<thead>
<tr>
<th>Federated Typename</th>
<th>Federated Lower Len</th>
<th>Federated Upper Len</th>
<th>Federated Lower Scale</th>
<th>Federated Upper Scale</th>
<th>Federated Bit Data Operators</th>
<th>Remote Typename</th>
<th>Remote Length</th>
<th>Remote Scale</th>
<th>Federated Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BIGINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DATE</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DATE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FLOAT</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>GRAPHIC</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>INTEGER</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>REAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>REAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIME</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIME</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>VARGRAPH</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARGRAPH</td>
<td>-</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARGRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**DB2 for System i data sources**

The following table lists the reverse default data type mappings for DB2 for System i data sources.
### DB2 for VM and VSE data sources

The following table lists the reverse default data type mappings for DB2 for VM and VSE data sources.

#### Table 181. DB2 for VM and VSE reverse default data type mappings (Not all columns shown)

<table>
<thead>
<tr>
<th>Federated Typename</th>
<th>Federated Lower Len</th>
<th>Federated Upper Len</th>
<th>Federated Lower Scale</th>
<th>Federated Upper Scale</th>
<th>Federated Bit Data</th>
<th>Federated Data Operations</th>
<th>Remote Typename</th>
<th>Remote Length</th>
<th>Remote Scale</th>
<th>Remote Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHARACTER</td>
<td>-</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHARACTER</td>
<td>-</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DATE</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DATE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>NUMERIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>FLOAT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>GRAPHIC</td>
<td>-</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>INTEGER</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>REAL</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>FLOAT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIME</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIME</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>VARGRAPH</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARGRAPH</td>
<td>-</td>
<td>N</td>
<td>-</td>
</tr>
</tbody>
</table>
DB2 for z/OS data sources

The following table lists the reverse default data type mappings for DB2 for z/OS data sources.

<table>
<thead>
<tr>
<th>Federated Typename</th>
<th>Federated Lower Len</th>
<th>Federated Upper Len</th>
<th>Federated Lower Scale</th>
<th>Federated Upper Scale</th>
<th>Federated Bit Data</th>
<th>Federated Data Operators</th>
<th>Remote Typename</th>
<th>Remote Length</th>
<th>Remote Scale</th>
<th>Remote Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DATE</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DATE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FLOAT</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>GRAPHIC</td>
<td>-</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>INTEGER</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>REAL</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>REAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIME</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIME</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARGRAPHIC</td>
<td>-</td>
<td>N</td>
<td>-</td>
</tr>
</tbody>
</table>

Informix data sources

The following table lists the reverse default data type mappings for Informix data sources.

<table>
<thead>
<tr>
<th>Federated Typename</th>
<th>Federated Lower Len</th>
<th>Federated Upper Len</th>
<th>Federated Lower Scale</th>
<th>Federated Upper Scale</th>
<th>Federated Bit Data</th>
<th>Federated Data Operators</th>
<th>Remote Typename</th>
<th>Remote Length</th>
<th>Remote Scale</th>
<th>Remote Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT  1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>19</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BIGINT  2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INT8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BLOB</td>
<td>1</td>
<td>2147483647</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BYTE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>-</td>
<td>CHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>BYTE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CLOB</td>
<td>1</td>
<td>2147483647</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TEXT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DATE</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DATE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>FLOAT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>INTEGER</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>REAL</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLFLOAT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIME</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DATETIME</td>
<td>6</td>
<td>10</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DATETIME</td>
<td>0</td>
<td>15</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 183. Informix reverse default data type mappings (continued)

<table>
<thead>
<tr>
<th>Federated Typename</th>
<th>Federated Lower Len</th>
<th>Federated Upper Len</th>
<th>Federated Lower Scale</th>
<th>Federated Upper Scale</th>
<th>Federated Bit Data</th>
<th>Federated Data Operators</th>
<th>Remote Typename</th>
<th>Remote Length</th>
<th>Remote Scale</th>
<th>Remote Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR 1</td>
<td>1</td>
<td>254</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR 1</td>
<td>255</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>TEXT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>BYTE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARCHAR 2</td>
<td>255</td>
<td>2048</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>LVARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR 2</td>
<td>2049</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>TEXT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Note:
1. This type mapping is valid only with Informix server Version 8 (or lower).
2. This type mapping is valid only with Informix server Version 9 (or higher).

For the Informix DATETIME data type, the federated server uses the Informix high-level qualifier as the REMOTE_LENGTH and the Informix low-level qualifier as the REMOTE_SCALE.

The Informix qualifiers are the "TU_" constants defined in the Informix Client SDK datetime.h file. The constants are:

| 0 = YEAR          | 8 = MINUTE          | 13 = FRACTION(3)    |
| 2 = MONTH         | 10 = SECOND         | 14 = FRACTION(4)    |
| 4 = DAY           | 11 = FRACTION(1)    | 15 = FRACTION(5)    |
| 6 = HOUR          | 12 = FRACTION(2)    |                      |

Microsoft SQL Server data sources

The following table lists the reverse default data type mappings for Microsoft SQL Server data sources.

Table 184. Microsoft SQL Server reverse default data type mappings (Not all columns shown)

<table>
<thead>
<tr>
<th>Federated Typename</th>
<th>Federated Lower Len</th>
<th>Federated Upper Len</th>
<th>Federated Lower Scale</th>
<th>Federated Upper Scale</th>
<th>Federated Bit Data</th>
<th>Federated Data Operators</th>
<th>Remote Typename</th>
<th>Remote Length</th>
<th>Remote Scale</th>
<th>Remote Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>bigint</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>image</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>binary</td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>char</td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>text</td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DATE</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>datetime</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>decimal</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>float</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>int</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>smallint</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>REAL</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>real</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIME</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>datetime</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>datetime</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>1</td>
<td>8000</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>varchar</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR 8001</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>text</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>varbinary</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>8001</td>
<td>32672</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>image</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Note:
1. This type mapping is valid only with Microsoft SQL Server Version 2000.
**ODBC data sources**

The following table lists the reverse default data type mappings for ODBC data sources.

<table>
<thead>
<tr>
<th>Typename</th>
<th>Lower Len</th>
<th>Upper Len</th>
<th>Lower Scale</th>
<th>Upper Scale</th>
<th>Bit Data</th>
<th>Lower Bit Data</th>
<th>Upper Bit Data</th>
<th>Operators</th>
<th>Length</th>
<th>Scale</th>
<th>Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>SQL_BIGINT</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SQL_LONGVARBINARY</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>SQL_BINARY</td>
<td>-</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>-</td>
<td>SQL_CHAR</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SQL_LONGVARCHAR</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SQL_TYPE_DATE</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>DBCLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SQL_WLONGVARCHAR</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SQL_DECIMAL</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>DOUBLE</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SQL_DOUBLE</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SQL_FLOAT</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SQL_WCHAR</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SQL_INTEGER</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>REAL</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SQL_REAL</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SQL_SMALLINT</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SQL_TYPE_TIME</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SQL_TYPE_TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>SQL_VARBINARY</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>-</td>
<td>SQL_VARCHAR</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>SQL_WVARCHAR</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

**Oracle NET8 data sources**

The following table lists the reverse default data type mappings for Oracle NET8 data sources.

<table>
<thead>
<tr>
<th>Typename</th>
<th>Lower Len</th>
<th>Upper Len</th>
<th>Lower Scale</th>
<th>Upper Scale</th>
<th>Bit Data</th>
<th>Lower Bit Data</th>
<th>Upper Bit Data</th>
<th>Operators</th>
<th>Length</th>
<th>Scale</th>
<th>Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>\0</td>
<td>\0</td>
<td>NUMBER</td>
<td>19</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>BLOB</td>
<td>0</td>
<td>2147483647</td>
<td>0</td>
<td>0</td>
<td>Y</td>
<td>\0</td>
<td>\0</td>
<td>BLOB</td>
<td>0</td>
<td>0</td>
<td>Y</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>1</td>
<td>254</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>\0</td>
<td>\0</td>
<td>CHAR</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>1</td>
<td>254</td>
<td>0</td>
<td>0</td>
<td>Y</td>
<td>\0</td>
<td>\0</td>
<td>RAW</td>
<td>0</td>
<td>0</td>
<td>Y</td>
</tr>
<tr>
<td>CLOB</td>
<td>0</td>
<td>2147483647</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>\0</td>
<td>\0</td>
<td>CLOB</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>DATE</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>\0</td>
<td>\0</td>
<td>DATE</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>\0</td>
<td>\0</td>
<td>NUMBER</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>\0</td>
<td>\0</td>
<td>FLOAT</td>
<td>126</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>FLOAT</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>\0</td>
<td>\0</td>
<td>FLOAT</td>
<td>126</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>INTEGER</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>\0</td>
<td>\0</td>
<td>NUMBER</td>
<td>10</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>REAL</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>\0</td>
<td>\0</td>
<td>FLOAT</td>
<td>63</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>\0</td>
<td>\0</td>
<td>NUMBER</td>
<td>5</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>TIME</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>\0</td>
<td>\0</td>
<td>DATE</td>
<td>0</td>
<td>0</td>
<td>N</td>
</tr>
</tbody>
</table>
Table 186. Oracle NET8 reverse default data type mappings (continued)

<table>
<thead>
<tr>
<th>Federated Typename</th>
<th>Federated Lower Len</th>
<th>Federated Upper Len</th>
<th>Federated Lower Scale</th>
<th>Federated Upper Scale</th>
<th>Federated Bit Data Operators</th>
<th>Remote Typename</th>
<th>Remote Length</th>
<th>Remote Scale</th>
<th>Remote Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMESTAMP ¹</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>\0</td>
<td>DATE</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TIMESTAMP ²</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>\0</td>
<td>TIMESTAMP</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>1</td>
<td>4000</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>\0</td>
<td>VARCHAR2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>1</td>
<td>2000</td>
<td>0</td>
<td>0</td>
<td>Y</td>
<td>\0</td>
<td>RAW</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Note:
1. This type mapping is valid only with Oracle Version 8.
2. This type mapping is valid only with Oracle Version 9 and Version 10.

Sybase data sources

The following table lists the reverse default data type mappings for Sybase data sources.

Table 187. Sybase CTLIB default reverse data type mappings

<table>
<thead>
<tr>
<th>Federated Typename</th>
<th>Federated Lower Len</th>
<th>Federated Upper Len</th>
<th>Federated Lower Scale</th>
<th>Federated Upper Scale</th>
<th>Federated Bit Data Operators</th>
<th>Remote Typename</th>
<th>Remote Length</th>
<th>Remote Scale</th>
<th>Remote Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>decimal</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>BLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>image</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>char</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>binary</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CLOB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>text</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DATE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>datetime</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>decimal</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>float</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>integer</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>REAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>real</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>smallint</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIME</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>datetime</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>datetime</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR ¹</td>
<td>1</td>
<td>255</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>\0</td>
<td>varchar</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR ¹</td>
<td>256</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>\0</td>
<td>text</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR ²</td>
<td>1</td>
<td>16384</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>\0</td>
<td>varchar</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR ²</td>
<td>16385</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>\0</td>
<td>text</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR ¹</td>
<td>1</td>
<td>255</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>\0</td>
<td>varbinary</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR ¹</td>
<td>256</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>\0</td>
<td>image</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR ²</td>
<td>1</td>
<td>16384</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>\0</td>
<td>varbinary</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR ²</td>
<td>16385</td>
<td>32672</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>\0</td>
<td>image</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Note:
1. This type mapping is valid only for CTLIB with Sybase server version 12.0 (or earlier).
2. This type mapping is valid only for CTLIB with Sybase server version 12.5 (or later).

Teradata data sources

The following table lists the reverse default data type mappings for Teradata data sources.
Table 188. Teradata reverse default data type mappings (Not all columns shown)

<table>
<thead>
<tr>
<th>Federated Typename</th>
<th>Federated Lower Len</th>
<th>Federated Upper Len</th>
<th>Federated Lower Scale</th>
<th>Federated Upper Scale</th>
<th>Federated Bit Data</th>
<th>Federated Data Operators</th>
<th>Remote Typename</th>
<th>Remote Length</th>
<th>Remote Scale</th>
<th>Remote Bit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOB ¹</td>
<td>1</td>
<td>64000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARBYTE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BYTE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CLOB ²</td>
<td>1</td>
<td>64000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DATE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DATE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DBCLOB ³</td>
<td>1</td>
<td>64000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARGRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>1</td>
<td>18</td>
<td>0</td>
<td>18</td>
<td>-</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>19</td>
<td>31</td>
<td>0</td>
<td>31</td>
<td>-</td>
<td>-</td>
<td>FLOAT</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>FLOAT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>GRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>INTEGER</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>REAL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>FLOAT</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SMALLINT</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIME</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIME</td>
<td>15</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>26</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>VARGRAPHIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Note:
1. The Teradata VARBYTE data type can contain only the specified length (1 to 64000) of a BLOB data type.
2. The Teradata VARCHAR data type can contain only the specified length (1 to 64000) of a CLOB data type.
3. The Teradata VARGRAPHIC data type can contain only the specified length (1 to 32000) of a DBCLOB data type.

Unicode default data type mappings

Specific data sources support forward data type mappings and reverse data type mappings for Unicode databases.

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 189. Unicode default forward data type mappings for the Microsoft SQL Server wrapper

<table>
<thead>
<tr>
<th>UTF-8 Data type</th>
<th>Microsoft SQL Server Data type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR</td>
<td>CHAR</td>
<td>1 to 254 bytes</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>CHAR</td>
<td>255 to 8000 bytes</td>
</tr>
<tr>
<td></td>
<td>VARCHAR</td>
<td>1 to 8000 bytes</td>
</tr>
<tr>
<td>CLOB</td>
<td>TEXT</td>
<td>-</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>NCHAR</td>
<td>1 to 127 characters</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>NCHAR</td>
<td>128 to 16336 characters</td>
</tr>
<tr>
<td></td>
<td>NVARCHAR</td>
<td>1 to 16336 characters</td>
</tr>
</tbody>
</table>
Table 189. Unicode default forward data type mappings for the Microsoft SQL Server wrapper (continued)

<table>
<thead>
<tr>
<th>UTF-8</th>
<th>Microsoft SQL Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>Data type</td>
</tr>
<tr>
<td>DBCLOB</td>
<td>NTEXT</td>
</tr>
</tbody>
</table>

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 190. Unicode default reverse data type mappings for the Microsoft SQL Server wrapper

<table>
<thead>
<tr>
<th>UTF-8</th>
<th>Microsoft SQL Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>Length</td>
</tr>
<tr>
<td>CHAR</td>
<td>1 to 254 bytes</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>1 to 32672 bytes</td>
</tr>
<tr>
<td>CLOB</td>
<td>1 to 2 147 483 647 bytes</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>1 to 127 characters</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>1 to 16336 characters</td>
</tr>
<tr>
<td>DBCLOB</td>
<td>1 to 1 073 741 823 characters</td>
</tr>
</tbody>
</table>

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 191. Unicode default forward data type mappings for the NET8 wrapper

<table>
<thead>
<tr>
<th>UTF-8</th>
<th>Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>Data type</td>
</tr>
<tr>
<td>CHAR</td>
<td>CHAR</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>CHAR</td>
</tr>
<tr>
<td></td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>DBCLOB</td>
<td>NCLOB</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>NCHAR</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>NCHAR</td>
</tr>
<tr>
<td></td>
<td>NVARCHAR2</td>
</tr>
</tbody>
</table>

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 192. Unicode default reverse data type mappings for the NET8 wrapper

<table>
<thead>
<tr>
<th>UTF-8</th>
<th>Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>Length</td>
</tr>
<tr>
<td>CHAR</td>
<td>1 to 254 bytes</td>
</tr>
</tbody>
</table>
**Table 192. Unicode default reverse data type mappings for the NET8 wrapper (continued)**

<table>
<thead>
<tr>
<th>UTF-8</th>
<th>Length</th>
<th>Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR</td>
<td>1 to 4000 bytes</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>CLOB</td>
<td>1 to 2 147 483 647 bytes</td>
<td>CLOB</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>1 to 127 characters</td>
<td>NCHAR</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>1 to 2000 characters</td>
<td>NVARCHAR2</td>
</tr>
<tr>
<td>DBCLOB</td>
<td>1 to 1 073 741 823 characters</td>
<td>NCLOB</td>
</tr>
</tbody>
</table>

**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 193. Unicode default forward data type mappings for the ODBC wrapper**

<table>
<thead>
<tr>
<th>UTF-8</th>
<th>ODBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>Data type</td>
</tr>
<tr>
<td>CHAR</td>
<td>SQL_CHAR</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>SQL_CHAR</td>
</tr>
<tr>
<td></td>
<td>SQL_VARCHAR</td>
</tr>
<tr>
<td>CLOB</td>
<td>SQL_LONGVARCHAR</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>SQL_WCHAR</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>SQL_WVARCHAR</td>
</tr>
<tr>
<td>DBCLOB</td>
<td>SQL_WLONGVARCHAR</td>
</tr>
</tbody>
</table>

**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 194. Unicode default reverse data type mappings for the ODBC wrapper**

<table>
<thead>
<tr>
<th>UTF-8</th>
<th>ODBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>Length</td>
</tr>
<tr>
<td>CHAR</td>
<td>1 to 254 bytes</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>1 to 32672 bytes</td>
</tr>
<tr>
<td>CLOB</td>
<td>1 to 2 147 483 647 bytes</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>1 to 127 characters</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>1 to 16336 characters</td>
</tr>
<tr>
<td>DBCLOB</td>
<td>1 to 1 073 741 823 characters</td>
</tr>
</tbody>
</table>

**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.
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 196. Unicode default reverse data type mappings for the Sybase CTLIB wrapper

<table>
<thead>
<tr>
<th>UTF-8</th>
<th>Sybase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>Data type</td>
</tr>
<tr>
<td>CHAR</td>
<td>char</td>
</tr>
<tr>
<td></td>
<td>nchar</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>char</td>
</tr>
<tr>
<td></td>
<td>varchar</td>
</tr>
<tr>
<td></td>
<td>nchar</td>
</tr>
<tr>
<td></td>
<td>nvarchar</td>
</tr>
<tr>
<td>CLOB</td>
<td>text</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>unichar</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>unichar</td>
</tr>
<tr>
<td></td>
<td>univarchar</td>
</tr>
</tbody>
</table>

Data types supported for nonrelational data sources

For most of the nonrelational data sources, you must specify the column information, including data type, when you create the nicknames to access the data source.

Some of the nonrelational wrappers create all of the columns required to access a data source. These are called fixed columns. Other wrappers let you specify some or all of the data types for the columns in the CREATE NICKNAME statement.

The following sections list the wrappers that you can specify the data types for, and the data types that the wrapper supports.

Data types supported by the BioRS wrapper

The following table lists the DB2 data types that the BioRS wrapper supports.

Table 197. BioRS data types that map to DB2 data types

<table>
<thead>
<tr>
<th>BioRS data types</th>
<th>DB2 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTHOR</td>
<td>CHARACTER, CLOB, VARCHAR</td>
</tr>
</tbody>
</table>
Table 197. BioRS data types that map to DB2 data types (continued)

<table>
<thead>
<tr>
<th>BioRS data types</th>
<th>DB2 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>CHARACTER, CLOB, VARCHAR</td>
</tr>
<tr>
<td>NUMBER</td>
<td>CHARACTER, CLOB, VARCHAR</td>
</tr>
<tr>
<td>REFERENCE</td>
<td>CHARACTER, CLOB, VARCHAR</td>
</tr>
<tr>
<td>TEXT</td>
<td>CHARACTER, CLOB, VARCHAR</td>
</tr>
</tbody>
</table>

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 the BLAST wrapper supports.

Table 198. BLAST data types that map to DB2 data types

<table>
<thead>
<tr>
<th>BLAST data types</th>
<th>DB2 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>definition line</td>
<td>CLOB (maximum length is 5 megabytes)</td>
</tr>
<tr>
<td>definition line</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>definition line</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>definition line</td>
<td>FLOAT</td>
</tr>
<tr>
<td>definition line</td>
<td>INTEGER</td>
</tr>
<tr>
<td>definition line</td>
<td>VARCHAR</td>
</tr>
</tbody>
</table>

Data types supported by the Entrez wrapper

The following table lists the DB2 data types that the Entrez wrapper supports.

Table 199. Entrez data types that map to DB2 data types

<table>
<thead>
<tr>
<th>Entrez data types</th>
<th>DB2 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>character</td>
<td>CLOB (maximum length is 5 megabytes)</td>
</tr>
<tr>
<td>date</td>
<td>DATE</td>
</tr>
<tr>
<td>number</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>number</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>number</td>
<td>FLOAT</td>
</tr>
<tr>
<td>integer</td>
<td>INTEGER</td>
</tr>
<tr>
<td>number</td>
<td>REAL</td>
</tr>
<tr>
<td>integer</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>time</td>
<td>TIMESTAMP</td>
</tr>
</tbody>
</table>
Table 199. Entrez data types that map to DB2 data types (continued)

<table>
<thead>
<tr>
<th>Entrez data types</th>
<th>DB2 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>VARCHAR</td>
</tr>
</tbody>
</table>

Data types supported by the Excel wrapper

The following table lists the DB2 data types that the Excel wrapper supports.

Table 200. Excel data types that map to DB2 data types

<table>
<thead>
<tr>
<th>Excel data types</th>
<th>DB2 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>date</td>
<td>DATE</td>
</tr>
<tr>
<td>number</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>number</td>
<td>FLOAT</td>
</tr>
<tr>
<td>integer</td>
<td>INTEGER</td>
</tr>
<tr>
<td>character</td>
<td>VARCHAR</td>
</tr>
</tbody>
</table>

Data types supported by the HMMER wrapper

The following table lists the DB2 data types that the HMMER wrapper supports.

Table 201. HMMER data types that map to DB2 data types

<table>
<thead>
<tr>
<th>HMMER data types</th>
<th>DB2 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>character</td>
<td>CLOB (maximum length is 5 megabytes)</td>
</tr>
<tr>
<td>character</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>character</td>
<td>FLOAT</td>
</tr>
<tr>
<td>character</td>
<td>INTEGER</td>
</tr>
<tr>
<td>character</td>
<td>VARCHAR</td>
</tr>
</tbody>
</table>

Data types supported by the Script wrapper

The following table lists the DB2 data types that the Script wrapper supports.

Table 202. Script data types that map to DB2 data types

<table>
<thead>
<tr>
<th>XML data types</th>
<th>DB2 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>BLOB</td>
</tr>
<tr>
<td>character</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>character</td>
<td>CHARACTER FOR BIT DATA</td>
</tr>
<tr>
<td>character</td>
<td>CLOB (maximum length is 5 megabytes)</td>
</tr>
<tr>
<td>date</td>
<td>DATE</td>
</tr>
<tr>
<td>number</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>number</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>number</td>
<td>FLOAT</td>
</tr>
<tr>
<td>integer</td>
<td>INTEGER</td>
</tr>
</tbody>
</table>
Table 202. Script data types that map to DB2 data types (continued)

<table>
<thead>
<tr>
<th>XML data types</th>
<th>DB2 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>REAL</td>
</tr>
<tr>
<td>integer</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>character</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>character</td>
<td>VARCHAR FOR BIT DATA</td>
</tr>
</tbody>
</table>

Data types supported by the table-structured file wrapper

The following table lists the DB2 data types that the table-structured file wrapper supports.

Table 203. Table-structured file data types that map to DB2 data types

<table>
<thead>
<tr>
<th>Table-structured file data types</th>
<th>DB2 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>character</td>
<td>CLOB (maximum length is 5 megabytes)</td>
</tr>
<tr>
<td>number</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>number</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>number</td>
<td>FLOAT</td>
</tr>
<tr>
<td>integer</td>
<td>INTEGER</td>
</tr>
<tr>
<td>number</td>
<td>REAL</td>
</tr>
<tr>
<td>integer</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>character</td>
<td>VARCHAR</td>
</tr>
</tbody>
</table>

Data types supported by the Web services wrapper

The following table lists the DB2 data types that the Web services wrapper supports. The Web services wrapper uses XML data types.

Table 204. XML data types that map to DB2 data types for the Web services wrapper

<table>
<thead>
<tr>
<th>XML data types</th>
<th>DB2 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>BLOB</td>
</tr>
<tr>
<td>character</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>character</td>
<td>CHARACTER FOR BIT DATA</td>
</tr>
<tr>
<td>character</td>
<td>CLOB (maximum length is 5 megabytes)</td>
</tr>
<tr>
<td>date</td>
<td>DATE</td>
</tr>
<tr>
<td>number</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>number</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>number</td>
<td>FLOAT</td>
</tr>
<tr>
<td>integer</td>
<td>INTEGER</td>
</tr>
<tr>
<td>number</td>
<td>REAL</td>
</tr>
<tr>
<td>integer</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>character</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>character</td>
<td>VARCHAR FOR BIT DATA</td>
</tr>
</tbody>
</table>
Data types supported by the WebSphere Business Integration wrapper

The following table lists the DB2 data types that the WebSphere Business Integration wrapper supports. The WebSphere Business Integration wrapper uses XML data types.

Table 205. XML data types that map to DB2 data types for the WebSphere Business Integration wrapper

<table>
<thead>
<tr>
<th>XML data types</th>
<th>DB2 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>BLOB</td>
</tr>
<tr>
<td>character</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>character</td>
<td>CHARACTER FOR BIT DATA</td>
</tr>
<tr>
<td>character</td>
<td>CLOB (maximum length is 5 megabytes)</td>
</tr>
<tr>
<td>date</td>
<td>DATE</td>
</tr>
<tr>
<td>number</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>number</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>number</td>
<td>FLOAT</td>
</tr>
<tr>
<td>integer</td>
<td>INTEGER</td>
</tr>
<tr>
<td>number</td>
<td>REAL</td>
</tr>
<tr>
<td>integer</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>character</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>character</td>
<td>VARCHAR FOR BIT DATA</td>
</tr>
</tbody>
</table>

Data types supported by the XML wrapper

The following table lists the DB2 data types that the XML wrapper supports.

Table 206. XML data types that map to DB2 data types for the XML wrapper

<table>
<thead>
<tr>
<th>XML data types</th>
<th>DB2 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>BLOB</td>
</tr>
<tr>
<td>character</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>character</td>
<td>CHARACTER FOR BIT DATA</td>
</tr>
<tr>
<td>character</td>
<td>CLOB (maximum length is 5 megabytes)</td>
</tr>
<tr>
<td>date</td>
<td>DATE</td>
</tr>
<tr>
<td>number</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>number</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>number</td>
<td>FLOAT</td>
</tr>
<tr>
<td>integer</td>
<td>INTEGER</td>
</tr>
<tr>
<td>number</td>
<td>REAL</td>
</tr>
<tr>
<td>integer</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>character</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>character</td>
<td>VARCHAR FOR BIT DATA</td>
</tr>
<tr>
<td>XML document</td>
<td>XML</td>
</tr>
</tbody>
</table>
Accessing information about the product

IBM has several methods for you to learn about products and services.

You can find the latest information on the Web:

To access product documentation, go to publib.boulder.ibm.com/inofcenter/
db2luw/v9r5/topic/

You can order IBM publications online or through your local IBM representative.
• To order publications online, go to the IBM Publications Center at
  www.ibm.com/shop/publications/order
• To order publications by telephone in the United States, call 1-800-879-2755.

To find your local IBM representative, go to the IBM Directory of Worldwide
Contacts at www.ibm.com/planetwide

Providing comments on the documentation

Please send any comments that you have about this information or other
documentation.

Your feedback helps IBM to provide quality information. You can use any of the
following methods to provide comments:
• Send your comments using the online readers’ comment form at
  www.ibm.com/software/awdtools/rcf/
• Send your comments by e-mail to comments@us.ibm.com. Include the name of
  the product, the version number of the product, and the name and part number
  of the information (if applicable). If you are commenting on specific text, please
  include the location of the text (for example, a title, a table number, or a page
  number).
Accessible documentation

Documentation is provided in XHTML format, which is viewable in most Web browsers.

XHTML allows you to view documentation according to the display preferences that you set in your browser. It also allows you to use screen readers and other assistive technologies.

Syntax diagrams are provided in dotted decimal format. This format is available only if you are accessing the online documentation using a screen reader.
Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing 2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.
Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.
Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

**Trademarks**

IBM trademarks and certain non-IBM trademarks are marked at their first occurrence in this document.


The following terms are trademarks or registered trademarks of other companies:

Adobe®, the Adobe logo, PostScript®, the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine™ is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Intel®, Intel logo, Intel Inside® logo, Intel Centrino®, Intel Centrino logo, Celeron®, Intel Xeon®, Intel SpeedStep®, Itanium® and Pentium® are trademarks of Intel Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT® and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL® is a registered trademark and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library® is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Other company, product or service names may be trademarks or service marks of others.
collating sequences
federated database 7
column options
BioRS 391
BLAST 389
DB2 database 395
Entrez 400
Informix 406
Microsoft SQL Server 411
ODBC 415
Oracle 420
Script 424
Sybase 429
table-structured files 437
Teradata 434
Web services 444
WebSphere Business Integration 439
XML 450
command line processor (CLP)
issuing 1
CONCENTRATOR parameter 1
configuration files
Informix client
setting up and testing 169
configuration properties 335
configuring
federated servers 16
connection problems
Microsoft SQL Server data sources 192
connection-level isolation
supported data sources 385
connections
testing 88
troubleshooting 180, 298
connectivity
testing Oracle server connections 240
troubleshooting for Oracle data sources 240
Control Center
accessing 1
CREATE FUNCTION statement
BioRS syntax 21
CREATE NICKNAME examples
Microsoft SQL Server data sources 192
ODBC data sources 204, 219
Oracle data sources 242
CREATE NICKNAME statement
BioRS examples 32
BioRS syntax 30
BLAST, examples 76
BLAST, syntax 70
business application data sources 338, 339, 341
Entrez examples 108
Entrez syntax 106
examples, Excel 139
HMMER examples 161
HMMER syntax 160
Informix 181
script data sources 253
syntax, Excel 138
table-structured file examples 281
Create-structured file syntax 279
CREATE NICKNAME statement
(continued)
Teradata 300
troubleshooting for Teradata data sources 300
Web services data sources 313, 314
XML examples 375
CREATE SERVER examples
OLE DB data sources 227
Oracle data sources 238
Sybase data sources 267
CREATE SERVER statement
BioRS examples 26
BioRS syntax 25
BLAST, example 68
BLAST, syntax 67
DB2 family data sources 86
Entrez examples 98
Entrez, syntax 97
example, Excel 138
examples for ODBC 201, 215
HMMER examples 156
HMMER syntax 155
Informix 176
script data sources 251
Sybase data sources 267
syntax, Excel 137
table-structured file examples 279
table-structured file syntax 279
Teradata 265
Web services example 310
XML examples 366
XML syntax 365
CREATE USER MAPPING examples
ODBC data sources 203, 217
OLE DB data sources 228
Oracle data sources 239
Teradata data sources 296
CREATE USER MAPPING statement
BioRS examples 29
BioRS syntax 28
BLAST syntax 66
BLAST, examples 66
Entrez examples 101
Entrez syntax 100
HMMER syntax 158
HMMER examples 159
Informix 178
Sybase data sources 269
CREATE VIEW statement
XML examples 380
CREATE WRAPPER
examples for ODBC 199, 214
CREATE WRAPPER statement
BioRS examples 24
BioRS syntax 22
BLAST examples 65
BLAST, example 64
Entrez examples 96
Entrez syntax 95
Excel, example 136
HMMER examples 154
HMMER syntax 153
registering OLE DB data sources 225
table-structured file example 277
table-structured file syntax 277
XML examples 364
CREATE WRAPPER statement
(continued)
XML syntax 362
create_function_mappings.ddl
Entrez data sources 93
creating wrappers
WebSphere Business Integration 336
custom functions
BioRS 35
BioRS, descriptions 21
BioRS, registering 21
BLAST, registering 78
Entrez data sources 110
Entrez data sources, registering 93
Entrez, description 94
Entrez, syntax 94
script 245

D
daemon
BLAST data sources, configuration examples 61
BLAST data sources, configuring 57
BLAST data sources, starting 61
BLAST data sources, troubleshooting examples 62
custom configuration file
script examples 246
custom configuration for script 246
HMMER, starting 147, 150
starting for script 248
daemon configuration file
HMMER examples 150
DAEMON_LOGFILE_DIR
script daemon 246
DAEMON_PORT
script daemon 246
data import
supported data sources 385
data sources
business applications 331
configuration overview 11
configuring 1, 83
configuring access to Informix 169
message files 4
options 389
registering 84
registering Informix server definitions
DB2 command line 175
registering Informix wrappers 174
supported features 385
testing the connection to the Informix server 179
troubleshooting connections 180, 298
valid server types 463
data type mappings
forward 465
reverse 474
data types
for nonrelational data sources 484
databank
description 19
databases
cataloging remote 83
DB2 data sources
configuring 83
Entrez data sources (continued)
  CREATE SERVER statement 86
  adding to a federated system 93
  CREATE NICKNAME statement
  examples 108
  CREATE NICKNAME statement
  syntax 106
  CREATE SERVER statement
  examples 98
  CREATE SERVER statement
  syntax 97
  CREATE USER MAPPING statement
  examples 101
  CREATE USER MAPPING statement
  syntax 100
  CREATE WRAPPER statement
  examples 96
  CREATE WRAPPER statement
  syntax 95
  custom functions
  syntax 110
  custom functions, description 94
  custom functions, syntax 94
  default directory path 96
  description 91
  invalid WHERE clauses 112
  limiting query results 112
  nickname options 109
  nicknames, alternative names 109
  nicknames, description 102
  nicknames, Nucleotide database 102
  nicknames, OMIM database 103
  nicknames, predefined names 108
  nicknames, PubMed database 105
  nicknames, registering 106
  Nucleotide nicknames, fixed
  columns 115
  prerequisites 93
  proxy server, user mapping
  example 101
  proxy server, wrapper examples 96
  proxy servers 101
  PubMed nicknames, fixed
  columns 126
  query examples 113
  registering custom functions 93
  relational predicates 112
  server definition, registering 97
  user mappings, creating 100
  wrapper library files 96
  wrapper, registering 95
  environment variables 2
  configuring Microsoft SQL Server
  variables 186
  configuring Sybase environment
  variables 260
  data sources 2
  Informix 171
  Informix wrappers 170
  Microsoft SQL Server data
  sources 187
  Oracle data sources 232
  required 2
  restrictions 3
  setting 2
  setting Oracle variables 231
  setting Teradata environment
  variables 288
  environment variables (continued)
  Sybase data sources 262
  Teradata data sources 290
  troubleshooting the Sybase
  environment variable 272
  error logging
  disabling Teradata access
  logging 302
  error tolerance
  supported data sources 385
  examples
  BioRS data sources
  proxy server, user mapping
  example 29
  BioRS queries 39, 41
  BLAST daemon configuration file 61
  BLAST queries 80
  business application wrappers 325, 352
  CREATE NICKNAME statement,
  BioRS 32
  CREATE NICKNAME statement,
  BLAST 76
  CREATE NICKNAME statement,
  Entrez 108
  CREATE NICKNAME statement,
  Excel 139
  CREATE NICKNAME statement,
  HMMER 161, 165
  CREATE NICKNAME statement,
  table-structured file 281
  CREATE NICKNAME, HMMER 161
  CREATE SERVER statement,
  BioRS 26
  CREATE SERVER statement,
  BLAST 68
  CREATE SERVER statement,
  Entrez 98
  CREATE SERVER statement,
  Excel 138
  CREATE SERVER statement,
  HMMER 156
  CREATE SERVER statement,
  HMMER, 165
  CREATE SERVER statement,
  table-structured file 279
  CREATE SERVER statement, Web
  services 310
  CREATE SERVER statement,
  XML 366
  CREATE USER MAPPING statement,
  BioRS 29
  CREATE USER MAPPING statement,
  BLAST 66
  CREATE USER MAPPING statement,
  Entrez 101
  CREATE USER MAPPING statement,
  Entrez 159
  CREATE VIEW statement, XML 380
  CREATE WRAPPER statement,
  BioRS 24
  CREATE WRAPPER statement,
  BLAST 65
  CREATE WRAPPER statement,
  Entrez 96
  CREATE WRAPPER statement,
  Excel 136
C

examples (continued)
CREATE WRAPPER statement, HMMER 154, 165
CREATE WRAPPER statement, table-structured file 277
CREATE WRAPPER statement, XML 364
db2blast_daemon command 62
db2hmmer_daemon command 151
db2script_daemon command 248
Excel files 131, 132
Excel queries 140
HMMER daemon configuration file 150
proxy server
BioRS user mapping example 29
queries, HMMER 165
script data sources, CREATE NICKNAME 253
script data sources, CREATE SERVER 251
UPDATE statement, BioRS 49
Web services wrappers 315, 325, 352
WebSphere Business Integration wrapper 341
XML document mapping 359
XML examples 382
examples, CREATE SERVER
Informix 176
Teradata 295
example 165
examples, CREATE USER MAPPING statement
Informix 178
Sybase data sources 269
eamples, queries
business application data sources 355
Nucleotide database 113
OMIM database 113
PubMed database 113
Web services 328
Excel
federated features supported 385
nickname options 403
server options 403
wrapper options 403
Excel data sources 131
adding Excel to a federated system 135
authorizations for file access 142
CREATE NICKNAME statement, examples 139
CREATE SERVER statement, examples 138
CREATE WRAPPER statement, examples 136
default directory path 137
description 131
nicknames, registering 138
prerequisites 135
sample queries 140
sample user scenario 140
server definition, registering 137
wrapper library names 137
wrapper, registering 136
Excel files
data types, supported 484
preparing to access 132
Excel ODBC driver
accessing Microsoft Excel workbooks 208
function mappings
options 461
G
GenBank, see Nucleotide 91
Genome database 91
global catalog
views containing federated information 457
H
HMMER
nickname options 403
server options 403
user mapping options 403
wrapper options 403
HMMER data source
data types, supported 484
HMMER data sources 143
adding HMMER to a federated system 146
CREATE NICKNAME statement example 165
CREATE NICKNAME statement examples 161
CREATE SERVER statement example 165
CREATE SERVER statement examples 156
CREATE USER MAPPING statement syntax 158
CREATE WRAPPER statement 154
CREATE WRAPPER statement example 165
daemon configuration file examples 150
daemon, configuring 147
daemon, starting 150
default directory path 153
description 143
eamples, queries 165
executable file version, verifying 146
fixed input columns 162
fixed output columns 163
list of utilities 143
nicknames, fixed columns 161
nicknames, registering 160
prerequisites 146
proxy server, server definition example 156
proxy server, user mapping example 159
proxy server, wrapper example 154
server definition, registering 155
user mappings, creating 158
wrapper library files 153
wrapper, registering 153
hmmpfam executable file
description 143
HTTP proxy
supported data sources 385
Informix  
- code page conversion 172
- column options 406
- CREATE NICKNAME statement 181
- CREATE SERVER statement 176
- CREATE USER MAPPING statement 178
- default directory path 174
- default forward type mappings 465
- default reverse type mappings 474
- djlinkInformix script 5
- library file names 174
- library files 5
- server options 406
- user mapping options 406
- wrapper options 406
- Informix client configuration file setting up and testing 169
- Informix data sources  
  - environment variables 2
  - library files 4
- Informix server  
  - testing the connection 179
- Informix synonyms  
  - registering nicknames 181
- Informix tables  
  - registering nicknames 181
- Informix views  
  - registering nicknames 181
- Informix wrappers  
  - creating user mappings 178
  - environment variables 170, 171
  - improving performance 180
  - registering 174

Label-based access control  
- supported data sources 385

Legal notices 493

Libraries  
- BioRS wrapper 23
- BLAST wrapper 64
- Excel wrapper 137
- table-structured file wrapper 278
- Teradata wrapper 293
- XML wrapper 363

Library  
- script wrapper 250
- library files  
  - file names 4
  - linking 4, 5
- license keys  
  - federation 6
- license policies  
  - federation 6
- licenses  
  - federation 6
- linking  
  - library files 4
- LOB  
  - supported data sources, read and write 385

LOB (continued)  
- supported data sources, read-only 385
- location, default 2
- lowercase characters  
  - use in SQL statements 13

M  
- mapping  
  - XML documents 323, 351
  - materialized query tables  
    - supported data sources 385
  - message queues required 335
- Microsoft Excel workbooks  
  - accessing 208
- Microsoft SQL data sources  
  - defining user mappings 190
- Microsoft SQL Server  
  - column options 411
  - default forward type mappings 465
  - default reverse type mappings 474
  - djlinkMssql script 5
  - federated features supported 385
  - library files 5
  - server options 411
  - Unicode support 10
  - user mapping options 411
  - wrapper options 411
- Microsoft SQL server data sources  
  - accessing 184
- Microsoft SQL Server data sources  
  - configuring access 183
  - configuring federated servers 183
  - configuring the environment variables 186
  - CREATE NICKNAME example 192
  - environment variables 2, 187
  - library files 4
  - nicknames 191
  - registering 187
  - registering server definitions 189
  - testing connections to Microsoft SQL Server 190
  - troubleshooting the configuration 192
  - wrapper library files 188

N  
- national language support (NLS)  
  - federated database 7
  - nickname and column options  
    - Web services wrapper 303, 323, 352
    - WebSphere business integration wrappers 323, 352
  - nickname hierarchy, examples 341
  - nickname options  
    - BioRS 391
    - Blast 389, 403
    - DB2 database 395
    - Entrez 400
    - HMMER 403
    - Script 424
  - table-structured files 437
  - Web services 444

nickname options (continued)  
- WebSphere Business Integration 439
  - WebSphere business integration wrappers  
    - nickname hierarchy, examples 341
- XML 450
- nickname statistics update facility  
  - supported data sources 385
- nickname statistics update facility  
  - supported data sources 385
- nicknames  
  - business application data sources 338, 339, 341
  - business applications 331
  - CREATE NICKNAME examples 270, 271
  - creating 313, 314, 338, 339, 341
  - Web services 315
  - database techniques 325, 352
  - description 252, 323, 351
  - Oracle data sources 241
  - registering 88
  - registering for Informix tables, views, and synonyms 181
  - registering for Microsoft SQL Server data sources 191
  - registering for ODBC data sources 204, 218
  - registering for scripts 252
  - required input columns  
    - business application data sources 339
    - Sybase data sources 269, 271
    - Teradata data sources 298, 299
    - Web services data sources 313, 314
    - Web services description language (WSDL) 303

Nodes  
- cataloging 83
- DB2 83
- nonrelational data sources  
  - supported data types 484

Nucleotide database  
- description 91
- query examples 113
- search fields 115

Object Discovery Agent  
- business applications 334

ODBC  
- column options 415
- default forward type mappings 465
- environment variables 2
- federated features supported 385
- server options 415
- Unicode support 10
- user mapping options 415
- wrapper options 415
- ODBC data sources  
  - accessing Microsoft Excel workbooks 208
  - configuring access to ODBC data sources 195

Index 501
ODBC data sources (continued)
configuring access to WebSphere Classic Federation Server for z/OS data sources 211
configuring federated servers 196
configuring the federated server 197
CREATE NICKNAME examples 204, 219
CREATE SERVER examples 201, 215
CREATE USER MAPPING examples 203, 217
CREATE WRAPPER examples 199, 214
defining user mappings 202, 216
registering 197, 212
registering nicknames 204, 218
registering server definitions 200, 215
test tables for the ODBC tuning utility 207, 222
testing access 203, 218
tuning with the ODBC tuning utility 205, 219
wrapper library files 198, 213
ODBC traces 192
ODBC tuning utility
creating test tables 207, 222
db2edsrvcfg command examples 206, 221
db2edsrvcfg command syntax 206, 221
tuning ODBC wrappers 205, 219
OLE DB data sources
configuring access 225
CREATE SERVER examples 227
CREATE USER MAPPING examples 228
defining user mappings 228
library file names 226
registering 225
registering server definitions 226
server options 227
OMIM database
description 91
nicknames, fixed columns 118
query examples 113
search fields 118
optimization
script data sources 257
Oracle
column options 420
default forward type mappings 465
default reverse type mappings 474
djlinkOracle script 5
federated features supported 385
library files 5
server options 420
user mapping options 420
wrapper options 420
Oracle client configuration file
configuring and testing 235
Oracle data sources
accessing 231
adding to federated systems 231
code page conversion 234
configuring connectivity 235
CREATE NICKNAME examples 242
CREATE SERVER examples 238
Oracle data sources (continued)
CREATE USER MAPPING examples 239
defining user mappings 239
evironment variables 2, 232
library files 4
registering 236
registering nicknames 241
registering server definitions 237
setting environment variables 231
troubleshooting connectivity problems 240
wrapper library files 236
Oracle server
testing the connection 240
query optimization
guidelines for BioRS queries 46
guidelines for table-structured files 286
XML 381
quotation marks
use in SQL statements 13
quotes
use in SQL statements 13
R
REMOTE_NAME function mapping option
valid settings 461
restrictions
db2dj.ini file 3
reverse type mapping
Unicode 482
reverse type mappings
default mappings 474
Unicode 482, 483, 484
S
SAP
business objects 331
scenarios
accessing Excel files 140
screen readers 489, 491
Script
column options 424
federated features supported 385
nickname options 424
server options 424
user mapping options 424
wrapper options 424
script data sources
CREATE NICKNAME statement example 253
CREATE SERVER statement 251
nickname options 253
optimizing performance 257
registering nicknames 252
sample queries 256
server definition 251
wrapper library names 250
script wrapper
creating 250
custom functions 245
overview 243
scripts
configure as data sources 243
search tags
Nucleotide nicknames 115
OMIM nicknames 118
PubMed nicknames 126
Secure Socket Layer (SSL)
supported data sources 385
security
Web services 312
server definitions 337
registering 85
registering for Microsoft SQL Server data sources 189
queries
accessing BioRS data sources 35, 41
accessing BLAST data sources 78, 80
accessing data sources
business applications 355
Web services 328
accessing Excel data sources 140
BioRS equality operations 37
examples
business applications 325, 352
Web services wrappers 325, 352
script data sources, accessing data sources 256
XML queries 382
server definitions (continued)
registering for ODBC data sources 200, 215
registering for OLE DB data sources 226
registering for Oracle data sources 237
registering for Sybase data sources 266
registering for Teradata data sources 294
registering Informix data sources DB2 command line 175
Web services 310
server options
BioRS 391
Blast 389
DB2 database 395
Entrez 400
examples for OLE DB 227
Excel 403
HMMER 403
Informix 406
Microsoft SQL Server 411
ODBC 415
Oracle 420
Script 424
setting the FOLD_ID and FOLD_PW server options 180
Sybase 429
table-structured files 437
Teradata 434
Web services 444
WebSphere Business Integration 439
XML 450
server types
valid federated types 463
servers
testing connections to Microsoft SQL Server 190
shredding business objects nicknames 331
Siebel
business objects 331
SOAP headers federated access 321
SOCKS proxy supported data sources 385
specifying values options 13
SQL10013N troubleshooting 272
SQL1097N troubleshooting 273
SQL1822N troubleshooting 272
statement-level isolation supported data sources 385
statistics
for BioRS data sources 47
switches
BLAST fixed input columns 74
Sybase
column options 429
default forward type mappings 465
default reverse type mappings 474
djxlinkSybase script 5
Sybase (continued)
federated features supported 385
library files 5
server options 429
user mapping options 429
wrapper options 429
Sybase client configuration file configuring and testing 263, 264
Sybase data sources configuring access 259
configuring the client configuration file 263, 264
configuring the environment variables 260
CREATE NICKNAME examples 270, 271
CREATE SERVER examples 267
CREATE USER MAPPING examples 269
CREATE USER MAPPING statement 269,
creating user mappings 268
environment variables 2, 262
library files 4
registering 265
registering nicknames 269, 271
registering server definitions 266
troubleshooting wrapper configurations 272, 273
wrapper default directory path 265
wrapper library file names 265
SYSCAT catalog views 457
SYSTAT catalog views 457

T

table-structured file data sources 275
adding to a federated system 277
authorizations for file access 285
CREATE NICKNAME statement examples 281
CREATE NICKNAME statement syntax 279
CREATE SERVER statement examples 279
CREATE SERVER statement syntax 279
CREATE WRAPPER statement example 277
CREATE WRAPPER statement syntax 277
default directory path 278
file type attributes 275
nicknames, registering 279
overview 275
prerequisites 277
query optimization 286
server definition, registering 279
wrapper library names 278
wrapper overview 276
wrapper, registering 277
table-structured files
column options 437
data types, supported 484
federated features supported 385
HMMER federated features supported 385
table-structured files (continued)
Informix federated features supported 385
nickname options 437
server options 437
Unicode support 10, 11
wrapper options 437
templates
business objects 334
Teradata
column options 434
CREATE NICKNAME statement example 300
CREATE SERVER statement 295
default forward type mappings 465
default reverse type mappings 474
djxlinkTeradata script 5
federated features supported 385
library files 5
server options 434
user mapping options 434
wrapper library names 293
wrapper options 434
Teradata data sources configuring access 287
CREATE USER MAPPING examples 296
defining user mappings 296
disabling access logging 302
environment variables 2, 290
library files 4
nicknames 299
registering 292
registering nicknames 298
registering server definitions 294
setting environment variables 288
testing the client configuration 287
testing the connection 287
testing the connection to the federated server 297
troubleshooting character sets 292, 301
troubleshooting connectivity 297
troubleshooting nicknames 301
troubleshooting the CREATE NICKNAME statement 300
tuning 302
verifying run-time linking 288
trademarks 495
troubleshooting
connections 180, 298
data source access 1
defederated server setup 4
defederation installation 4, 5
TurboBlast
configuring 77
two-phase commit supported data sources 385

U

Unicode 8, 10, 11, 481, 482, 483, 484
supported data sources 385
UPDATE or DELETE operation errors troubleshooting for Teradata data sources 301
XML data sources (continued)
query optimization  371
query optimization, cost model
  facility  381
query optimization, federated views  379
query optimization, overview  378
server definition, registering  365
tuples  370
wrapper library names  363
wrapper, registering  362
XPath expressions  370
XML nicknames
  federated views  379
XML schemas
  business applications  334
  mapping to nicknames  323, 351
XPath
  using with nicknames  323, 351