Heterogeneous data access for i5/OS applications

Version 3

Kent Milligan and Colin Hendricks
ISV Business Strategy and Enablement
January 2008

Table of contents

Abstract

Many IBM clients have business data scattered across disparate data sources. Applications need to be able to access and consolidate the data in different databases. This white paper makes you aware of the options available for IBM i5/OS applications to access heterogeneous data. It features IBM WebSphere Federation Server, which helps solve the heterogeneous data-access problem for i5/OS applications by providing a seamless SQL interface for accessing data that is not stored in an IBM DB2 table.

Introduction

What are the chances that all of the data that your company needs for its day-to-day operations is stored on the IBM® System i™ platform in its IBM DB2® for i5/OS® database? In today's IT world, the answer to this question is most likely *slim* or *none*. Disparate data sources appear in many situations. For example, another organization in your company might have hired some PC programmers to create a departmental solution and a couple of your existing System i applications now need to access the data that resides in the database on the PC server. Or your company might have merged with another business that does not use i5/OS systems, yet you need to share data.

Obviously, it is important to know the available methods for i5/OS programs to access data that is not stored in DB2 (such as Oracle and SQL Server). This paper makes you aware of the options for i5/OS applications to access heterogeneous data.

Why extra software?

Why does your i5/OS system need additional software for heterogeneous database access? DB2 for i5/OS (and the IBM DB2 Family) has long supported the Open Group DRDA Distributed Relational Database Architecture (DRDA) standard for remote database access and interoperability. Today, i5/OS applications can use DRDA to access the other DB2 products that run on UNIX® and Intel® servers, but not other databases.

Other database vendors, such as Microsoft® SQL Server and Oracle, offer tools that are based on DRDA. Unfortunately, they have chosen to only support part of the DRDA standard. They have implemented the specifications that enable them to access data in other databases, but do not support the DRDA specifications that allow DRDA compliant databases (that is, DB2 for i5/OS) to access and update their data. Other methods must be used until all of the database products fully embrace the Open Group DRDA standard.

Note that many client-based tools and data-replication products tout their ability to move data from one database product to another (for example, Oracle to DB2). However, an i5/OS application is unable to use these stand-alone data-movement utilities to directly access data that is not stored in a DB2 table — data is copied to an i5/OS system where the application can access copies of the data.

However, accessing copies of data has a few drawbacks. First, the data in the copied tables is not current — an application on the other server might have updated the data the moment that you finished copying it. Second, it takes time and network resources to copy blocks of data from one server to another. Finally, consider the storage and management costs of having two copies of data on your systems.

The Java approach

One option that allows i5/OS applications to directly access data that is not stored in DB2 is to leverage the portability of Java™ code. Many databases have JDBC drivers are implemented in pure Java — meaning that any standard Java virtual machine (JVM) can use the JDBC driver. (This is also known as a *Type 4 JDBC driver*.)

The JDBC driver that is included in the IBM Toolbox for Java is an example of a Type 4 JDBC driver that can run on different platforms. Java applications that run on an i5/OS system can use the JDBC driver that is in IBM Toolbox for Java to access DB2 for i5/OS. Java applications that run on a UNIX or Intel server can also use the IBM Toolbox for Java JDBC driver to access DB2 for i5/OS.

In a similar fashion, Java applications that run on i5/OS can use a Type 4 JDBC driver to enable heterogeneous data access to data that is stored in a remote database, such as Oracle. For example, you can install the Oracle Type 4 JDBC driver in the Integrated File System (IFS) so that a Java program on an i5/OS system can use the Type 4 JDBC driver to process Oracle databases. The Java code that runs in i5/OS applications issues JDBC requests to retrieve or change the data in the target data source. Then, the JDBC driver that is associated with the target converts the JDBC requests into a database language that the target data source supports.

Clearly, this approach is only palatable to companies that have Java skills, as it requires the use of JDBC. One possible upside of this approach is that some databases have Type 4 JDBC drivers that can be downloaded at no charge. Type 4 JDBC drivers are available from such vendors as Oracle, Microsoft SQL Server and HiT Software. (**Note:** Not all of these vendors' drivers are free.)

The IBM solution

When you need to access data that is not stored in DB2 from an i5/OS high-level language program, the ideal solution is IBM WebSphere® Federation Server. You might not be aware of this IBM product because it has undergone some names changes. Here are previous names used for the WebSphere Federation Server product:

- IBM DB2 DataJoiner®
- IBM DB2 Relational Connect
- IBM DB2 Information Integrator
- IBM WebSphere Information Integrator

With this product, now being part of the WebSphere brand, you might wonder if the product requires you to create and run an IBM WebSphere Application Server instance to enable access to data that is not stored in DB2. Despite the name, a server instance is **not** required for the WebSphere Federation Server. Another caveat with this solution is that it requires that you access the data with an SQL-based interface such as CLI, embedded SQL or JDBC.

But this is an advantage, as these SQL-based interfaces are industry standards. Many products take a similar approach to provide heterogeneous data access, but they use proprietary SQL-based interfaces instead of standards. Products in this category include EDA and iWay Software from Information Builders (www.informationbuilders.com), RPG2SQL Integrator from RJS Software (www.rjssoftware.com), Attunity Connect (www.attunity.com) and DataGlider software from DataGlider (www.dataglider.com).

WebSphere Federation Server offers another advantage — it is a single product that lets you access several different data sources, instead of requiring a distinct access product for each unique data source.

Most i5/OS programs will probably use WebSphere Federation Server for heterogeneous data access through embedded SQL. More and more i5/OS programs are using embedded SQL and WebSphere Federation Server, which makes that type of programming even more attractive and powerful. With the DB2 for i5/OS SQL precompilers, you can embed SQL statements in RPG, C, C++ and COBOL applications on an i5/OS system.

Figure 1 is a sample of the embedded SQL that is necessary to read an Oracle table with WebSphere Federation Server from an i5/OS application. (**Note:** This example assumes that the needed setup, which is discussed later in this white paper, is already complete.)

```
EXEC SQL CONNECT TO remotedb;
EXEC SQL SELECT custpref INTO :locpref FROM orcustomer
                      WHERE custid=22;
```

Figure 1. Sample of embedded SQL that reads an Oracle table

If you use DRDA to access data on another i5/OS system or DB2 server, you will notice that the SQL interface is the same. These embedded SQL statements coexist with an application that uses the native (that is, non-SQL) i5/OS database interfaces, so you do not need to switch the entire application to use SQL. Other SQL-based interfaces on an i5/OS system, such as JDBC, CL and the IBM DataPropagator™ product, can also leverage WebSphere Federation Server for heterogeneous database access.

A closer look

A simple way to understand how WebSphere Federation Server accesses data outside of DB2 is to think of it as a language translator (although, in a moment, you will see that it is more than a translator).

If a doctor can only speak English and the patient can only speak French, then the only way the doctor can communicate his diagnosis to that patient is to use a language translator who listens to the English words and then speaks the corresponding French words to the patient. WebSphere Federation Server fills a similar role by being able to speak the DRDA language with DB2 databases and non-DRDA language with heterogeneous data sources (for example, Oracle SQL*Net and ODBC).

The i5/OS applications send SQL requests to WebSphere Federation Server through DRDA, which translates that request into a language (for example, ODBC) that the other database (for example, SQL Server) can understand. When the other database server finishes the requested task, WebSphere Federation Server sends the results of the database request back to the i5/OS application through DRDA.

WebSphere Federation Server does not run on an i5/OS system — it runs only on IBM AIX®, Linux®, Microsoft Windows and Sun Solaris operating systems. You can easily install WebSphere Federation Server on the same system that hosts the other database that you need to access. Another plausible scenario is to use the IBM System i platform's integration capabilities with the IBM BladeCenter® and IBM System x™ platforms. You can install WebSphere Federation Server on one of the servers that is compatible with Intel and that is available with this integration point. This approach achieves tighter integration between the i5/OS system and WebSphere Federation Server.

Microsoft SQL Server, Oracle, Sybase and Teradata are just some of the data sources that WebSphere Federation Server supports for both read and write access.

Setup and configuration overview

After seeing how simple the SQL code is for connecting to an Oracle database (see Figure 1) from an i5/OS program with WebSphere Federation Server, you might wonder how WebSphere Federation Server knows that it should connect to an Oracle server instead of another i5/OS or DB2 server. The answer is found in the fact that you must tell your federated database how to do this interpretation with some simple, one-time configuration steps.

In Figure 1, you connect to a WebSphere Federation Server database called remotedb. You need to create that database for WebSphere Federation Server in DB2 and then set up your DB2 relational database directory on an i5/OS system. You do this with CL commands WRKRDBDIRE and ADDRDBDIRE so that the i5/OS system can connect to the database remotedb on your Microsoft Windows or UNIX server. The remotedb DB2 database has its own catalog that contains access information for the Oracle database.

To access a data source with WebSphere Federation Server, update the catalogs by running the following DB2 SQL statements on the Windows or UNIX server:

```
CREATE WRAPPER
CREATE SERVER
CREATE USER MAPPING
CREATE NICKNAME
```

Some detailed configuration samples are included in the next section, but essentially, the listed SQL statements let you tell WebSphere Federation Server where to find the target database server, how to communicate with the target database server (for example, ODBC and SQL*Net), and how to find the objects you want to access on the target database server.

Depending on the server that you are accessing, you might also have to load a data-access module (for example, ODBC and Sybase Open Client) on the same server where WebSphere Federation Server is installed. This configuration allows WebSphere Federation Server to fulfill its role as a language translator for an i5/OS system — it receives a DRDA request from the i5/OS system and then looks in the catalogs to determine how to translate the request into a language that the target database understands. After this configuration is complete, the i5/OS application issues normal SQL requests against the target database.

You are not limited to accessing a single database at a time. You can actually reference tables and join them together from different databases — all on the same SQL statement (the *join* part of the parent product name). It is very easy for an i5/OS application to present a consolidated view of the enterprise data with a single SQL statement, even when the user data is scattered across multiple databases (such as DB2, Oracle and SQL Server).

One-time setup details

The majority of the setup and configuration for WebSphere Federation Server occurs on the Microsoft Windows or UNIX server where that product is installed.

As mentioned in the previous section, a DB2 database is needed to catalog all of the information on how to access the target-data source. The next step is to create a DB2 database on the system. You can accomplish this with the CREATE DATABASE SQL statement. For this purpose, a quick way to create a database with sample data is to use the DB2 sample database utility — simply issue the DB2SAMPL statement in the DB2 Command window. This utility creates a database, appropriately named SAMPLE, on the Windows or UNIX server. In this example, the CREATE DATABASE SQL statement builds a database named FEDDB, which is the repository of the configuration objects that need to be created for WebSphere Federation Server.

You need to perform this configuration while you are connected to the DB2 federated database that catalogs all the details for WebSphere Federation Server. FEDDB is the DB2 database that is used for that purpose in this example, so make sure a CONNECT TO FEDDB statement runs before you begin the configuration steps manually.

You can do the configuration for the data access through the graphical interfaces provided by DB2 Control Center or with the CLI that the DB2 command-line processor (CLP) provides. The configuration examples that follow for SQL Server database access use the DB2 Control Center interface, but also include the SQL statement that the graphical interface generates.

Starting up DB2 Control Center

This section explains the steps taken in this example to start up DB2 Control Center.

From a Windows client, launch DB2 Control Center by clicking **Start -> Programs -> IBM DB2 -> General Administration Tools -> Control Center**. Figure 2 shows the Control Center interface.

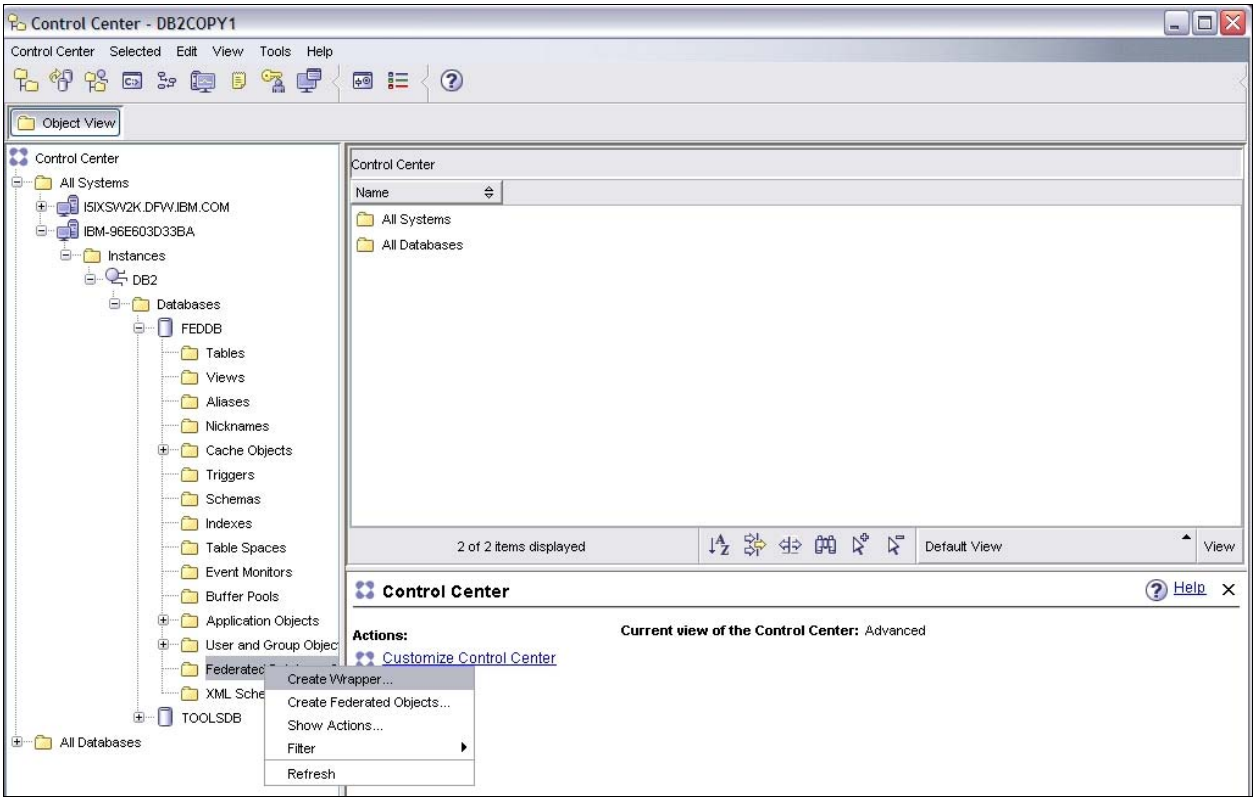

Figure 2. DB2 Control Center

The data access configuration must occur on the server where WebSphere Federation Server is installed. From the Control Center interface, select the Windows system **IBM-96E603D33BA** (instead of selecting a System i model).

> After you select the system, a connection is established to the DB2 for Windows database that is used to store the configuration for the remote databases. As you can see in Figure 2, FEDDB is the database in this example.

Setting up SQL Server access

Now, that a connection to the federated database exists, everything is in place to set up access to the SQL Server database from an i5/OS application.

1. You initiate the configuration process by right-clicking the **Federated Databases** folder in the FEDDB database. The easiest way to walk through the entire configuration process is to select the **Create Federated Objects** task. This brings up the dialog shown in Figure 3, which displays the four configuration objects that you need to create for accessing a federated database: wrappers, server definitions, user mappings and nicknames. As you will see in this SQL Server example, the creation of nicknames is probably the only configuration step that you need to repeat.

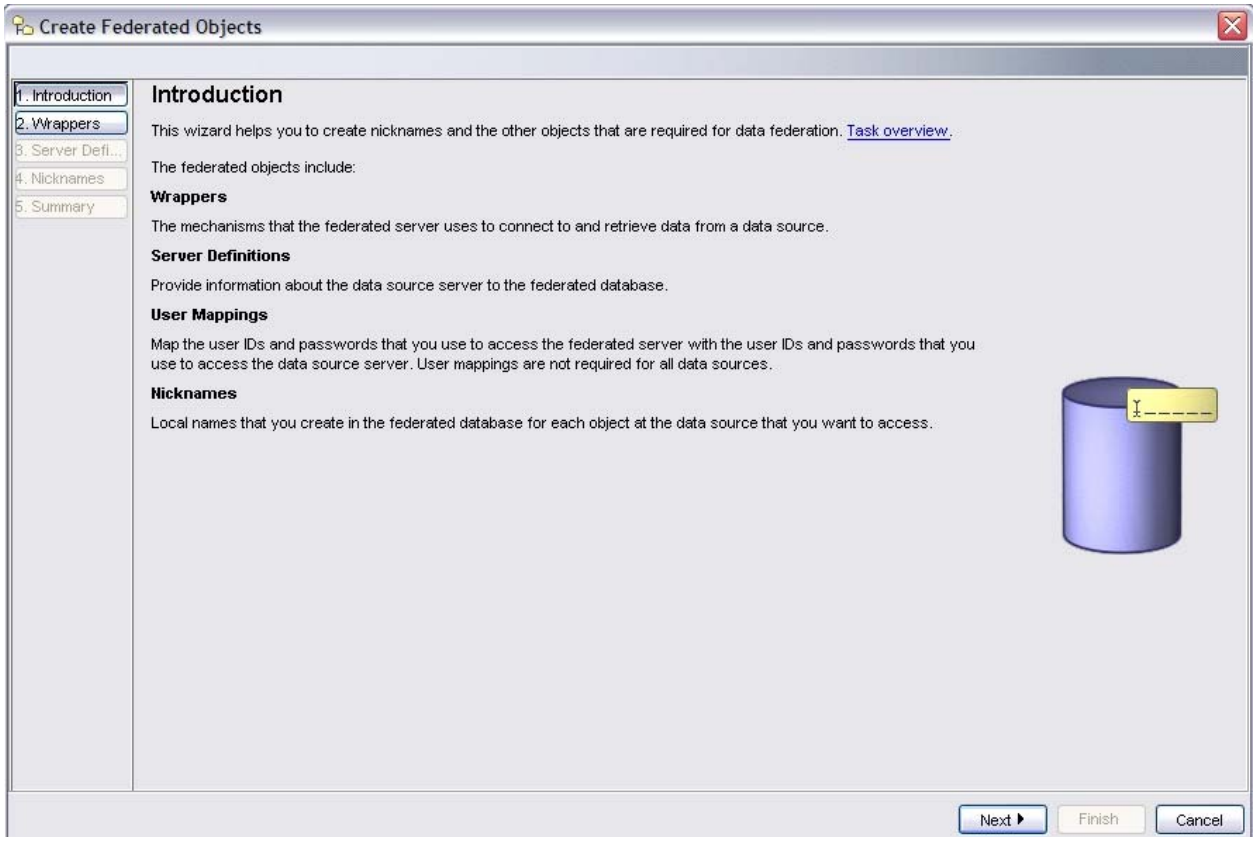

Figure 3. Create Federated Objects interface

Click **Next** to take the first step of creating a wrapper. The federated wrapper identifies the language that will be used to access the SQL Server database. Thus, SQL Server is selected as the data source type in Figure 4.

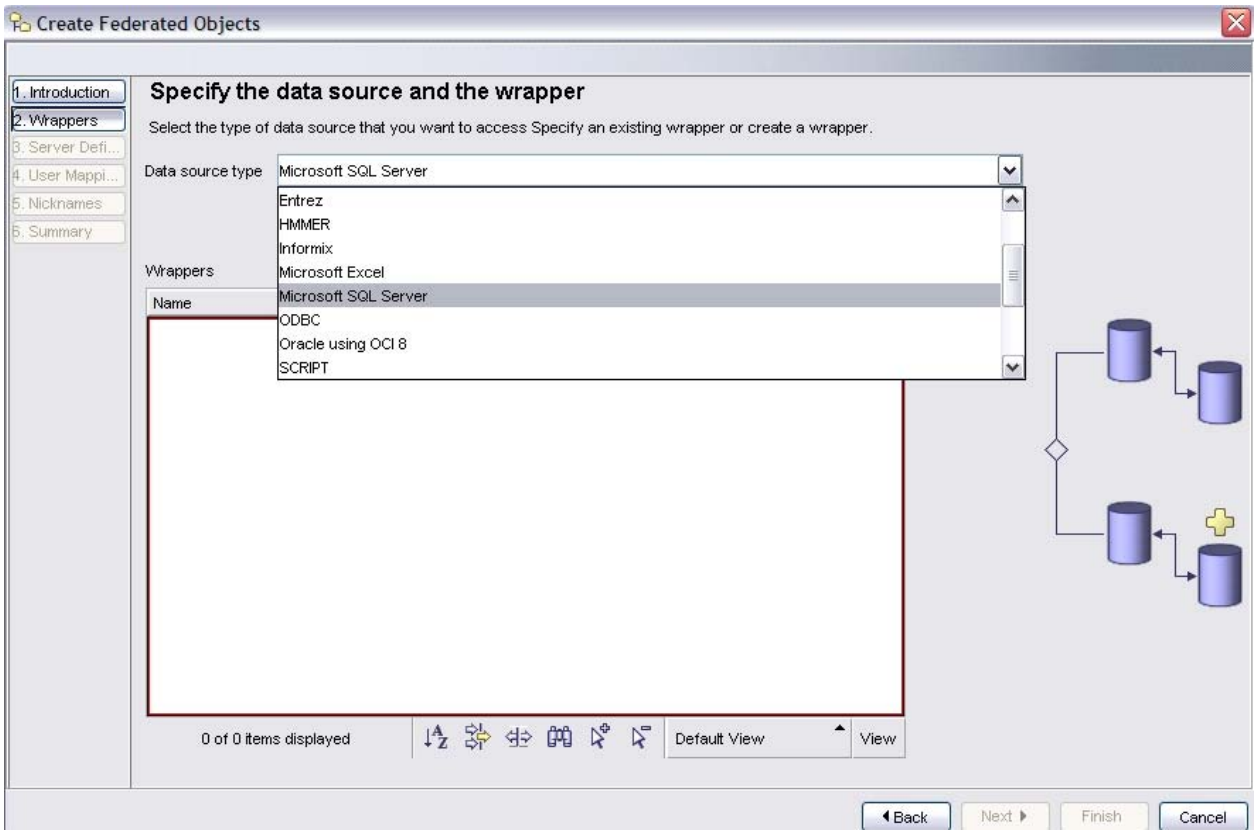

Figure 4. Wrapper data source

After you identify the data source, click **Create** to display the Create Wrapper panel (see Figure 5).

Use the default values for the wrapper and library name and click **OK** to create the wrapper. The Create Wrapper panel then generates and runs the following SQL statement:

```
CREATE WRAPPER MSSQL3 LIBRARY 'db2mssql3.dll'
```

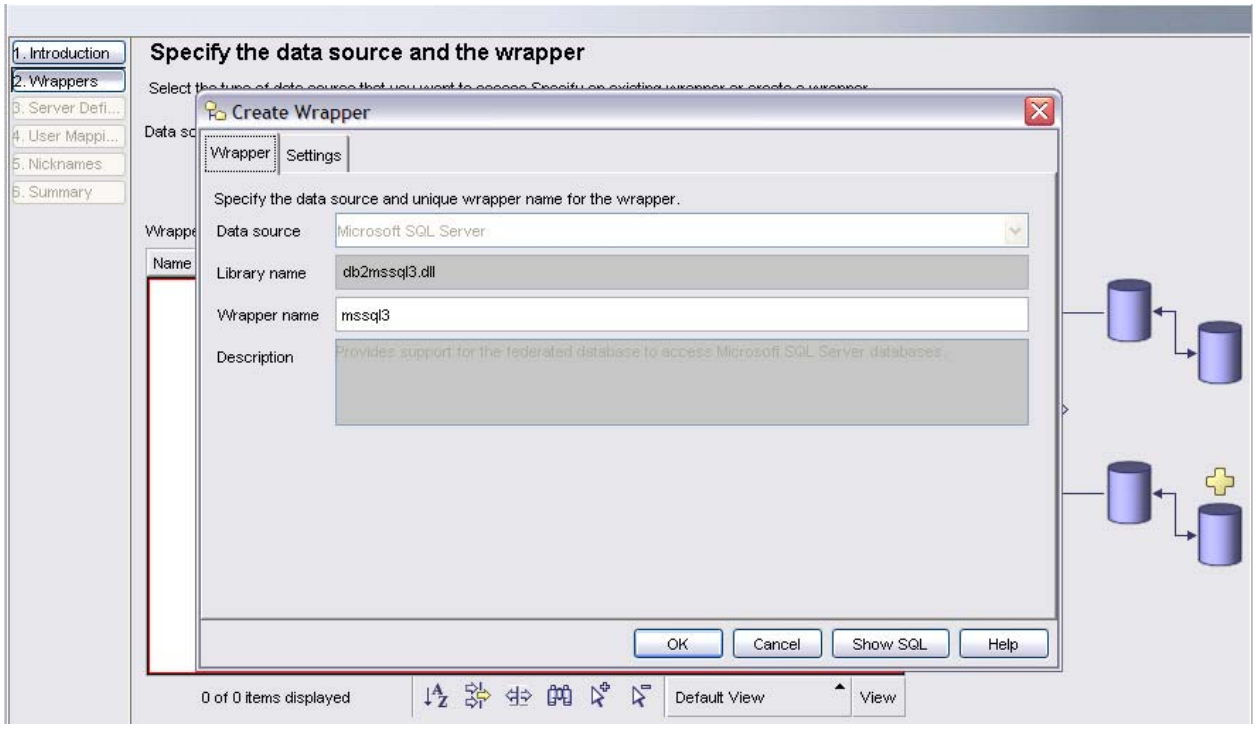

Figure 5. Create Wrapper interface

An action output window is then displayed to show that the wrapper configuration object was created successfully. After that, a newly created wrapper appears in the Create Federated Objects window, as shown in Figure 6.

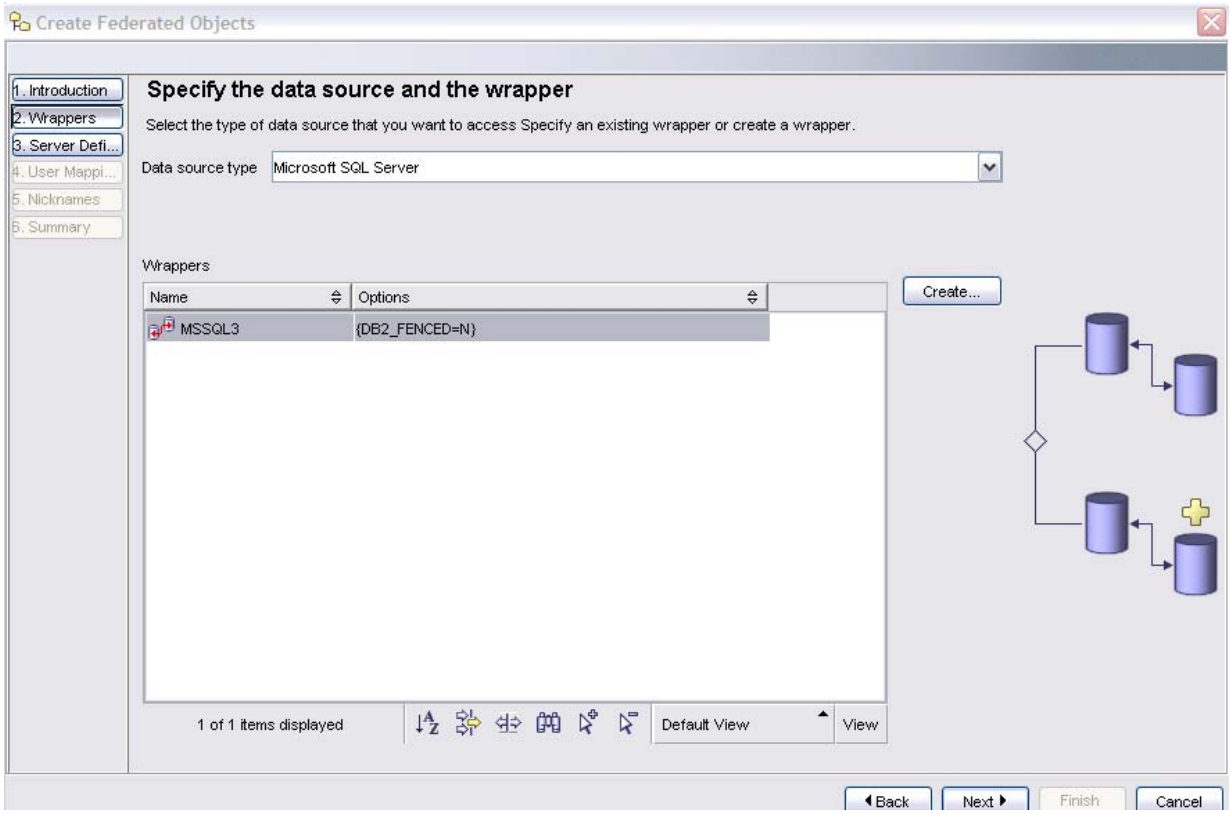

Figure 6. SQL Server wrapper created successfully

Click **Next** to advance to the next step of creating the server definition. The server definition identifies the name, type and location of the target-database server. Figure 7 shows the server-definition creation interface.

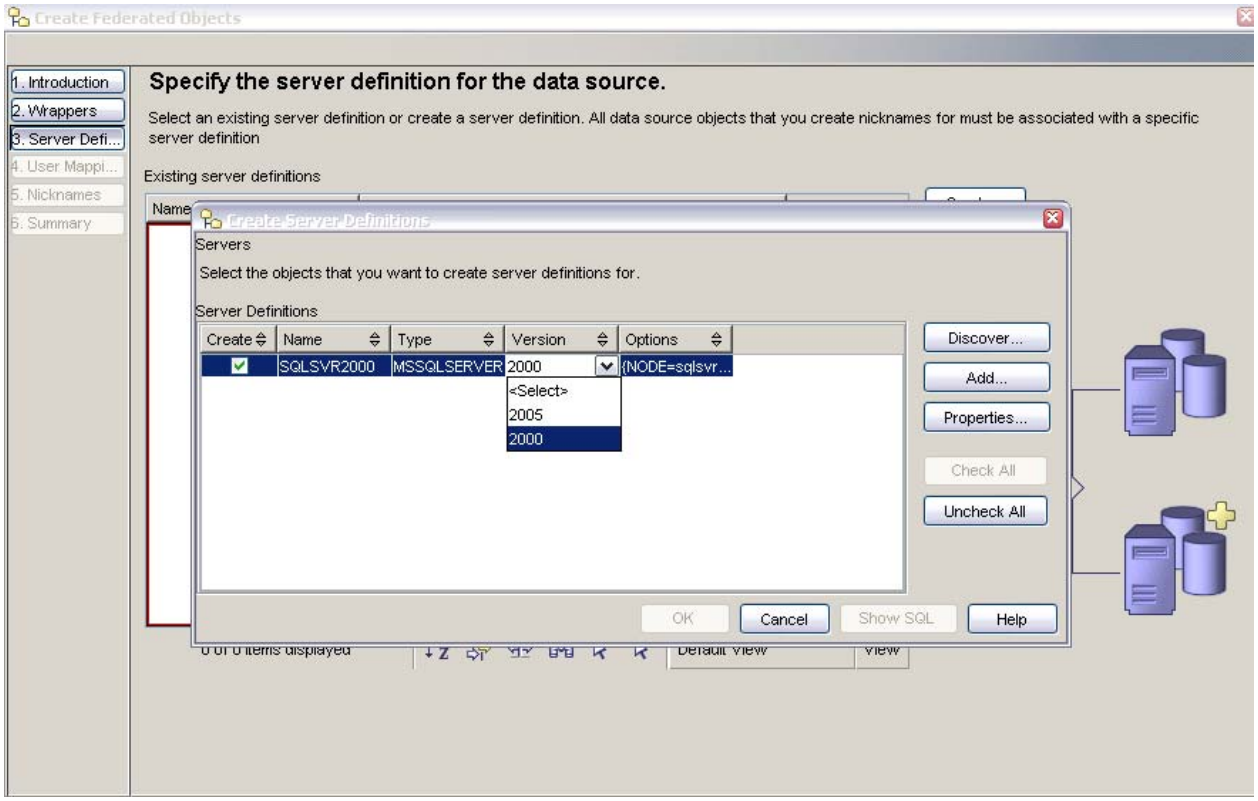

Figure 7. Server-definition interface

For SQL Server databases, the database is identified with an ODBC system data-source name (DSN). If the system DSN has already been created, you can click **Discover** to find and select the existing ODBC DSN. In this example, a system DSN named sqlsvr2000 is already created. (**Note:** Refer to Appendix A for the steps involved in creating an ODBC system DSN for a SQL Server database.)

After you identify the ODBC system DSN, select the version (**2000**). Then, click **Properties** to review and possibly change the settings for the server definition.

Figure 8 displays the Server Definition settings dialog.

Click the **Settings** tab and add a setting for the DBNAME option — in this case, the SQL Server database name that needs to be entered is *db2wii*.

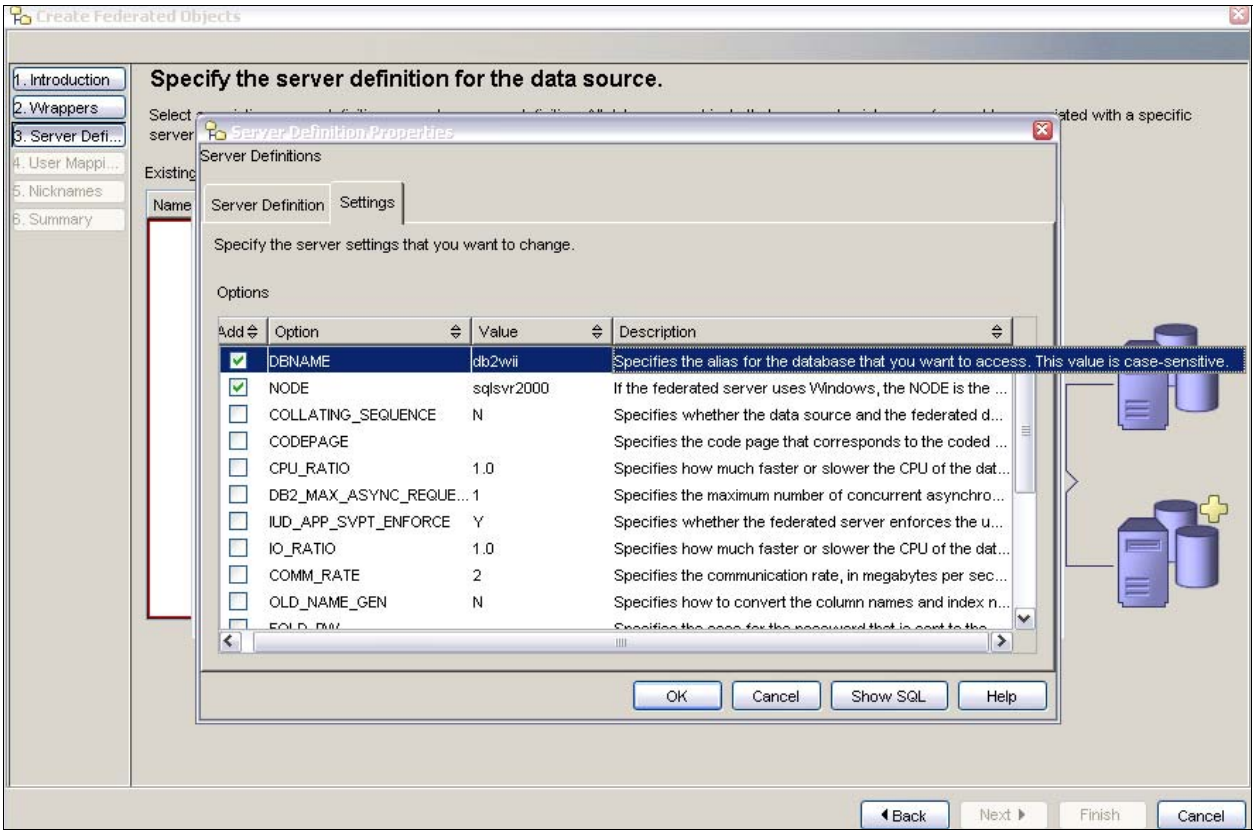

Figure 8. Server-definition settings

Click **OK** to finalize this setting. Then, on the Create Server Definition panel, click **OK**, which generates and runs the following SQL statement:

```
CREATE SERVER sqlsvr2000
  TYPE mssqlserver
  VERSION '2000'
  WRAPPER MSSQL3
  OPTIONS(ADDNODE 'sqlsvr2000', DBNAME 'db2wii')
```

Another Action Output panel is displayed to show whether the creation of the server object was successful. (**Note:** For brevity purposes, this panel is not shown here.)

With the server definition now complete, click **Next** to advance to the user-mapping interface.

Click **Create** to bring up the user-mapping dialog (see Figure 9).

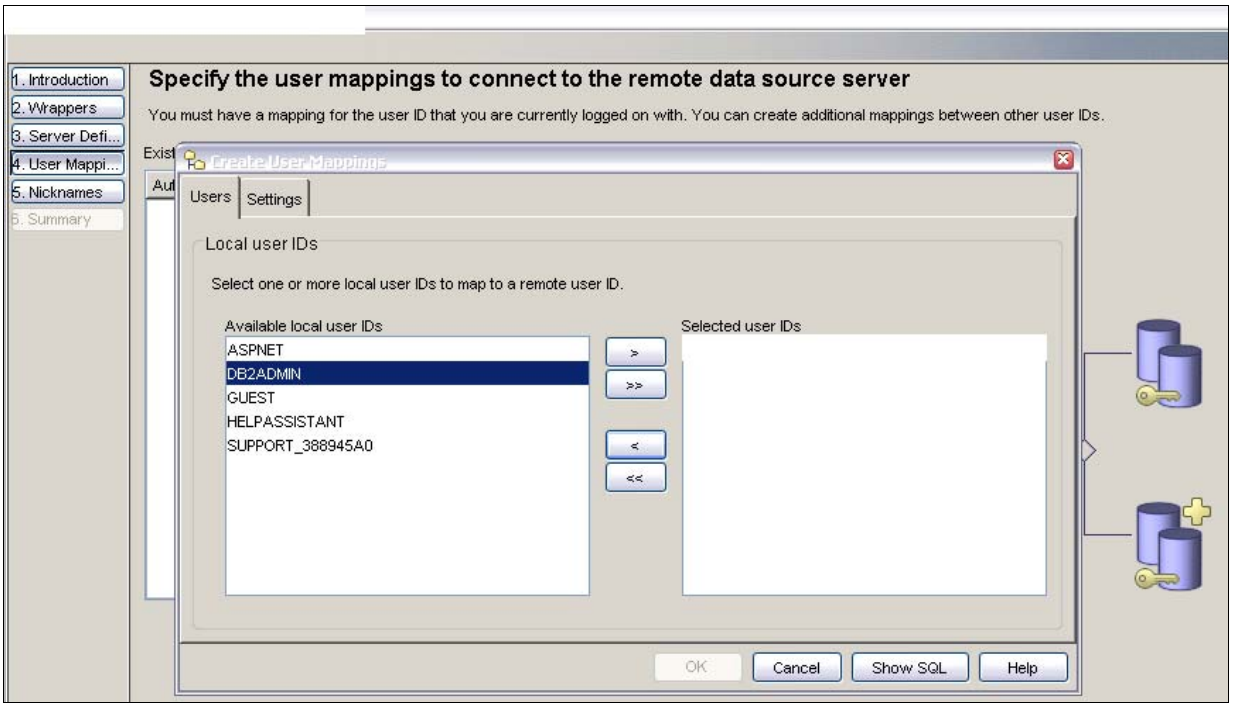

Figure 9. User-mapping interface

The user mapping associates one of the local user IDs that is authorized to the federated database, FEDDB, with the target data source.

On this first panel, the DB2ADMIN local user ID is highlighted; you select it for the mapping by clicking the **>** button to move it over to the *Selected user IDs* box.

After selecting DB2ADMIN, click the **Settings** tab to access the user-mapping settings (see Figure 10).

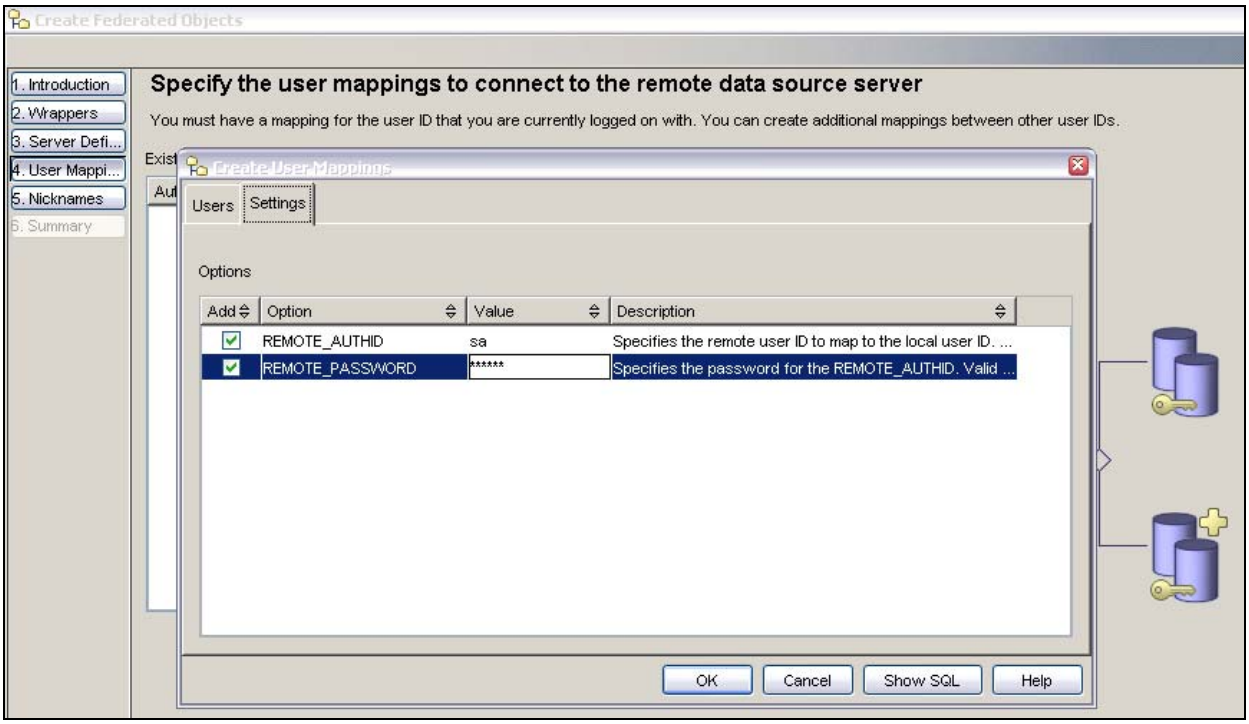

Figure 10. User-mapping settings

In the Settings tab, you identify which SQL server database user ID to associate with the local DB2 user ID (DB2ADMIN). For this example, you specify the SQL server database user `sa` for the REMOTE_AUTHID setting.

> The REMOTE_PASSWORD setting value is set to the password string for the `sa` user ID. When DB2ADMIN connects to the specified SQL server database, the credentials for SQL server user `sa` are used to establish the database connection.

Click **OK** to complete the user-mapping creation. This action results in the following SQL statement being created and run:

```
CREATE USER MAPPING FOR DB2ADMIN
      SERVER sqlsvr2000
      OPTIONS (ADD REMOTE_AUTHID 'sa', REMOTE_PASSWORD '******')
```

After an Action Output window confirms that the user-mapping object is created successfully, click **Next** on the User Mappings window to advance to the final step of creating nicknames. Figure 11 shows the Nickname-definition interface (including the Discover dialog).

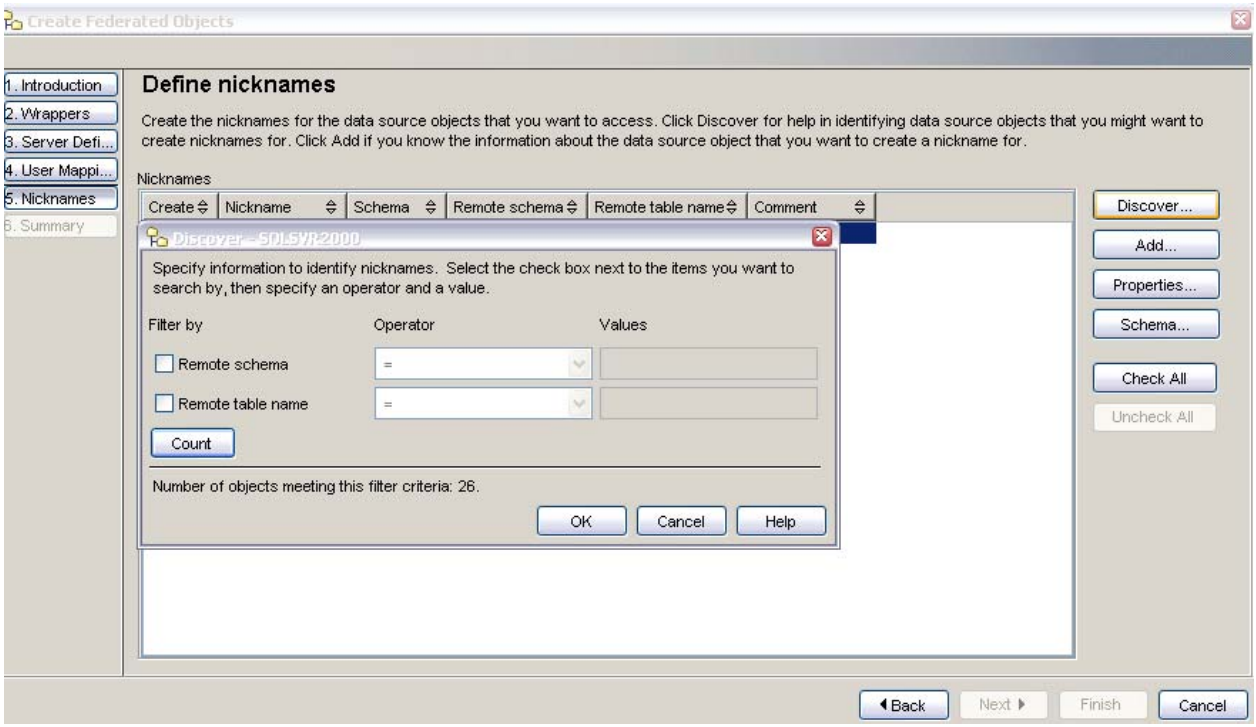

Figure 11. Nickname-discover interface

Clicking **Discover** is the easiest method for defining nicknames because it allows you to graphically select the SQL Server object that needs a nickname (instead of manually entering the name of the SQL Server database object).

As you can see, the Discover dialog lets you filter a specific set of database objects. In this case, no filtering is specified. Instead, you can click **Count** to determine how many objects are in the SQL Server database — this database has 26 objects. Click **OK** to display the list of these objects that reside in the SQL Server database (see Figure 12).

In Figure 12, notice that, by default, all the SQL Server objects are selected for nickname creation. It is not necessary to have federated nicknames for all the objects in this example, so you can click **Uncheck All** to clear all the SQL Server objects.

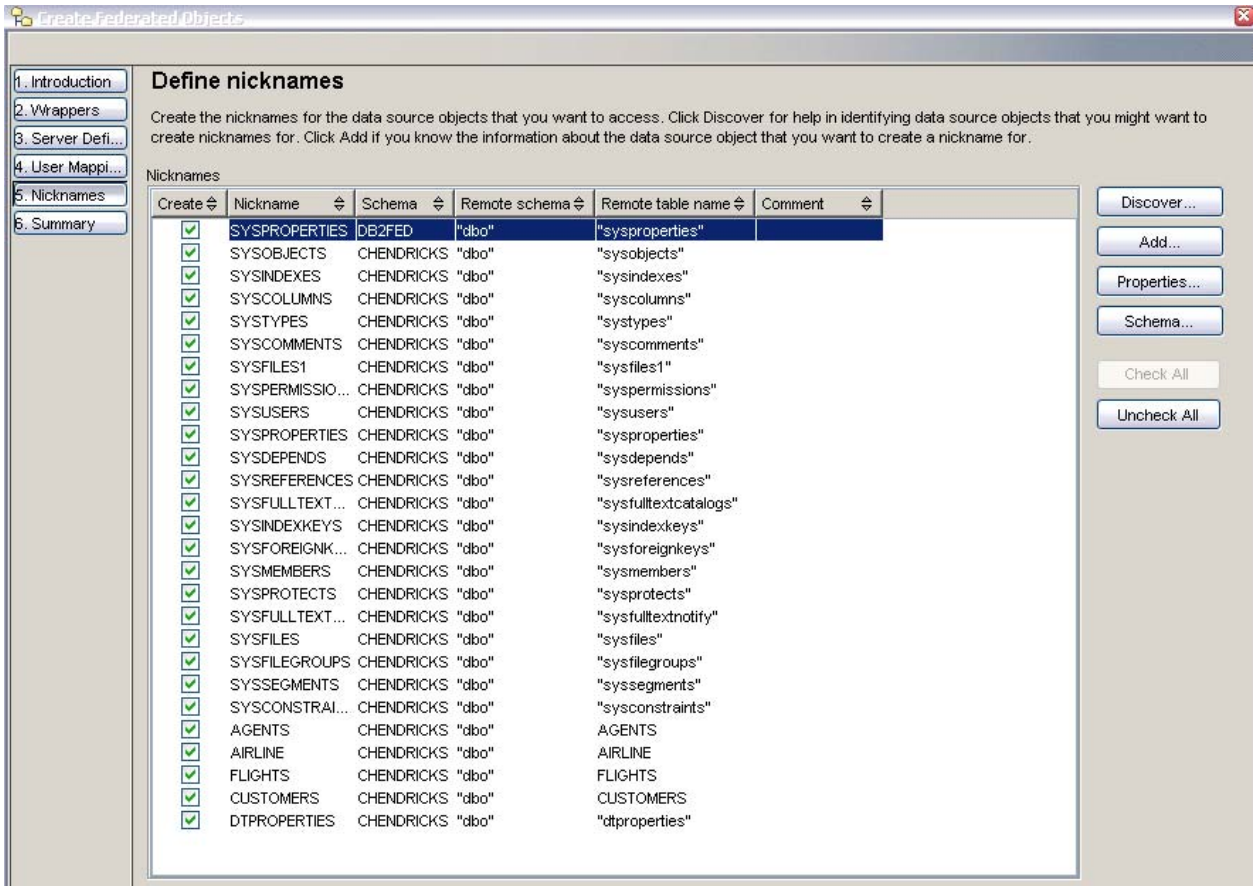

Figure 12. Discovered-object list

After clearing all the objects, highlight only the four SQL Server tables (Agents, Airline, Flights and Customers) shown in Figure 13 to have nicknames created for them.

Before creating the nicknames, click **Schema** to specify that the nicknames are created in a DB2 schema named DB2FED in the FEDDB database.

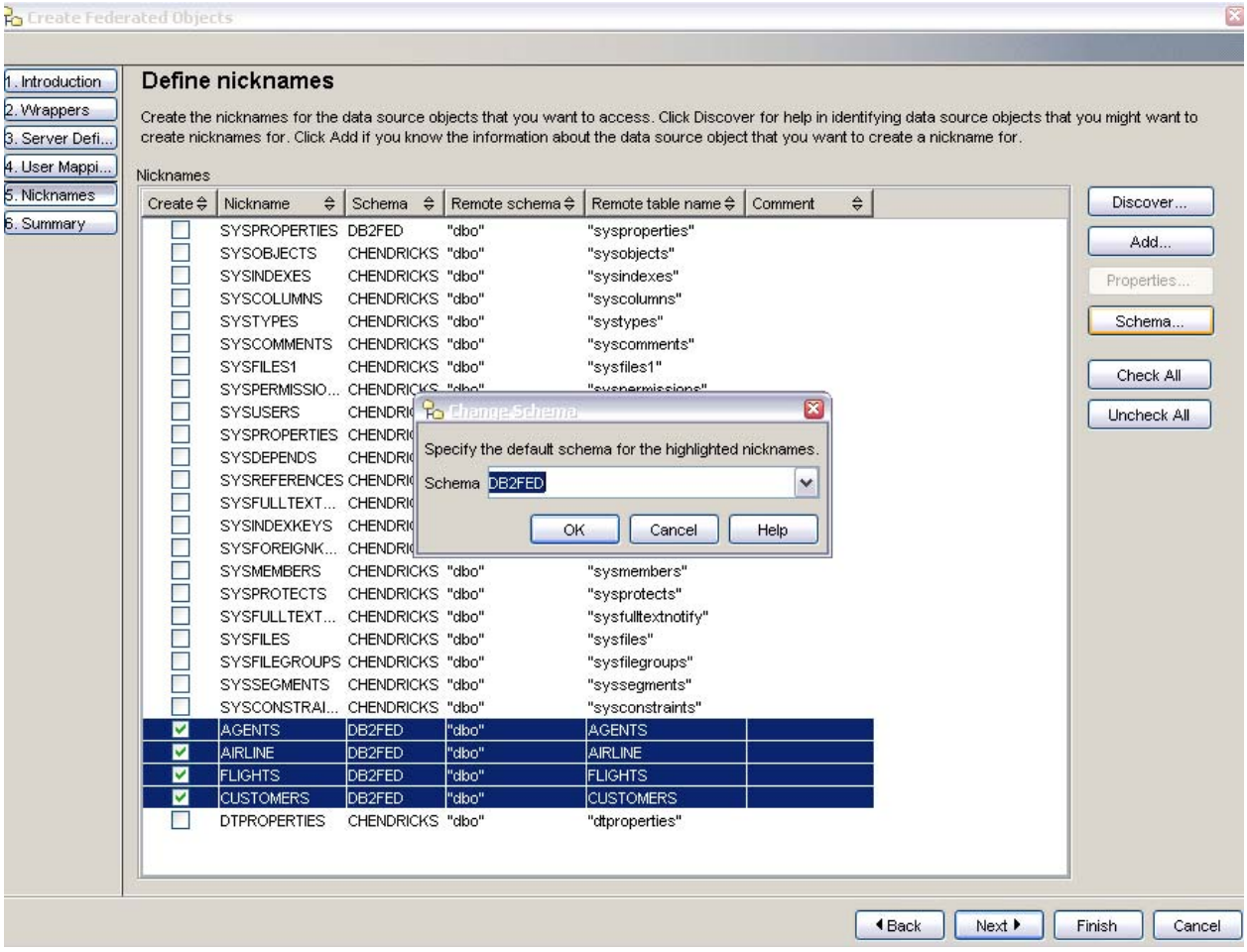

Figure 13. Selected-nickname candidates

Click **OK** to change the schema name.

Click **Next** to display the review panel (see Figure 14) before creating the nicknames.

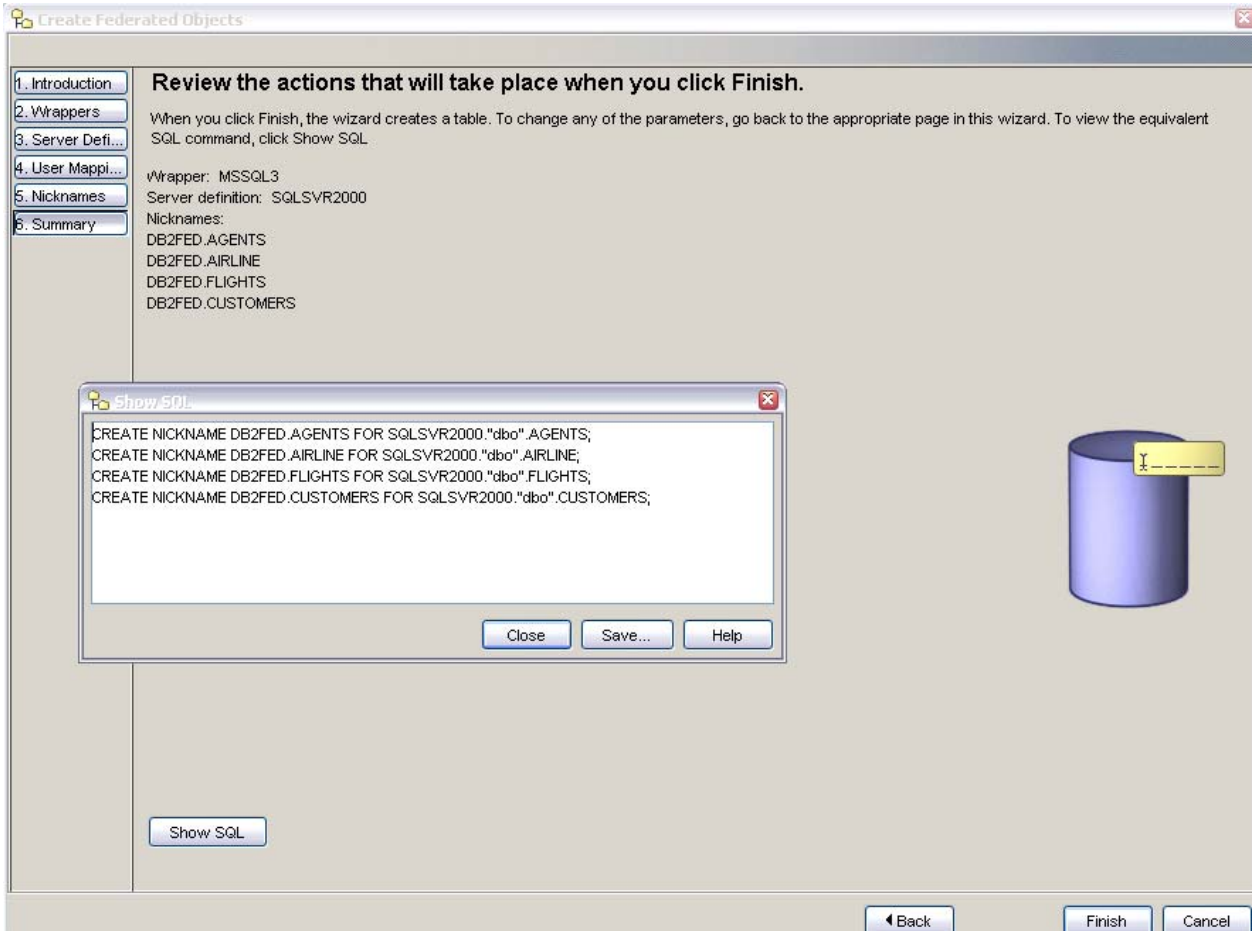

Figure 14. Nickname-review window

> **Note:** The Show SQL display in Figure 14 contains the SQL statements that Control Center generates for the Nickname creation. The CREATE NICKNAME statements are included here for completeness:
>
> ```
> CREATE NICKNAME db2fed.agents FOR sqlsvr2000.”dbo”.agents
>    CREATE NICKNAME db2fed.airline FOR sqlsvr2000.”dbo”.airline
>    CREATE NICKNAME db2fed.flights FOR sqlsvr2000.”dbo”.flights
>    CREATE NICKNAME db2fed.customers FOR sqlsvr2000.”dbo”.customers
> ```

Click **Finish** to run these Nickname statements. An Action Output window is again displayed to show the status of the nickname creation.

After you have successfully created the nicknames, refresh the DB2 Control Center window by clicking **View -> Refresh**.

Expand the **Federated Database Objects** folder to review the federated configuration objects that you just created (see Figure 15).

Navigating to the Nicknames folder allows you to test one of the newly created nicknames. You can verify the nickname by right-clicking the object and selecting **Open**, which results in the WebSphere Federation Server connecting to the specified SQL Server database server. Then, the Open Nickname task uses the

language that is defined in the wrapper to return all of the rows from the table identified by the DB2FED.AGENTS nickname. This task produces the Open Nickname output window found on the right-hand side of Figure 15.

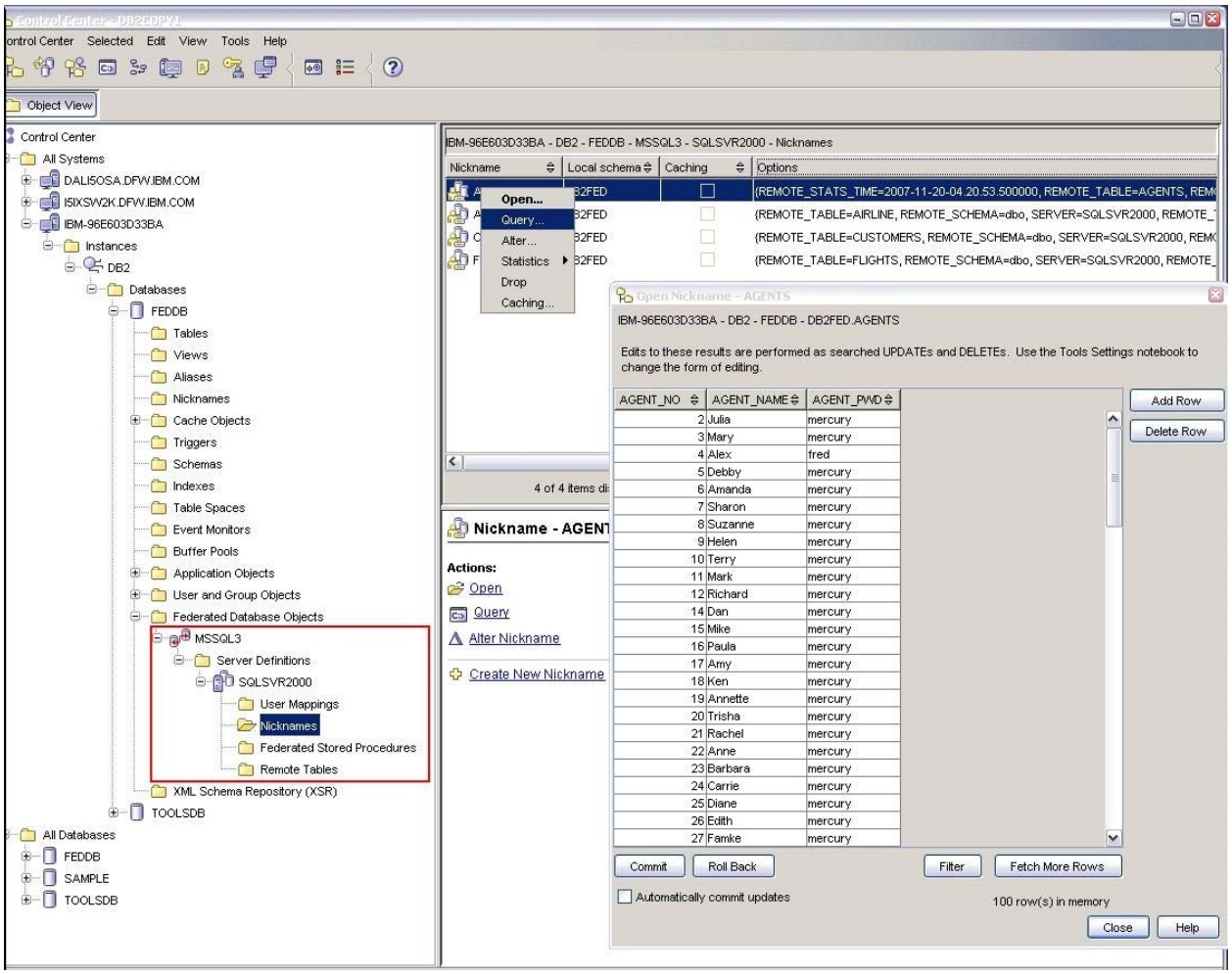

Figure 15. Control Center Federated-objects and open-nickname interfaces

You can cause any SQL interface or application that uses the WebSphere Federation Server can retrieve data from the agents table in SQL Server in a similar fashion as the Open task — by just issuing the following simple SQL statements:

```
CONNECT TO FEDDB USER DB2ADMIN USING password
SELECT * FROM agents
```

In the next section, you will learn how to set up an i5/OS application to interact with WebSphere Federation Server to access this same SQL Server database.

Enabling i5/OS and RPG application access

Before discussing the RPG programming, it is important to enable the i5/OS system to access the federated database server. This configuration involves the creation of a DB2 Relational Database Directory Entry (assuming that a TCP/IP network connection already exists between the i5/OS system and the server that houses WebSphere Federation Server).

1. To create this entry, you need to know the TCP/IP address of the server where WebSphere Federation Server is installed, as well as the database name (FEDDB in this case). Then, you can run the following CL command on an i5/OS system:

   ```
   ADDRDBDIRE RDB(FEDDB) RMTLOCNAME('9.67.25.230' *IP) PORT(50000)
   ```

After defining the directory entry, run a simple test to verify that the i5/OS system and WebSphere Federation Server can communicate. First, press F11 to verify that a Commitment Control setting other than *NONE is used for the STRSQL session. Then run the following SQL statements by using interactive SQL (that is, the STRSQL command), which is the easiest way to test this from an i5/OS system:

```
RELEASE ALL
COMMIT
CONNECT TO FEDDB USER myuser USING passwd
SELECT * FROM db2fed.agents
COMMIT
CONNECT RESET
```

Note: It is important to authorize the user ID specified on the CONNECT statement (myuser) to the DB2 database (that is, FEDDB) that was created to house the WebSphere Federation Server configuration objects. The user ID on the CONNECT statement might not match the i5/OS user ID

If the STRSQL command is not available on your server, the DB2 Query Manager and SQL Development Kit licensed product is not installed. This product also needs to be installed if you want to embed SQL in an RPG program similar to the one shown in Figure 16.

If these SQL statements have been completed successfully, all the basic plumbing is in place to allow the i5/OS applications to use WebSphere Federation Server for heterogeneous data access.

As mentioned, Figure 16 contains a snippet of an RPG application that uses embedded SQL to retrieve data from the SQL Server table using WebSphere Federation Server. Notice that the SQL statements look identical to those that you would use to access a local DB2 table. With the upfront configuration complete, you can code basic SQL statements without concern for creating wrappers or server definitions. You only need to know the nickname of the SQL Server table that you must access and the name of WebSphere Federation Server. The federated server also takes care of converting the ASCII data into EBCDIC.

```
FFRS000DF  CF   E             WORKSTN INFDS(WSDS)
D*
d agentsRow     e ds                  extname(Agents)  qualified
D*
. . .
D*
D  CONNECT        DS
D  USERID                  1     13    INZ('MYUSER')
D  PASSWORD               14     22    INZ('*********')
D*
D LOGON          UDS
D  AGENTC                  1      9
D  AGENT#                  1      9  0
I*
D agnmbr          s                        like(agentsRow.agent_no)
D agname          s                        like(agentsRow.agent_name)
D agpass          s                        like(agentsRow.agent_pwd)
D agentKey        s                        like(agentsRow.agent_no)
. . .
C                   SETOFF                                      505152
C                   MOVE      ' '           MSGID             7
C                   MOVE      *BLANKS       MSGDTA           80
C                   MOVE      @PGM          MSGQUE           10
C                   MOVE      @FALSE        @EXIT             1
C                   MOVE      @PGM          PGMQ
C*Connect to Federated database
C/EXEC SQL
c+   CONNECT TO FEDDB USER :userid USING :password
C/END-EXEC
. . .
C     AGNTNM        IFEQ      *BLANKS
C                   SETON                                         5051
C                   MOVE      'FRS0050'     MSGID
C                   EXSR      @SNDMS
C                   END
C     PASSWD        IFEQ      *BLANKS
C                   SETON                                         5052
C                   MOVE      'FRS0055'     MSGID
C                   EXSR      @SNDMS
C                   GOTO      ERREND
C                   END
C                   MOVE      *BLANKS       AGNKEY
C     ' '           CHECKR    AGNTNM        AGNLEN                            C
MOVEL     AGNTNM         AGNNAM
C*    AGNKEY        CHAIN     AGENTSL                           44
C/EXEC SQL
c+   SELECT * INTO :agentsRow
c+   FROM db2fed.agents
c+   WHERE agent_name = :agnnam and agent_pwd = :passwd
C/END-EXEC
      /free
       if %subst(sqlState:1:2) <> '00';
      /end-free
. . .
```

Figure 16. RPG Code snippet that shows access to the SQL Server table

The creation of the RPG program object is also straight-forward, as Figure 17 shows. You specify WebSphere Federation Server on the RDB parameter, along with the user ID and password for that server.

Again, you must also specify these same parameters if the RPG program has to access a DB2 for i5/OS table in another partition or system.

You must also set other compiler settings, such as *Date Format* and *Commitment Control level*, to values that the other DB2 products support.

```
CRTSQLRPGI OBJ(FLGHT400/AGTPASS)
           SRCFILE(*LIBL/QRPGLESRC) SRCMBR(AGTPASS)
           RDB(FEDDB) USER(myuser) PASSWORD()
           OPTION(*XREF *SQL)
           DATFMT(*ISO) DATSEP(*JOB)
```

Figure 17. RPG compile command example

Joining SQL Server and DB2 data

If you want to be able to join your DB2 for i5/OS tables with tables in the federated SQL Server database, you must also register the DB2 for i5/OS tables in WebSphere Federation Server. Without WebSphere Federation Server, an i5/OS SQL statement can only contain references to DB2 objects that reside on a single server.

This process involves configuring a connection from the federation server to DB2 for i5/OS. This step is needed because WebSphere Federation Server performs the join of the SQL Server and DB2 data, the joined results are then returned to the i5/OS application.

1. The easiest way to set up a connection to DB2 for i5/OS is to use the Configuration Assistant. You can launch this configuration wizard from the Start menu: **IBM DB2 -> Setup tools -> Configuration Assistant**. The Configuration Assistant displays the initial panel (see Figure 18), which is used to capture the communication protocol.

Select **TCP/IP** as the communications protocol. If there is an IBM OS/400® host-system input option on the Communication Protocol panel, then you should also select it before advancing. Click **Next**.

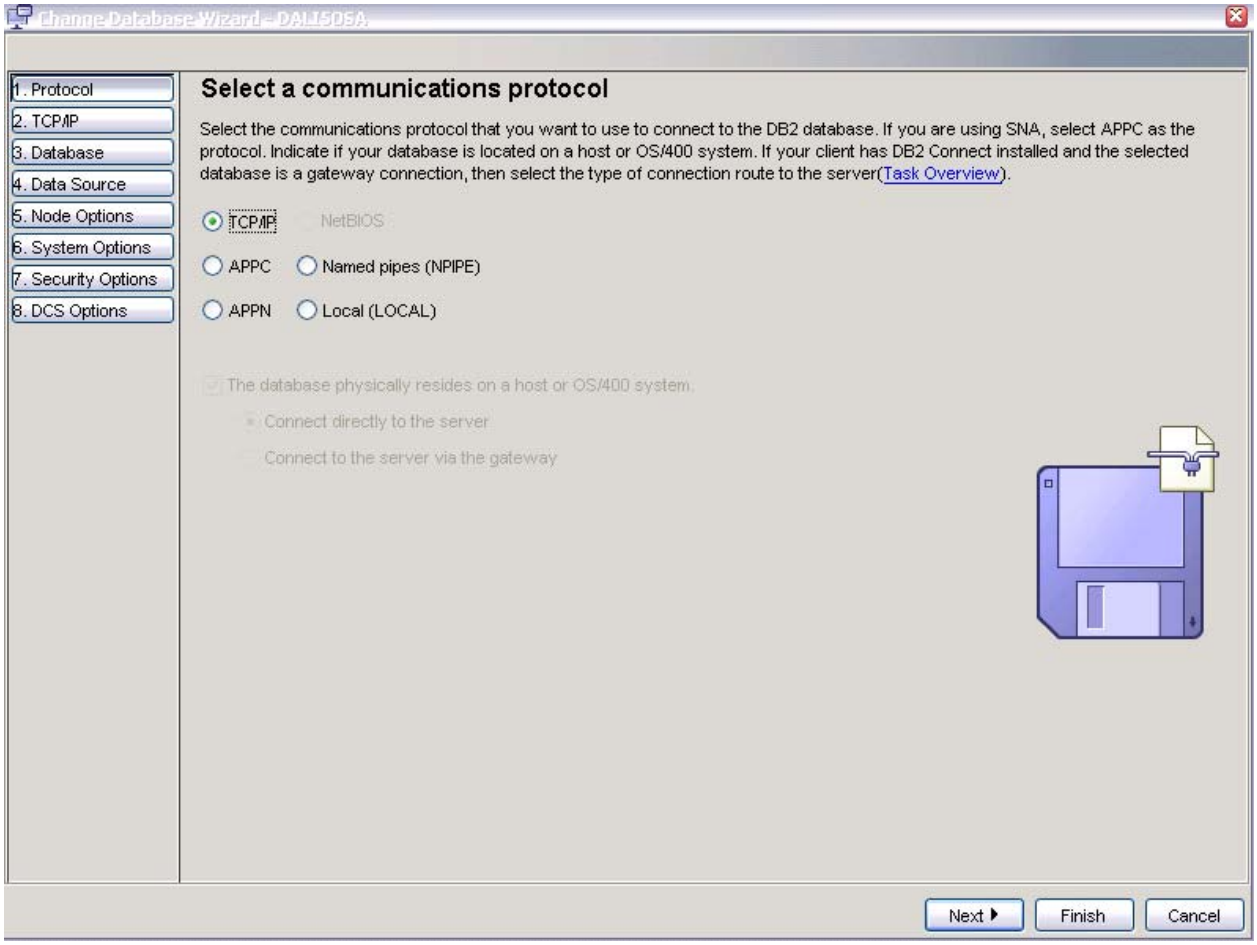

Figure 18. Configuration Assistant communication protocol

The connection parameters for the TCP/IP connection are displayed in Figure 19.

For the host name of the i5/OS system, type `dali5osa.dfw.ibm.com`; and for the default value for the Port number, type `446`. Click **Next** to continue the configuration.

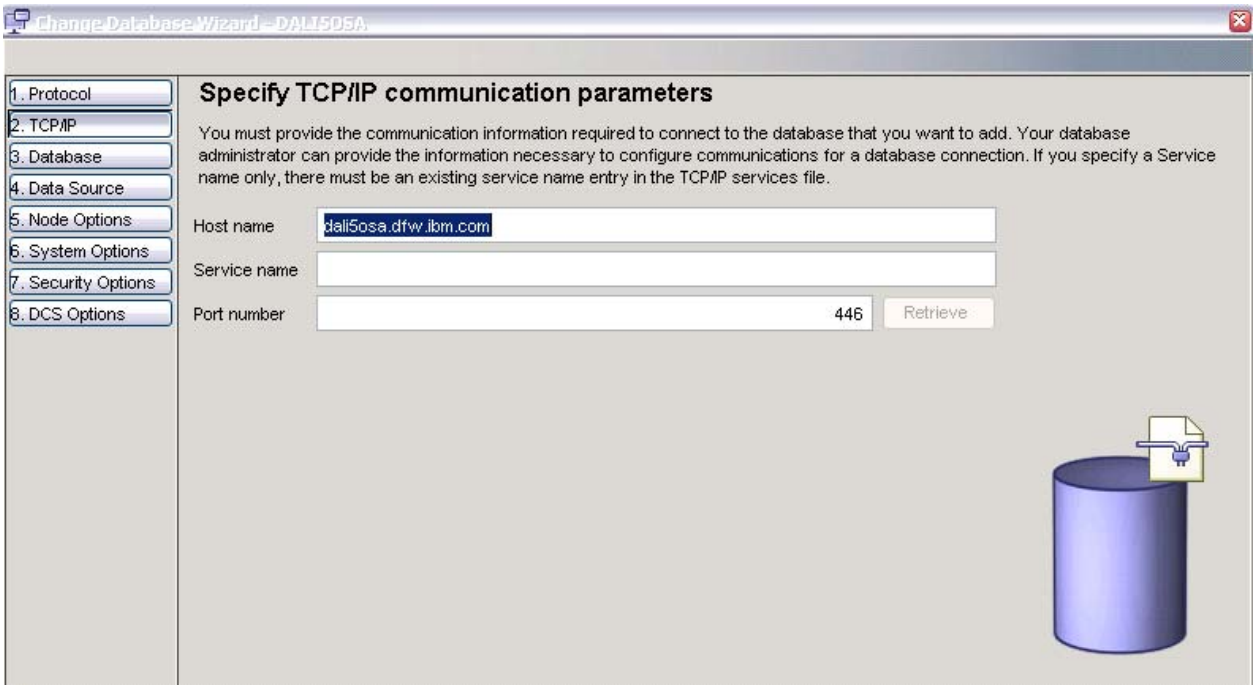

Figure 19. Configuration Assistant TCP/IP parameters

The Database input panel only requires that you specify the database name, which you can obtain from the i5/OS relational database directory by using the Display Relational Database Directory (DSPRDBDIRE) command. The relational database name is the name that is associated with the local entry (*LOCAL). The alias name is automatically set to the database name (see Figure 20).

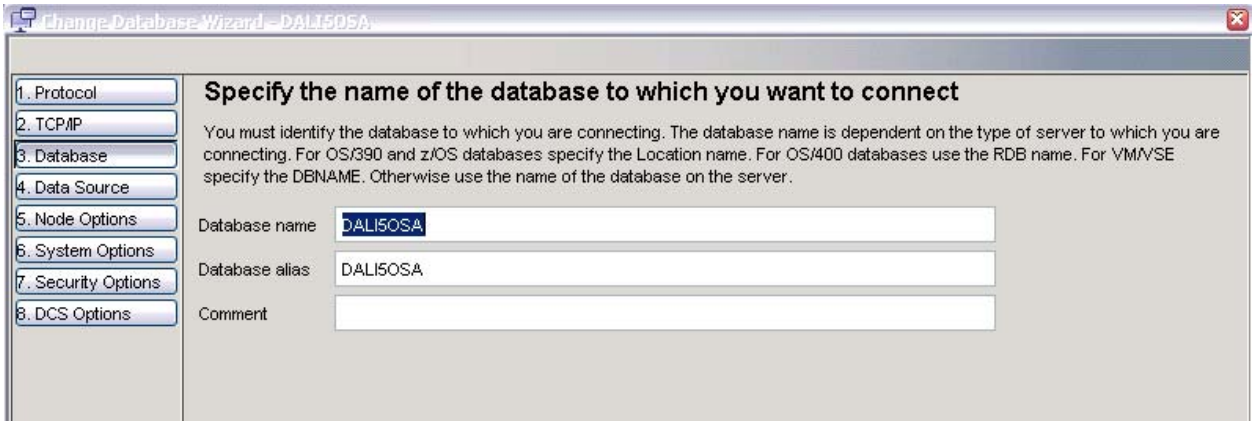

Figure 20. Configuration Assistant database name

After selecting **Next**, the data-source configuration screen is displayed — this is an optional step and is not shown here.

After this optional step, configuring the node option is the next phase of the configuration process. The operating system is the only node option that you must specify — OS/400 is the value that is needed for a connection to a DB2 for i5/OS database. Click **Next** to advance to the system options.

Figure 21. Configuration Assistant node options

The System name is the only thing you need to specify on the options panel shown in Figure 22. The system name value matches the TCP/IP host name (dali5osa.dfw.ibm.com) that you specified in the earlier step.

After specifying the system name, click **Next**.

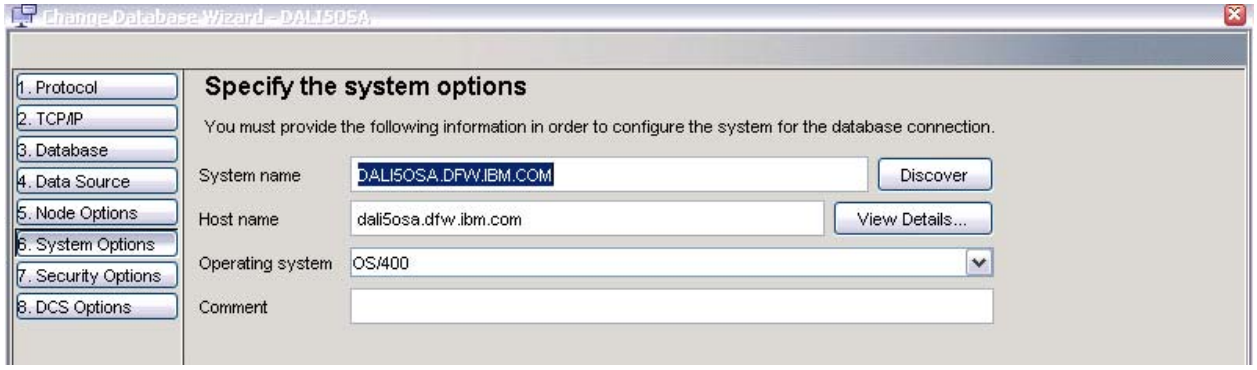

Figure 22. Configuration Assistant system options

Configuring the security attributes for the database connection is the next step. As you can see in Figure 23, several security options are available for the DB2 for i5/OS database connection. Select the option that meets your company's security policy and then click **Next**.

Figure 23. Configuration Assistant security options

(Optional) The next step is optional, because the database connection server (DCS) options are not needed for federated access, so click **Finish** to create the database connection.

> If the connection is created successfully, the following window (see Figure 24) is displayed, providing an opportunity to test the newly created connection. Testing the connection to the DB2 for i5/OS server is recommended before moving any further in the configuration process.

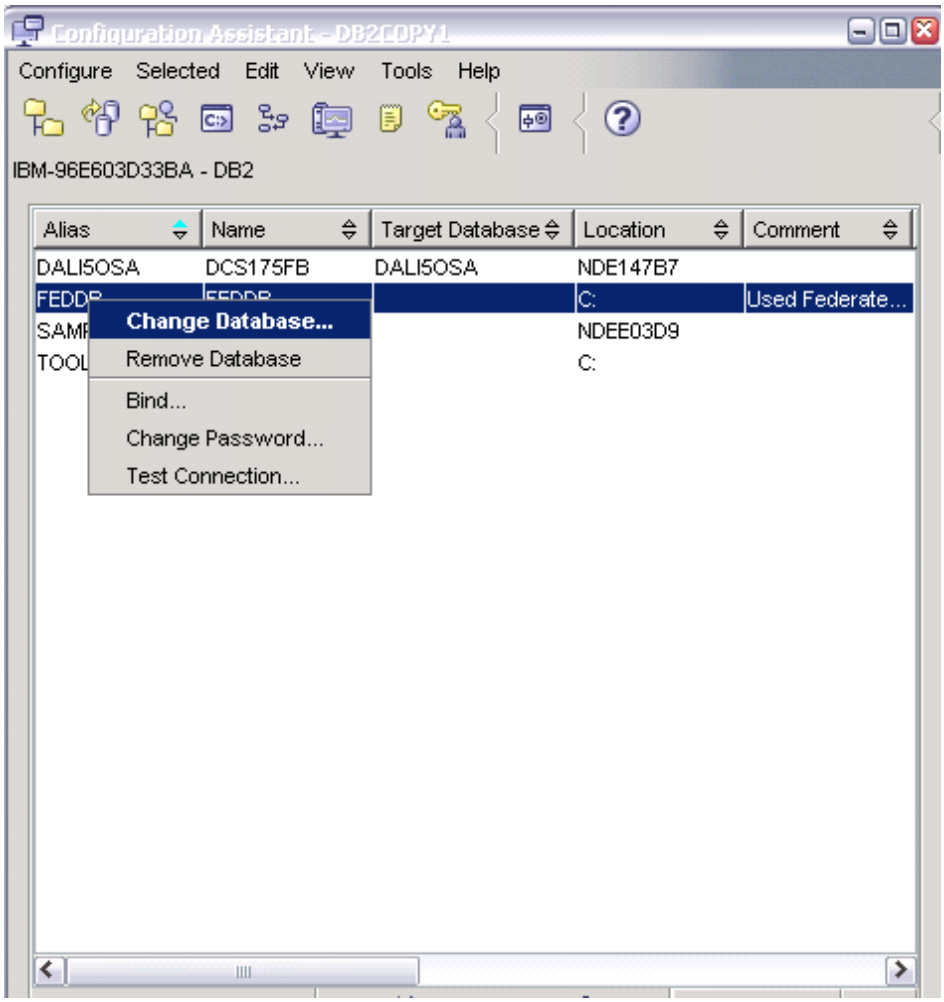

Figure 24. Database-connection options

> Assuming that the connection test has been completed successfully, the next step is creating the federated configuration objects for the DB2 for i5/OS object that you need to join to the Microsoft SQL Server table.

You can create the configuration objects by returning to the Federated Database Objects folder in Control Center.

Just as with the SQL Server configuration, the first step involves creating the wrapper to define the language used for accessing the DB2 for i5/OS database. Start the wrapper-creation process by right-clicking the **Federated Database Objects** folder and clicking **Create**.

Create the DRDA wrapper for a DB2 database by selecting **DB2** for the data-source type on the first

panel (Figure 25) and **DRDA** for the wrapper name (Figure 26) on the next panel. DRDA is the default wrapper value. Click **OK** to create the wrapper; this causes the following SQL statement to run:

```
CREATE WRAPPER DRDA LIBRARY 'db2drda.dll'
```

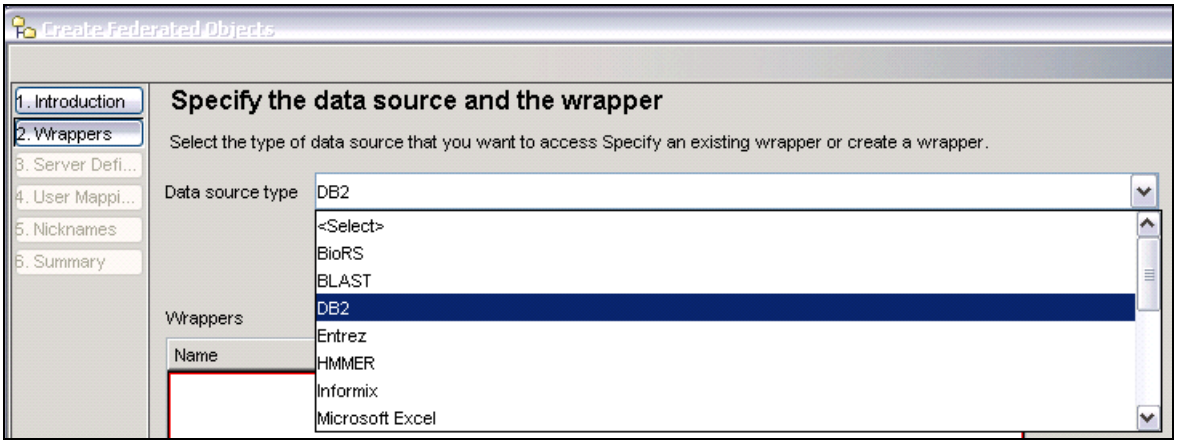

Figure 25. DB2 data source

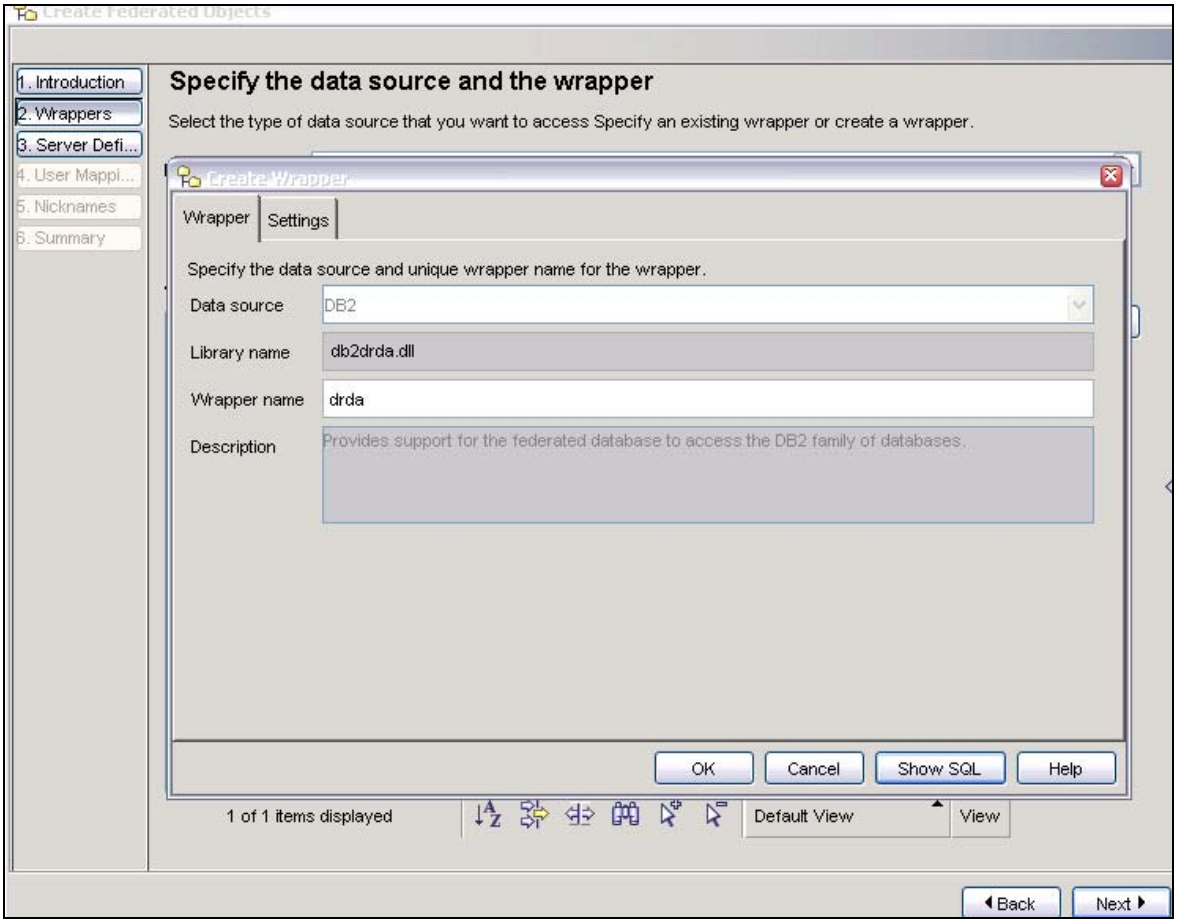

Figure 26. DB2 DRDA wrapper

After successfully creating the wrapper, click **Next** to advance to the server-definition interface (see Figure 27).

Click **Create** to launch the wizard for creating a DB2 for i5/OS server definition. Click **Discover** to find the database connection created earlier in this section.

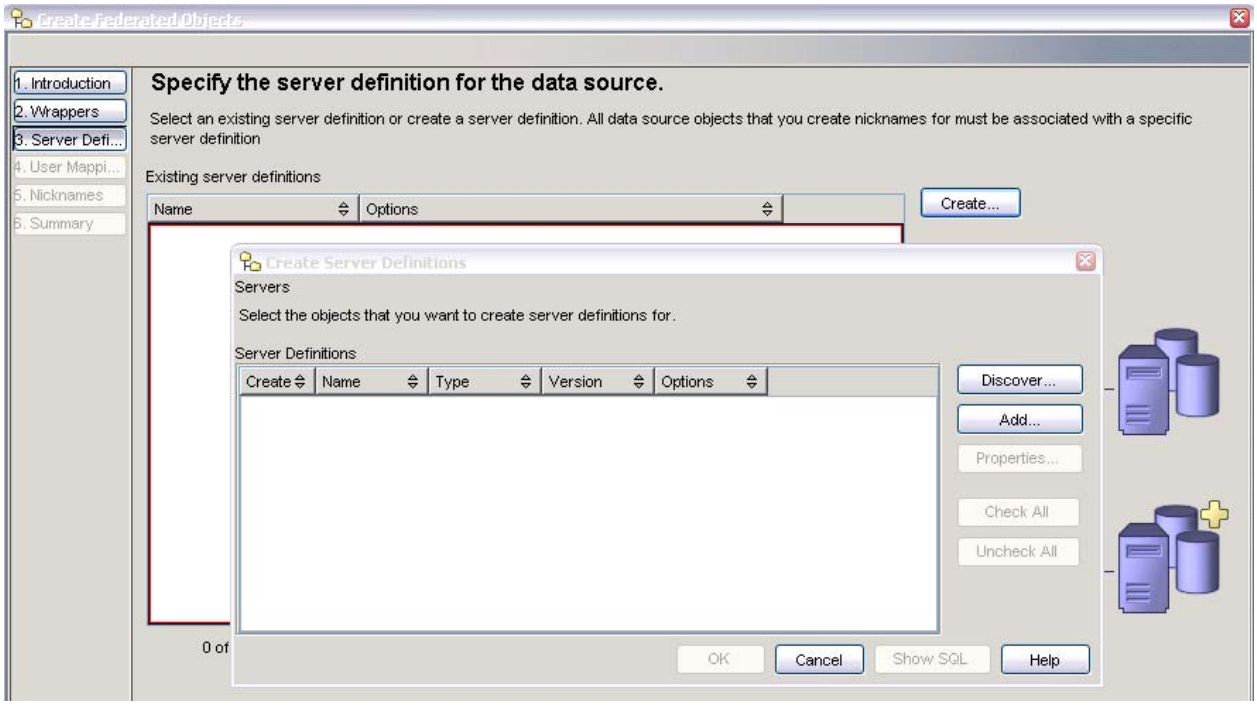

Figure 27. DB2 for i5/OS server definition

To identify the DB2 for i5/OS connection manually, click **Add** to open the window shown in Figure 28.

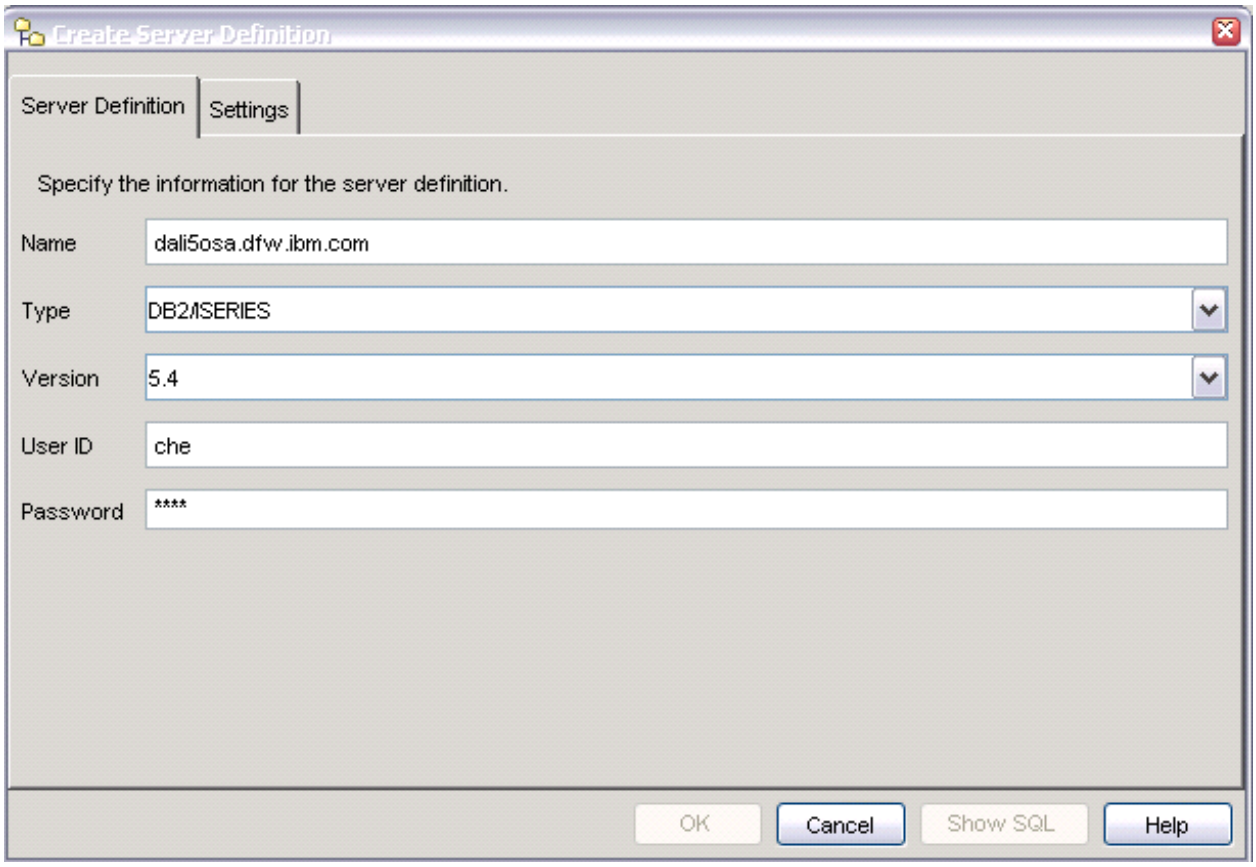

Figure 28. DB2 for i5/OS server-definition window

You can set the server name to any value. This example uses the name of the database connection that was created earlier in this section (dali5owa.dfw.ibm.com).

Specify **DB2/ISERIES** as the value for server type. Additionally, the version should match the operating-system version that is installed on the server.

The DRDA wrapper requires that you specify an i5/OS user ID and password when creating a server definition.

Click the **Settings** tab to access additional configuration options for the server (see Figure 29).

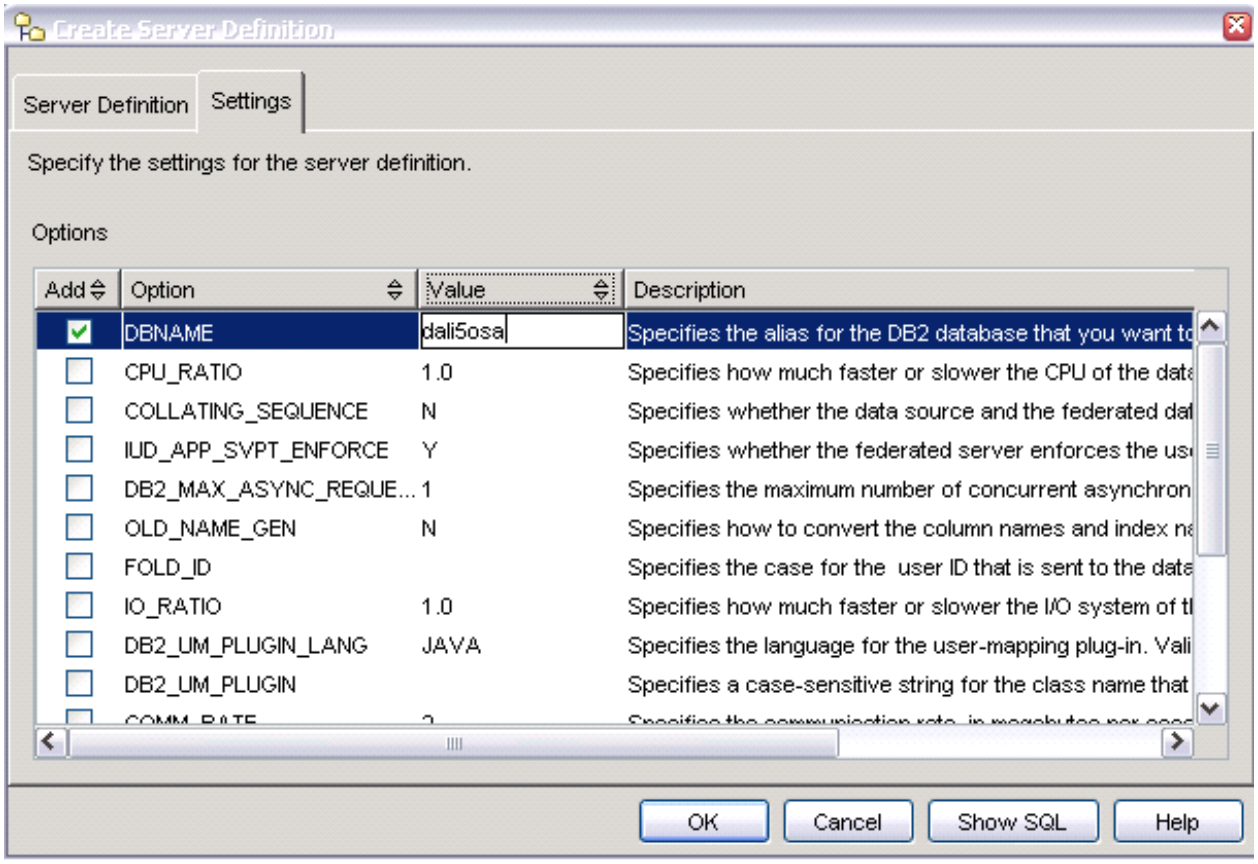

Figure 29. DB2 for i5/OS server-definition settings

Select the **DBNAME** option and enter `dali50sa`. Click **OK** to create the server definition, which generates and runs the following SQL statement:

```
CREATE SERVER “DALI5OSA.DFW.IBM.COM” TYPE DB2/400
   WRAPPER DRDA AUTHID che PASSWORD xxxx
   OPTIONS(ADD DBNAME ‘dali5osa’)
```

After creating the server definition, click **Next** to advance to the user-mapping step (see Figure 30), where a user who is authorized to the federation-server database is associated with an i5/OS user profile. In this example, CHENDRICKS is the authorized user to the federation server database.

Click the **Settings** tab (see Figure 31) to map the CHENDRICKS federated-database user ID to an i5/OS profile.

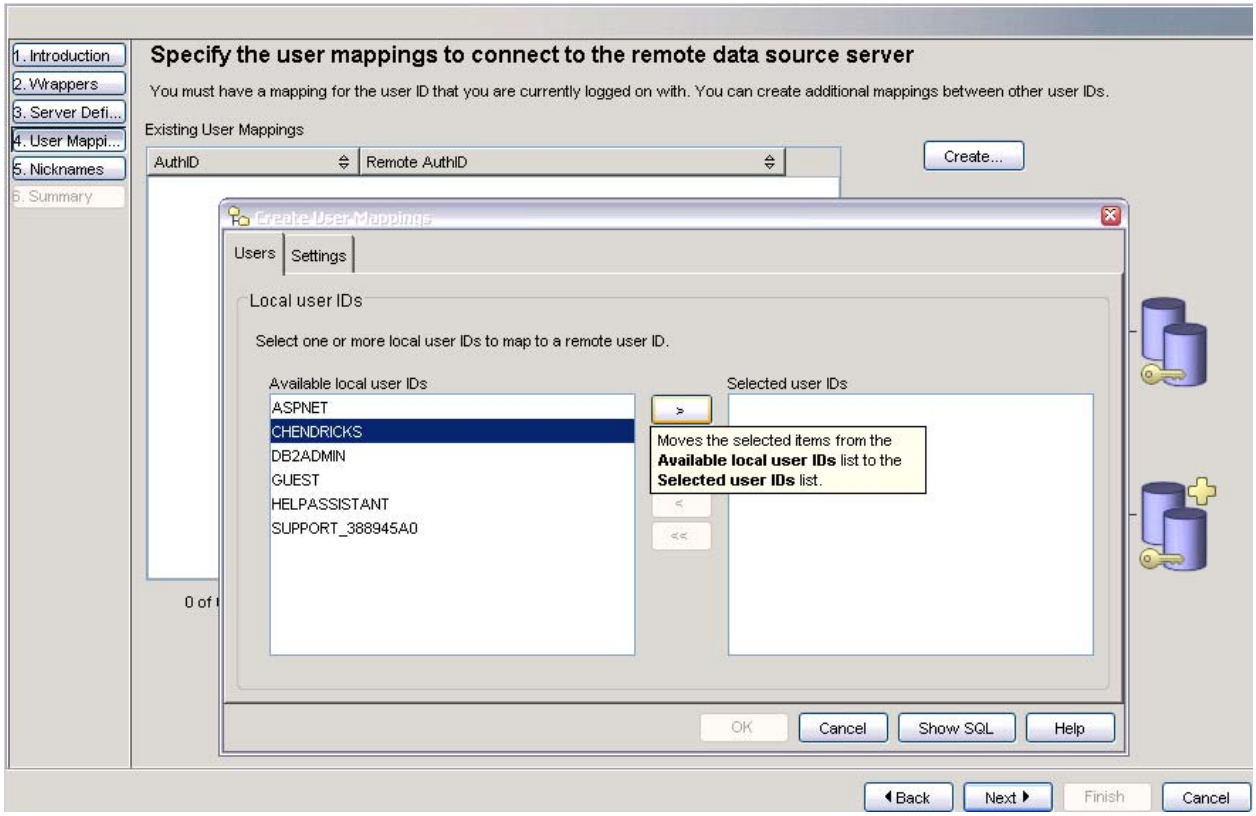

Figure 30. DB2 for i5/OS user mapping

For this example, select the *CHE* i5/OS user profile for the user-mapping configuration object.

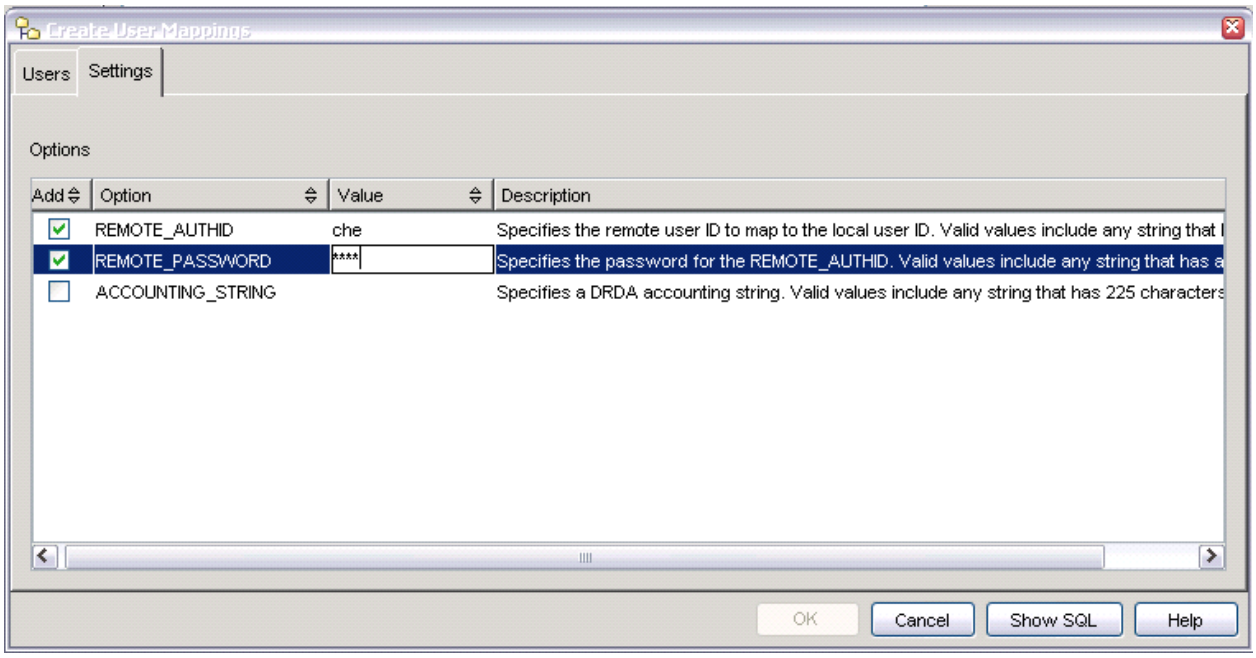

Figure 31. DB2 for i5/OS user-mapping settings

Click **OK** to complete the creation of the user-mapping object. Control Center uses the following SQL statement to create the object:

```
CREATE USER MAPPING FOR chendricks
  SERVER “dali5osa.dfw.ibm.com”
    OPTIONS (REMOTE_AUTHID 'che', REMOTE_PASSWORD '*****')
```

With the user mapping completed, nicknames are the only federated objects that you need to create for DB2 for i5/OS access.

Click **Next.**

On the Nicknames definition panel, click **Discover** (see Figure 32).

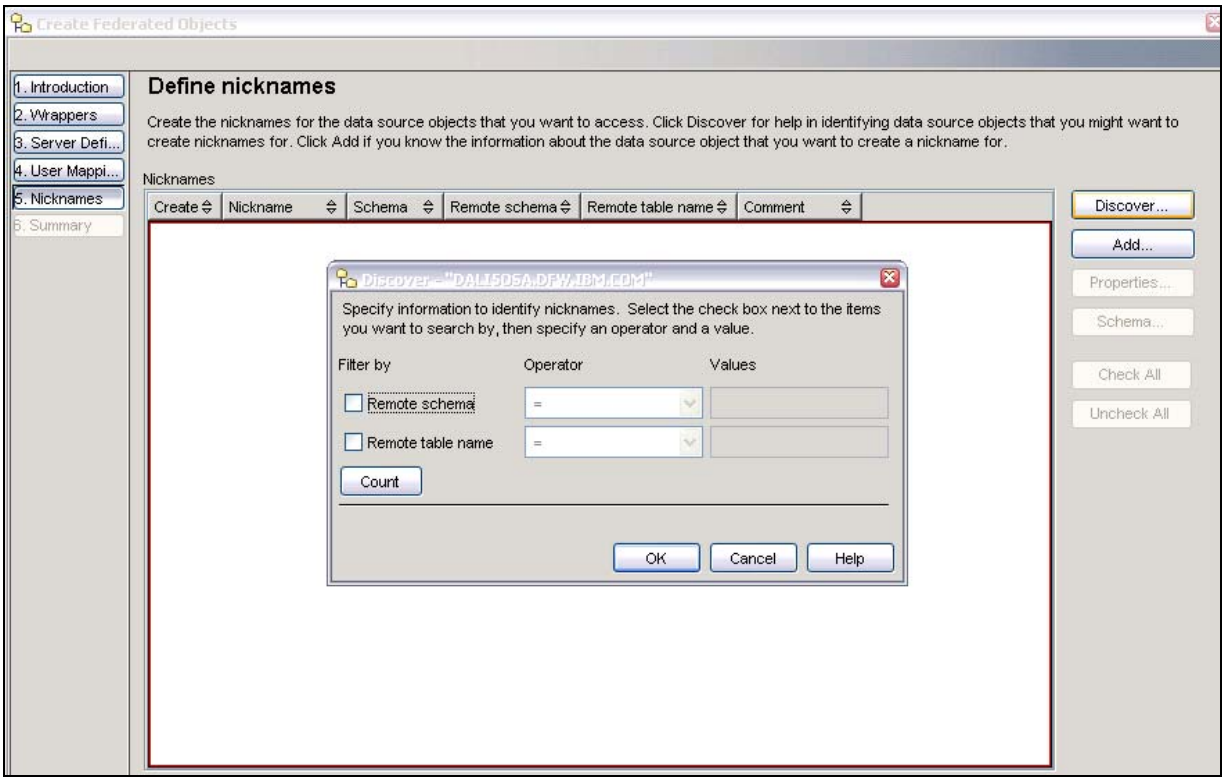

Figure 32. DB2 for i5/OS nickname-definitions window

Under Filter by, select **Remote schema**. Type a value of `FLGHT400` and click **Count** to determine how many DB2 for i5/OS objects reside in this schema.

After the DB2 object count is returned, click **OK** to generate a list of DB2 objects in the specified schema (see Figure 33).

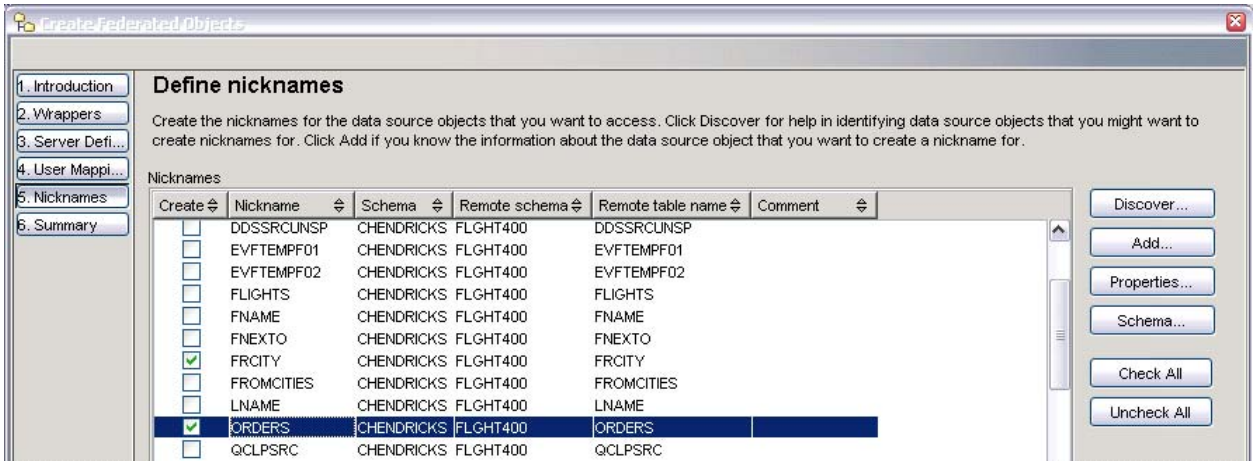

Figure 33. DB2 for i5/OS nicknames listing

Select the needed DB2 for i5/OS tables and then click **Next** to advance to the Nickname Review panel

(see Figure 34).

Figure 34. DB2 for i5/OS nickname reviews

After reviewing the settings, click **Finish** to create the following nicknames:

```
CREATE NICKNAME db2fed.orders
                    FOR “DALI5OSA.DFW.IBM.COM”.flght400.orders
CREATE NICKNAME db2fed.frcity
                    FOR “DALI5OSA.DFW.IBM.COM”. flght400.frcity
CREATE NICKNAME db2fed.tocity
                    FOR “DALI5OSA.DFW.IBM.COM”. flght400.tocity
```

Now an i5/OS application or SQL interface must issue the following SQL statement to join the DB2 for i5/OS table with the SQL Server table. This join example shows how you can reference tables from different databases on a single SQL statement when using WebSphere Federation Server:

```
CONNECT TO FEDDB USER myuser USING passwd
SELECT * FROM db2fed.agents, db2fed.orders
   WHERE db2fed.agents.agent_no = db2fed.orders.agent_no
```

Pass-through facility

The WebSphere Federation Server pass-through facility allows you to pass any SQL statement directly to the target database.

1. Write the SQL statement that uses a pass-through into the SQL syntax that is supported by the target database. Here is an example of how to use pass-through to perform change operations on an Oracle database with WebSphere Federation Server:

```
CONNECT TO SAMPLE USER udbuser USING password
SET PASSTHRU orasvr
UPDATE orasvr.scott.emp SET DEPT='PFV' WHERE DEPT='VQM'
COMMIT
```

   The pass-through facility does require that you define the target database server. Also notice that you must use the Oracle table identifier (and not the table nickname) on the UPDATE statement.

Additional mappings

WebSphere Federation Server tries to map functions and data types automatically that DB2 does not directly support. For example, the SQL Server PUBS sample database has a data type of *empid* that you must map into a DB2 data type. If data types and functions such as this are in the target data sources where the mappings are not clear, you can use the CREATE TYPE MAPPING and CREATE FUNCTION MAPPING statements to define those mappings in the WebSphere Federation Server catalogs.

More than middleware

Not only does WebSphere Federation Server translate the SQL request into a language that the target database understands — it also optimizes the request by employing global optimization. When formulating how to translate the query, the DB2 optimizer considers relative processor and I/O speed for each server, as well as relative network bandwidth. The optimizer then combines these factors with the automatically gathered statistics for each database server and determines the most efficient path to the needed information. The DB2 optimizer also incorporates a sophisticated query-rewrite phase to automatically transform poorly written queries into better, logically equivalent forms that are less costly to run.

The function-compensation support in WebSphere Federation Server is also powerful, and comes into play when you run DB2 SQL features, regardless of whether the target database supports them. Here is an example that shows the power of this feature. Earlier DB2 for i5/OS releases had a restriction on the ORDER BY clause, whereby any column specified must also be specified on the SELECT list. If the following SQL statement runs directly on an i5/OS system, a syntax error is signaled. However, if the SQL statement runs against an i5/OS system through WebSphere Federation Server, it runs successfully.

```
SELECT name, position FROM staff ORDER BY dept
```

WebSphere Federation Server makes up for limitations in the target database by altering the SQL request to compensate for the missing functionality. Sometimes this might result in the data being copied from the target database into DB2, where the compensated function can be performed on the copy of the data. Again, the WebSphere Federation Server engine looks at the costs involved to determine the most efficient way of compensating for the missing feature.

Summary

If you need to seamlessly access disparate data sources, there is an answer. WebSphere Federation Server solves the heterogeneous data-access problem for i5/OS applications by providing a seamless SQL interface to access data that is not stored in a DB2 table. For more information about WebSphere Federation Server, visit: www-304.ibm.com/jct03002c/software/data/integration/federation_server/.

Online documentation for WebSphere Federation Server can be found at:
ibm.com/support/docview.wss?rs=3170&uid=swg27007246

Appendix A

Here is an example of creating an SQL-server data source:

1. In the Windows Start menu, click **Control Panel**.
2. Click **Administrative Tools** and select **Data Sources (ODBC)**.

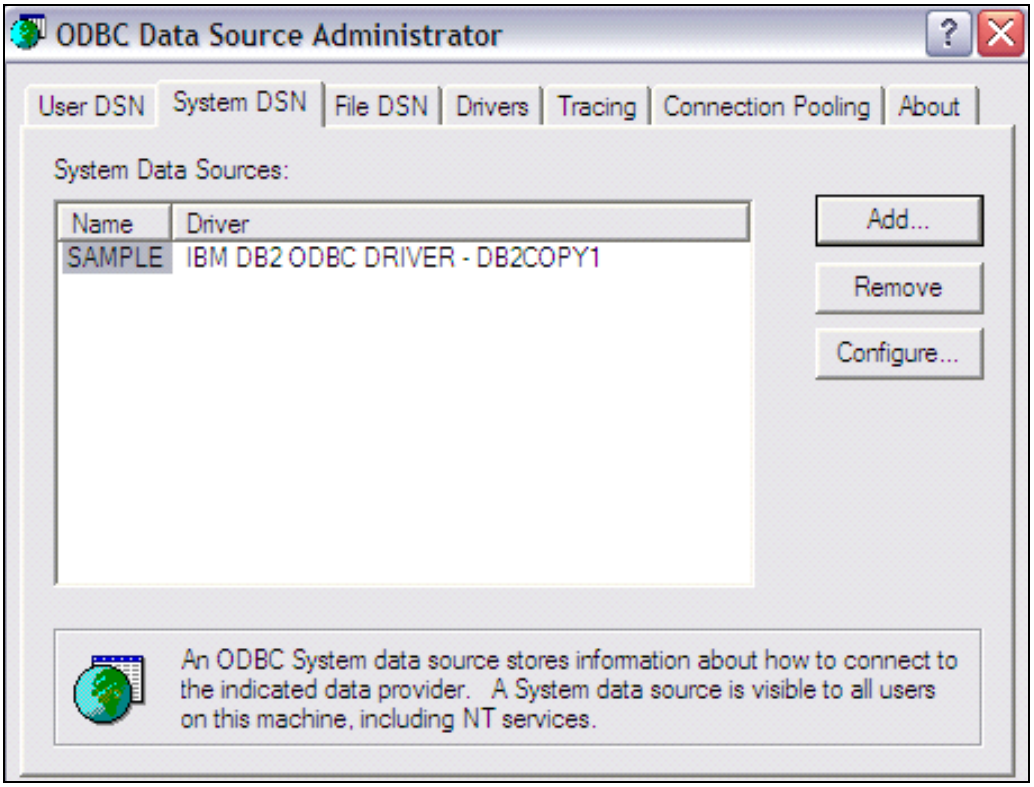

Figure 35. ODBC Data Source Administrator

Select the system DSN and click **Add**.

Select the **SQL Server driver** from the list of ODBC drivers on the Create New Data Source window. The Microsoft SQL Server DNS Configuration window shown in Figure 36 opens, which allows you to configure the DSN for the Microsoft SQL Server database.

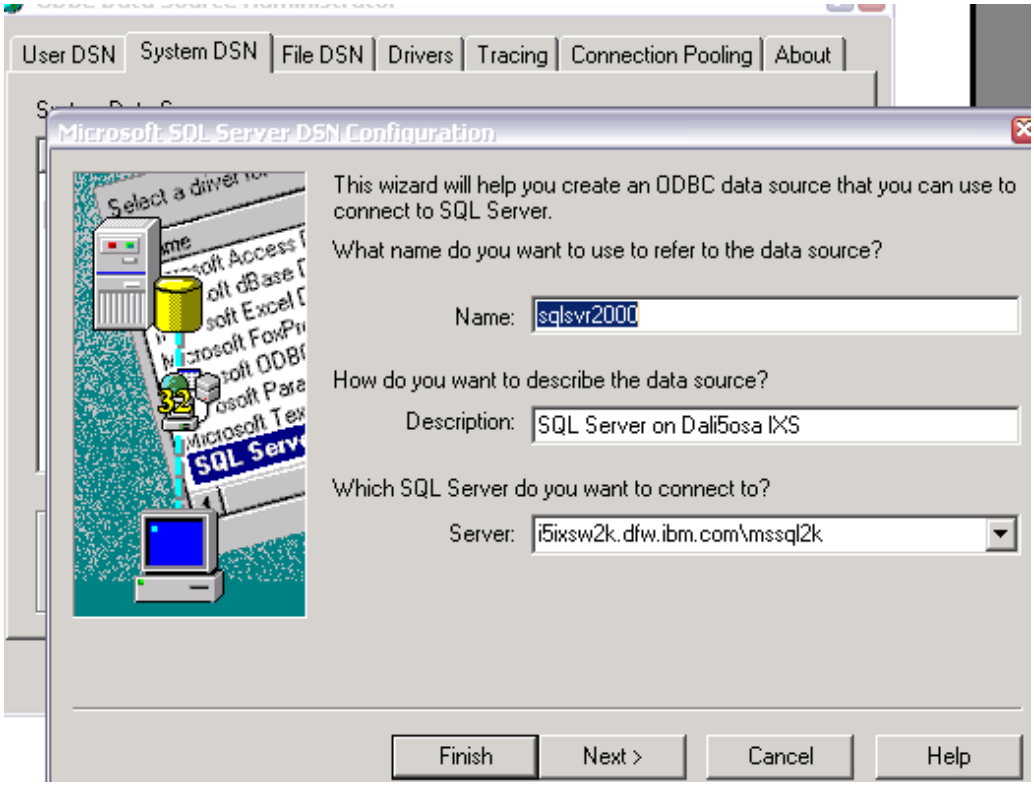

Figure 36. SQL Server DSN Configuration window

Specify the system DSN name as `sqlsvr2000` and the server name of the database as `i5ixs2wk.dfw.ibm.com\mssql2k`. Then, click **Next** (see Figure 37).

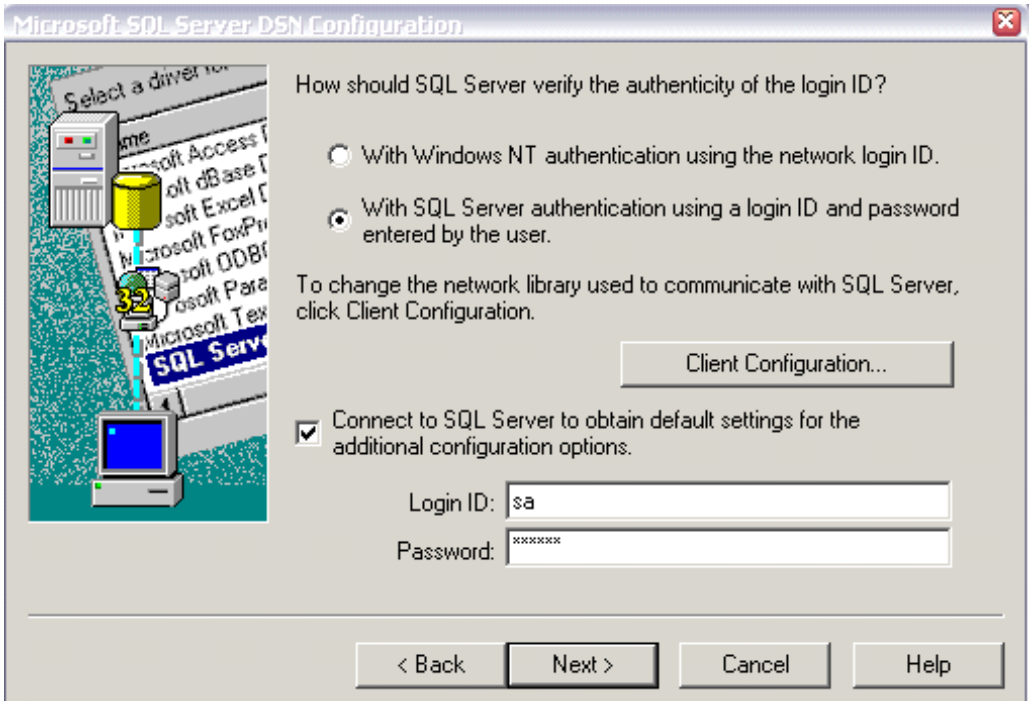

Figure 37. SQL Server connection configuration

Enter an SQL Server ID and password to access additional configuration options. Click **Next** to log in and access the configuration options (see Figure 38).

Enter the default database that this system DSN should use. For this example, the default database is db2wii.

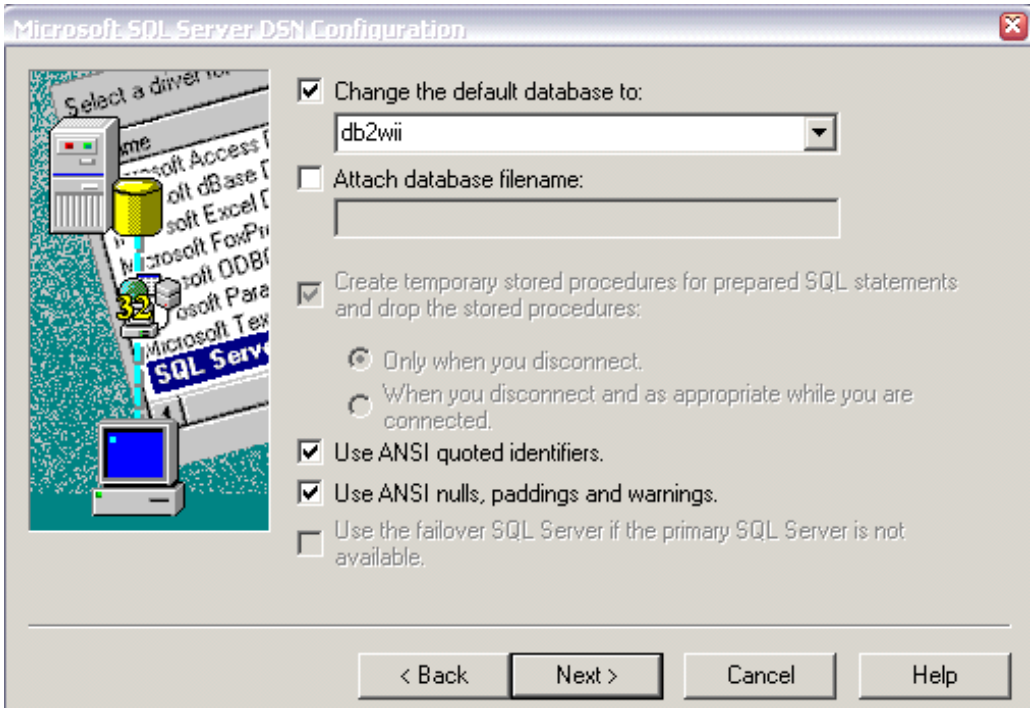

Figure 38. Connection configuration options

After specifying the default database, click **Next**.

Click **Finish** to complete the creation of the ODBC system DSN for the SQL Server database.

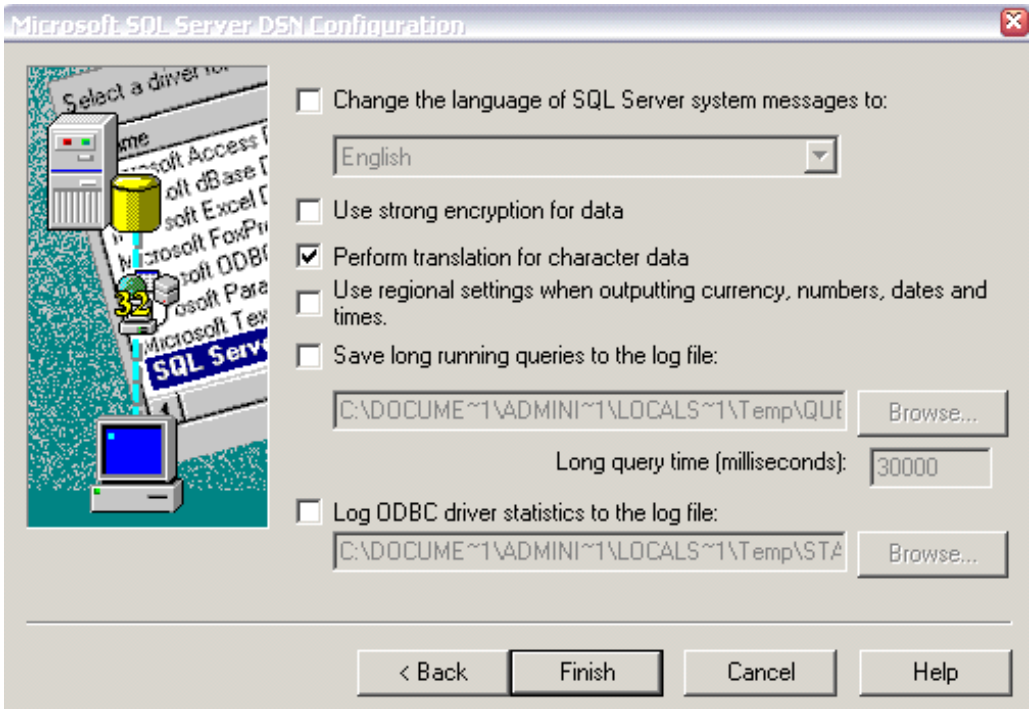

Figure 39. Connection configuration options (continued)

Resources

These Web sites provide useful references to supplement the information contained in this document:

- IBM System i Information Center
  http://publib.boulder.ibm.com/iseries
- i5/OS on IBM PartnerWorld®
  ibm.com/partnerworld/i5os
- IBM Publications Center
  www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi?CTY=US
- IBM Redbooks®
  ibm.com/redbooks
- Online documentation for WebSphere Federation Server
  ibm.com/support/docview.wss?rs=3170&uid=swg27007246
- WebSphere Federation Server
  www-304.ibm.com/jct03002c/software/data/integration/federation_server/
- EDA and iWay Software from Information Builders
  www.informationbuilders.com
- RPG2SQL Integrator from RJS Software
  www.rjssoftware.com
- Attunity Connect
  www.attunity.com
- DataGlider software from DataGlider
  www.dataglider.com

About the authors

Kent Milligan is a DB2 technology specialist in IBM ISV Business Strategy and Solutions Enablement for the System i platform. After graduating from the University of Iowa, Kent spent the first eight years of his IBM career as a member of the DB2 development group in Rochester, Minnesota. He speaks and writes regularly on various DB2 for i5/OS relational database topics.

Colin Hendricks is a certified IBM I/T specialist in IBM Advance Technical Support, providing technical sales support on i5/OS database solutions and information-management products that access the System i platform. His career spans more than 18 years with IBM, and he has 29 years, overall, of experience in the I/T field. Throughout his I/T career he has worked exclusively on the IBM midrange platform, performing various technical-support roles, application design, application development, I/T-operations management and project management.

Trademarks and special notices

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

AIX, BladeCenter, DataJoiner, DataPropagator, developerWorks, DB2, i5/OS, IBM, the IBM logo, PartnerWorld, Redbooks, System i, System x, and WebSphere are trademarks of International Business Machines 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.

Intel, Intel Inside (logos), MMX and Pentium are trademarks of Intel Corporation in the United States, other countries or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries or both.

Other company, product or service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.