IBM eserver

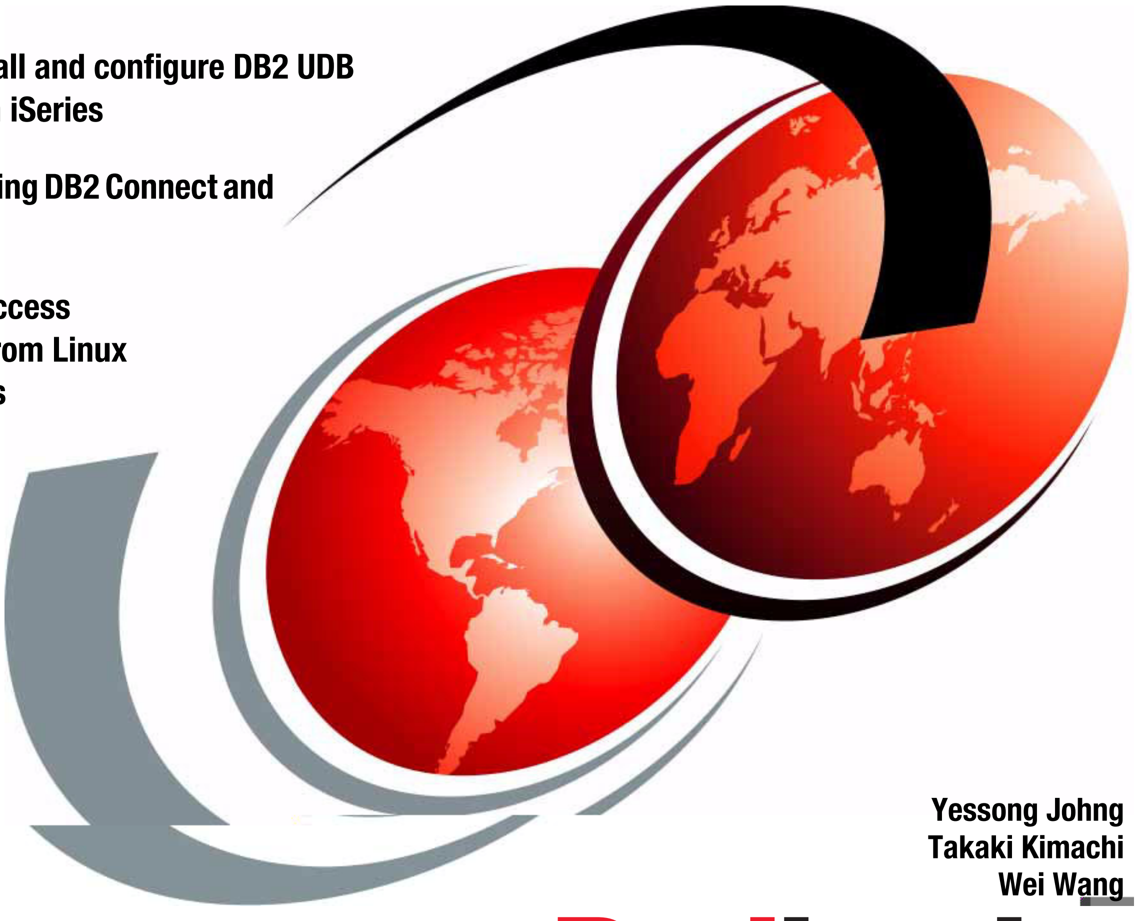
IBM

DB2 UDB for Linux on iSeries Implementation Guide

How to install and configure DB2 UDB for Linux on iSeries

Understanding DB2 Connect and its features

Database access scenarios from Linux applications



Yessong Johng
Takaki Kimachi
Wei Wang

Redbooks

ibm.com/redbooks

International Technical Support Organization

DB2 UDB for Linux on iSeries Implementation Guide

April 2004

SG24-6081-00

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (April 2004)

This edition applies to DB2 UDB for Linux on iSeries V8.1.2 and OS/400 V5R2.

Contents

Notices

Preface

This IBM® Redbook discusses DB2® UDB for Linux on iSeries™. We begin with a brief introduction and provide the installation and configuration steps for DB2 UDB for Linux on iSeries. We also present connectivity scenarios between OS/400® resources and Linux resources, mainly from the perspective of Linux applications accessing DB2 for OS/400 using DB2 Connect™.

This redbook will help administrators and developers of DB2 UDB for Linux on iSeries. Some of the information provided in this redbook should be helpful for IT managers or decision makers to consider the advantages and disadvantages between the DB2 products offerings for either OS/400 or Linux on iSeries.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Rochester Center.

Yessong Johng is an IBM Certified IT Specialist at the IBM International Technical Support Organization, Rochester Center. He started his IT career at IBM as a S/38 Systems Engineer in 1982 and has been involved with S/38, AS/400®, and now iSeries for 20 years. He writes extensively and develops and teaches IBM classes worldwide in the areas of e-business on iSeries. His major responsibilities are Linux, IT Optimization, and WebSphere® on iSeries. Additionally, new responsibilities include AIX® support on iSeries.

Takaki Kimachi is an IBM Certified IT Specialist at the iSeries Techline, since 2000 in the areas of Linux on iSeries, iSeries LPAR and hardware configuration, covering Japan and Asia Pacific. He joined IBM Japan in 1984 as a Software Engineer providing technical support for customers and business partners. His career has spanned 18 years in IBM midrange server S/36, AS/400, and iSeries.

Wei Wang is the Technical Director responsible for industry technical solutions at Tsinghua Unisplendour Co., Ltd., the main IBM Business Partner and iSeries/400 distributor in China. He is an IBM Certified Professional in IBM DB2, AIX, OS/400 and Tivoli® Technology since 1997. His email address is wangwei@gp.thunis.com.

Thanks to the following people for their contributions to this project:

Doug Mack
IBM

Connie Tsui
Rav Ahuja
Wayne Young
IBM Toronto

Jarek Miszczyk
IBM Rochester

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. JLU Building 107-2
  3605 Highway 52N
  Rochester, Minnesota 55901-7829

1

Introduction to DB2 for Linux on iSeries

This chapter provides a brief introduction to DB2 Universal Database™ (UDB) for Linux on iSeries.

1.1 What is DB2 UDB for Linux on iSeries ?

DB2 Universal Database (UDB) represents IBM's family of Relational Database Management Systems (RDBMS) that run on over twenty platforms. With V8.1, DB2 UDB for Linux is available for running in a Linux partition on an IBM @server™ iSeries system.

The following section answers some commonly asked questions regarding DB2 for Linux on iSeries.

1. How does DB2 UDB for Linux on iSeries compare to DB2 UDB versions that run in UNIX®, Windows®, and Linux on Intel® platforms?

   DB2 UDB for Linux on iSeries leverages the same code base as DB2 UDB for Linux/UNIX/Windows (LUW). It is the same product, now supported on Linux on iSeries. This support is available with V8.1.2.

2. Where would DB2 UDB for Linux be most applicable in an existing iSeries installation?

   Linux is one of the fastest growing operating systems in the industry today. While initially it has been used mostly for Web serving, e-mail/file/print serving and firewalls, Linux is quickly becoming a key enabler for e-business applications that are demanding more robust local database capabilities. Customers are looking for the flexibility that the open source Linux OS brings, while not sacrificing the reliability and scalability benefits of an Enterprise Server.

   Applications running in the Linux partition can certainly leverage DB2 UDB for iSeries through a variety of interfaces (DB2 Connect, JDBC, ODBC), and this is an excellent choice to leverage the low administration overhead, autonomic computing, and security benefits provided by DB2's tight integration with OS/400. DB2 for Linux is an excellent choice to support Linux (or other client/server) applications when there is a need to have local data storage within the Linux application environment or to leverage specific features of DB2 UDB V8 such as Federated Database or Microsoft® .NET Framework support.

3. What are the Linux distributions supported with DB2 for Linux on iSeries?

   SLES 8 (SUSE Linux Enterprise Server 8) or TLES 8 (TurboLinux Enterprise Server 8) or RHEL 3 (Red Hat Enterprise Linux 3) offer support for DB2 for Linux on iSeries.

   Note: Support for new distributions can be frequently updated. Review the DB2 for Linux validation Web site for the latest status:

   `http://ibm.com/db2/linux`

1.2 DB2 UDB for Linux on iSeries products overview

Several DB2 products are available for Linux on iSeries. These include:

- DB2 Universal Database, Version 8.1 - Enterprise Server Edition
- DB2 Universal Database, Version 8.1 - Workgroup Server Edition
- DB2 Universal Database, Version 8.1 - Workgroup Server Unlimited Edition
- DB2 Connect, Version 8.1 - Enterprise Edition
- DB2 Connect, Version 8.1 - Application Server Edition
- DB2 Connect, Version 8.1 - Unlimited Edition

Enterprise Server Edition

Enterprise Server Edition (ESE) meets the database server needs of midsize to large businesses. ESE is the ideal foundation for building data warehouses, transaction processing, or Web-based solutions as well as a backend for packaged solutions like ERP, CRM, and SCM. Additionally, ESE offers connectivity and integration for other enterprise DB2 and Informix® data sources.

The Database Partitioning Feature (DPF) is a licensing option that allows ESE customers to partition a database within a single system or across a cluster of systems. The DPF capability provides the customer with multiple benefits including scalability to support very large databases or complex workloads, and increased parallelism for administration tasks.

Workgroup Server Edition

Workgroup Server Edition (WSE) is the database server designed for deployment in a departmental or small business environment that involves a small number of internal users. WSE has a user-based licensing model designed to provide an attractive price point for smaller installations while still providing a full-function database server. WSE can be deployed on systems with up to four CPUs. As of this writing, a five user license of WSE is included with the iSeries Enterprise Edition option on certain iSeries models.

Workgroup Server Unlimited Edition

Workgroup Server Unlimited Edition (WSUE) offers a simplified per-processor licensing model for deployment in a departmental or small business environment that has Internet users or number of users that makes per processor licensing more attractive than the WSE licensing model. The WSUE is also for use on systems with up to four CPUs.

DB2 Connect

DB2 Connect comes in two editions for DB2 for Linux:

- **DB2 Connect Enterprise Edition**: DB2 Connect Enterprise Edition addresses the needs of organizations that require robust connectivity from a variety of desktop systems to zSeries® and iSeries database servers. DB2 Client software is deployed on desktop systems and provides API drivers that connect client-server applications running on these desktop systems to a DB2 Connect server (gateway) that accesses host data. The licensing model for this product is user based.
- **DB2 Connect Application Server Edition**: DB2 Connect Application Server Edition product is identical to the DB2 Connect Enterprise Edition in its technology. However, its licensing terms and conditions are meant to address specific needs of multi-tier client-server applications as well as applications that utilize Web technologies. DB2 Connect Application Server Edition license charges are based on the number of processors available to the application servers where the application is running.

1.3 Positioning of DB2 UDB for Linux on iSeries

Positioning of DB2 for Linux on iSeries can be discussed from two different angles. One angle is product comparison between DB2 for Linux and DB2 UDB for iSeries, looking at feature/function, administrative, and packaging differences. The other angle is from the perspective of the environment/platform differences that impact the choice of where to run DB2.

This section lists what's common across the DB2 family, then the new features available for DB2 for Linux on iSeries, and the benefits of running DB2 on the iSeries platform.

1.3.1 Common features across the DB2 family

The following features are common across the DB2 family:

- Common SQL support

> **Note:** At any point in time there may not be 100% compatibility of SQL across the DB2 family. For a complete list of COMMON SQL, refer to:
>
> `http://www-1.ibm.com/servers/enable/site/db2/db2common.html`

- Exploitation of SMP (Symmetrical Multiprocessing) and cluster-based parallelism

- Advanced analytical application support through OnLine Analytical Processing extensions and IBM patented indexing technologies
- Ability to manage multiple data types including Binary Large Objects
- Built on open industry standards: SQL, DRDA®, CLI, ODBC, JDBC and more
- Leader in query optimization technology and performance
- Stored procedures, UDFs, data encryption and globalization
- Fully Web-enabled and certified for leading enterprise applications
- High Availability features and Failover support
- J2EE certified and Java™ application interfaces for JDBC and SQLJ
- Support for open source interfaces such as PHP, Python and Perl
- DB2 XML Extender for storage and retrieval of XML documents
- A graphical, integrated toolset for managing local and remote databases
- Productivity tools for visual development and migration
- Exploitation of the latest Linux kernel capabilities

1.3.2 DB2 UDB for Linux on iSeries V8 features

Features of DB2 UDB for Linux on iSeries include:

- Innovative technologies to reduce DBA intervention, such as Health Center and Memory Visualizer
- Multidimensional Clustering and Materialized Query Tables for improved performance of complex queries
- Federated database support
- Plug ins for Eclipse, Rational®, and Microsoft .NET Framework support

1.3.3 iSeries benefits

iSeries has many benefits as a database server as listed below:

- **Server consolidation**: The iSeries can consolidate many Linux servers. The mid-to-high end iSeries models can support up to 31 Linux partitions, and up to 9 partitions on a single CPU iSeries system can be defined on lower end servers. The next generation of iSeries can support even more. In this environment, different distributions of Linux is supported, such as Red Hat, SUSE, TurboLinux, and UnitedLinux. So different versions of DB2 on different platforms can be supported simultaneously.
- **Dynamic logical partitioning**: Resources can be shared between OS/400 and Linux partitions, providing more flexibility to optimize system resources in

dynamic workload environments. With the dynamic logical partitioning feature of the iSeries servers, you can move processor and memory resources across the partitions. Partitioning also offers virtual Ethernet which provides high speed communications between the environments. For DB2 users on this partition, you can easily allocate a reasonable amount of processing power to avoid over configuration.

- **Simplified management**: Additional benefits include management of complex heterogeneous environments through Virtual I/O support. This support allows sharing of tape, disk, CD/DVD devices while being centrally managed by OS/400. Visualized storage allows for disk storage to be dynamically allocated by OS/400 for use by Linux environments.
- **Leverage existing hardware**: With shared processor and virtual I/O support, there is a minimal hardware requirement for Linux support on iSeries. On some selected iSeries models, an additional CPU is even provided at no extra charge to run Linux.
- **On/Off Capacity on Demand**: The iSeries offers On/Off Capacity on Demand with several models that allow customers the flexibility and immediate ability to manage to changing (expected or unexpected) workloads. Turn on additional CPUs temporarily or permanently to manage end of the month processing or unexpected spikes in e-Business workloads. It is extremely useful for customers running their business application based on DB2 for Linux on iSeries.
- **Reliability**: The iSeries offers customers the reliability of a mainframe with the simplicity of a midrange server. The Linux for iSeries are usually the most robust and advanced versions of different Linux Vendors, along with the leading Linux technology from them, iSeries provides an integrated hardware/OS platform for DB2 application.

2

Installation and configuration

This chapter shows you how to install the DB2 UDB V8.1 for Linux iSeries. We first discuss the basic knowledge required to handle the DB2 on Linux platform, and then show the step-by-step installation process of DB2. After installing DB2, we create a DB2 sample database and test it, simply by using a DB2 Command Line Processor (CLP).

2.1 Tutorial for a basic understanding of DB2 for Linux

In this section, we describe the minimum knowledge that is required to install and configure DB2 UDB for Linux. Since DB2 UDB on Linux or UNIX or Windows platforms has a different system structure than DB2 UDB for iSeries, those who are not used to the structure, terms, should read this section first. If you are familiar with DB2 on these platforms, that is DB2 UDB for Linux/UNIX/Windows (LUW), you do not need to read this section.

For further information, refer to these DB2 manuals:

- *IBM DB2 Universal Database Administration Guide: Planning*, SC09-4822
- *IBM DB2 Universal Database Administration Guide: Implementation*, SC09-4820

2.1.1 Database objects

For DB2 UDB for LUW, there is a nested structure that includes the data stored in the database. We will discuss that here.

System

A *system* is usually a physical machine on which DB2 UDB is installed. In a parallel database system, *system* means the set of machines which have all the database partitions. A *system* can have multiple *instances*.

Instances

An *instance* (sometimes called a *database manager*) is the top level concept of the structure. Databases have to be created in an instance. It controls what can be done to the data, and manages system resources assigned to it. Each instance is a complete environment. In a parallel database system, it contains all the database partitions defined for a the parallel database system. An instance has its own databases (which other instances cannot access), and all its database partitions share the same system directories. It also has separate security from other instances on the same machine (system). All the objects of an instance are created under a directory which has the same name as the instance name. The directory is the home directory of the instance owner and called “instance directory”.

For DB2 UDB for iSeries, there is no concept of *instance*.

Databases

A *relational database* presents data as a collection of tables. A table consists of a defined number of columns and any number of rows. Each database includes a set of system catalog tables that describe the logical and physical structure of the data, a configuration file containing the parameter values allocated for the database, and a recovery log with ongoing transactions and archival transactions.

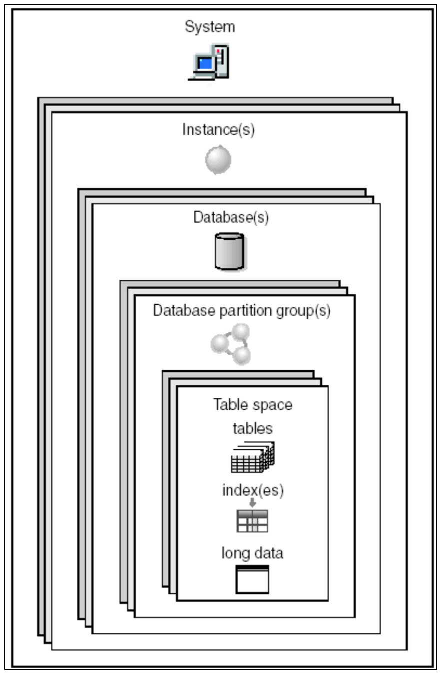


Figure 2-1 The structure of database objects

Database partition groups

A *database partition group* is a set of one or more database partitions. When you want to create tables for the database, you first create the database partition group where the table spaces will be stored, then you create the table space where the tables will be stored. In a multi-partitioned database, a database can have two or more database partitions in a database, which are in different machines. Multi partitioned databases are out of the scope of this book.

In earlier versions of DB2, database partition groups were known as *node group*s.

For DB2 UDB for iSeries, you can have database partition groups by installing the DB2 Multisystem feature, which is an option of OS/400.

Table spaces

A database is organized into parts called *table spaces*. A table space is a place to store tables. When creating a table, you can decide to have certain objects such as indexes and large object (LOB) data kept separately in another table spaces. A table space can also be spread over one or more physical storage devices.

Table spaces reside in database partition groups. *Containers* are assigned to table spaces. A *container* is an allocation of physical storage (such as a file or a device).

A table space can be either system managed space (SMS), or database managed space (DMS). For an SMS table space, each container is a directory in the file space of the operating system, and the operating system's file manager controls the storage space. For a DMS table space, each container is either a fixed size pre-allocated file, or a physical device such as a disk, and the database manager controls the storage space.

For DB2 UDB for iSeries, there is no concept of *table spaces*.

Tables, views, and indexes

A relational database presents data as a collection of *table*s. A table consists of data logically arranged in columns and rows. All database and table data is assigned to table spaces.

A *view* provides an efficient way of representing data without needing to maintain it. A *view* is not an actual table and requires no permanent storage. A "virtual table" is created and used.

An *index* is a set of keys, each pointing to rows in a *table*. The key values provide a pointer to the rows in the table.

Containers

A *container* is a physical storage device. It can be identified by a directory name, a device name, or a file name. A container is assigned to a table space. A single table space can span many containers, but each container can belong to only one table space. Note that containers are not shown in Figure 2-1 because the diagram is the logical view of the database objects.

Schemas

A schema is an identifier, such as a user ID, that helps group tables and other database objects. A schema can be owned by an individual, and the owner can control access to the data and the objects within it. A schema is also an object in the database. It may be created automatically when the first object in a schema is created. Such an object can be anything that can be qualified by a schema name, such as a table, index, view, package, distinct type, function, or trigger. A schema name is used as the first part of a two-part object name. When an object is created, you can assign it to a specific schema. If you do not specify a schema, it is assigned to the default schema, which is usually the user ID of the person who created the object. The second part of the name is the name of the object. For example, a user named Smith might have a table named SMITH.PAYROLL. SMITH is the schema in this case.

For DB2 UDB for iSeries, *schema* is implemented as the *library*.

In this book, we use only one Linux *system* (the Linux LPAR partition of an iSeries) and only one instance named **`db2inst1`**.

2.1.2 Users and authorities

DB2 has its own security and even the root user is not ultimate in DB2. If you have any problem when using DB2, the authority should also be checked.

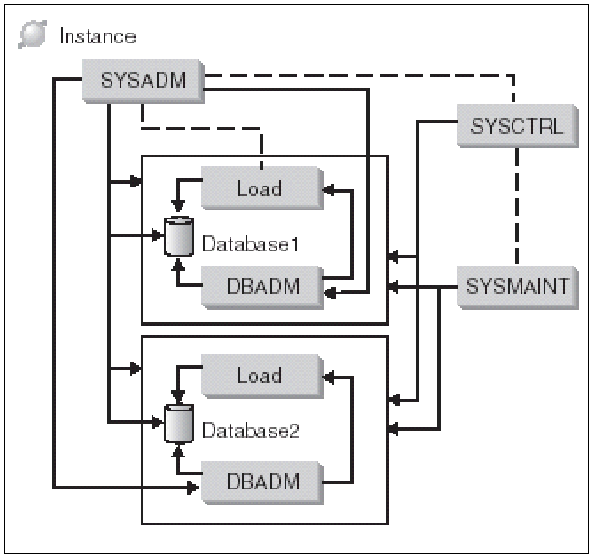


Figure 2-2 Hierarchy of authorities

Figure 2-2 shows the DB2 hierarchy of authorities. Some authorities are managed in the instance scope, and some are managed in database scope.

A user or group can have one or more of the following levels of authorization:

- Administrative authority (SYSADM or DBADM) gives full privileges for a set of objects.
- System authority (SYSCTRL or SYSMAINT) gives full privileges for managing the system, but does not allow access to the data.

- LOAD authority (LOAD) gives LOAD utility or AutoLoader utility privileges to load external data into DB2 tables.
- Ownership privilege (also called CONTROL privilege in some cases) gives full privileges for a specific object.
- Individual privileges may be granted to allow a user to carry out specific functions on specific objects.
- Implicit privileges may be granted to a user who has the privilege to execute a package. While users can run the application, they do not necessarily require explicit privileges on the data objects used within the package.

Users with administrative authority (SYSADM or DBADM) or ownership privileges (CONTROL) can grant and revoke privileges to and from others, using the GRANT and REVOKE statements. A user or group can be authorized for any combination of individual privileges or authorities.

SYSADM, STSCTRL, or SYSMAINT privilege is assigned to the group specified by the *sysadm_group*, *sysctrl_group*, *sysmaint_group* of the DB2 configuration parameter. If a user is assigned as a member of these groups, the user automatically gets the related DB2 privileges. These privileges are managed in instance level. By default, when creating an instance, the *sysadm_group* configuration parameter is set as the group of the instance owner and *sysctrl_group*, *sysmaint_group* are set to blank. The easiest way to be a database administrator is to issue the command **`su - (instance owner)`**. To check these configuration parameters, issue the following command:

```
db2 get database manager configuration
```

Or,

```
db2 get dbm cfg
```

DBADM or other privileges are given to any users or groups by SYSADM or other privileged users and the information is stored in the database itself.

2.1.3 Topics for DB2 operation

This section provides some tips for DB2 operation.

The db2profile and db2cshrc scripts

The db2profile is a useful shell script to set up the DB2 environment for the user. A user who wants to use DB2 must have some environment variables set properly. The db2profile script sets all of them properly. The db2profile script is for Bourne shell or Korn shell. For C shell, you can use the db2cshrc script instead.

The db2profile script is under the **sqllib** directory of the instance directory. It mainly does the following:

- Set the environment variable DB2INSTANCE to the instance name so that the user can access the DB2 instance implicitly.
- Set the environment variable INSTHOME to the instance directory.
- Add the DB2 command directories to the environment variable PATH, which is the command search path of the user.
- Add the DB2 Java files (.jar and .zip files) to the CLASSPATH environment variable.

Note that the db2profile script is under the instance directory and it is used to setup the environment of the *instance*.

To run the db2profile script in our environment where the instance directory is **/home/db2inst1**, issue the following command. Note that the first characters of the line is "**.**" (dot space).

```
. /home/db2inst1/sqllib/db2profile
```

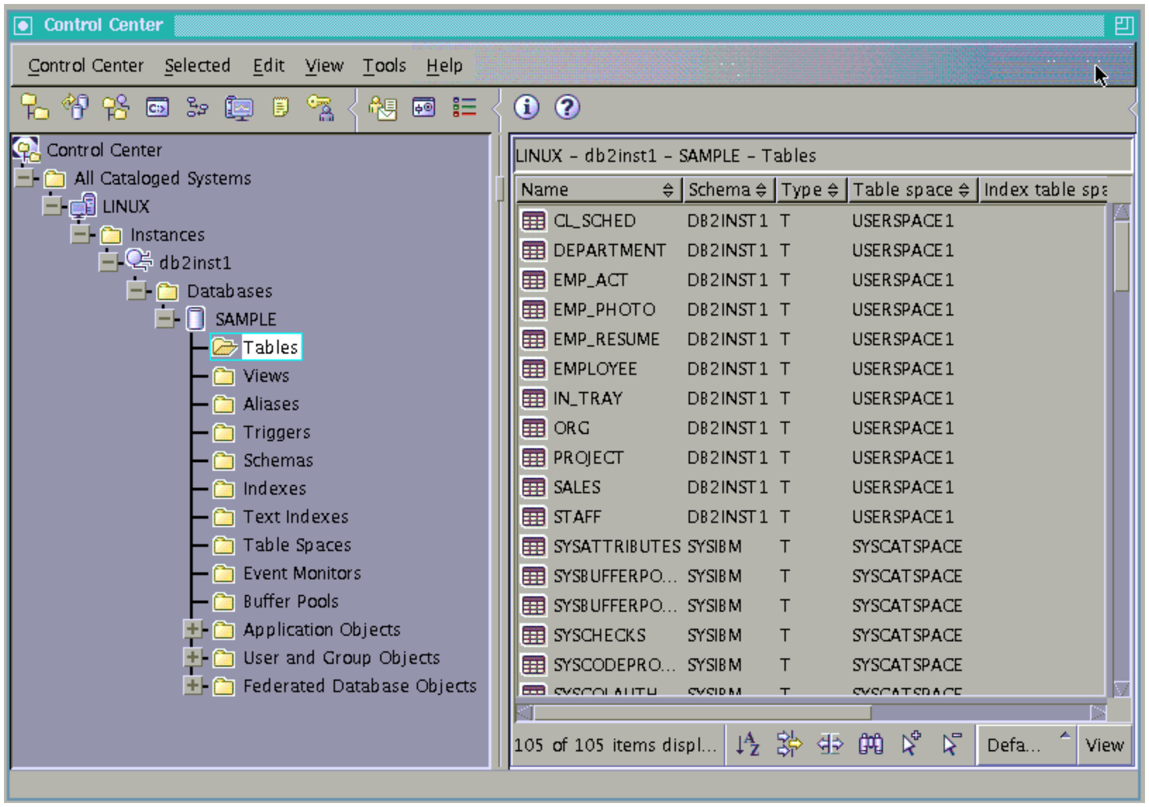
If needed, you can modify the db2profile script or you can include it in the **.bashrc** or other startup scripts. The db2prifile script is also included in the **.bashrc** of the instance owner db2inst1 who is created during the DB2 installation process.

DB2 Control Center (db2cc)

The DB2 Control Center (db2cc) is the GUI tool by which most of the DB2 administration tasks can be done. On the Linux platform, it needs the Java environment and X window. You can use db2cc on vnc or any other X window for the Linux on iSeries.

To launch db2cc, first check the DB2 authority of the current user, then check the DB2 environment of the current user (run db2profile if needed), and then type **db2cc** from a terminal emulator of a X window to have a window illustrated as in Figure 2-3.

When launching db2cc from a vnc environment, make sure that the user of the vncserver task has the proper authority.



LINUX - db2inst1 - SAMPLE - Tables

Name	Schema	Type	Table space	Index table spa
CL_SCHED	DB2INST1	T	USERSPACE1	
DEPARTMENT	DB2INST1	T	USERSPACE1	
EMP_ACT	DB2INST1	T	USERSPACE1	
EMP_PHOTO	DB2INST1	T	USERSPACE1	
EMP_RESUME	DB2INST1	T	USERSPACE1	
EMPLOYEE	DB2INST1	T	USERSPACE1	
IN_TRAY	DB2INST1	T	USERSPACE1	
ORG	DB2INST1	T	USERSPACE1	
PROJECT	DB2INST1	T	USERSPACE1	
SALES	DB2INST1	T	USERSPACE1	
STAFF	DB2INST1	T	USERSPACE1	
SYSATTRIBUTES	SYSIBM	T	SYSCATSPACE	
SYSBUFFERPO...	SYSIBM	T	SYSCATSPACE	
SYSBUFFERPO...	SYSIBM	T	SYSCATSPACE	
SYSCHECKS	SYSIBM	T	SYSCATSPACE	
SYSCODEPRO...	SYSIBM	T	SYSCATSPACE	

Figure 2-3 DB2 Control Center

2.2 Installation of DB2 UDB 8.1

In this section, we show you the step-by-step installation process to install DB2 UDB 8.1 Enterprise Server Edition on Linux into Linux on iSeries (SLES8). The following conditions are assumed.

- The SLES8 is installed in all default settings and TCP/IP is set up.
- A Windows PC is used as the vnc client to operate the X window.

2.2.1 Requirements

In this book, we use DB2 UDB Enterprise Server Edition on Linux 8.1. This topic lists the hardware, distribution, software, and communication requirements to install it into the Linux on iSeries platform.

Hardware requirements

DB2 Enterprise Server Edition is supported on iSeries. There are no special hardware requirements.

Distribution requirements

For the latest information on supported distribution and kernel levels, point your browser to `http://www.ibm.com/db2/linux/validate`. At the time of writing this redbook, these three distributions are supported on the iSeries platform.

- SUSE Enterprise Server (SLES) 8
- Turbolinux Enterprise Server (TLES) 8
- Red Hat Enterprise Linux (RHEL) 3

Software requirements

The IBM Developer Kit for Java 1.3.1 is required for DB2 servers, for using the DB2 Control Center, and for creating and running Java applications, including stored procedures and user-defined functions. During the installation process, if the correct level of JDK is not already installed, it will be installed.

Communication requirements

TCP/IP is required to access remote databases.

2.2.2 vnc overview

vnc is a remote control software works on various platforms. On Linux, it is often included in the distribution package. SLES8 for iSeries also includes vnc. After starting vncserver, an X server process of vnc runs on Linux. There are two ways of getting a vnc session from your workstation. One is the vncviewer and the other is the vnc client Java applet. The vncviewer can be downloaded from the vnc Web site.

`http://www.realvnc.com`

The vnc applet can be downloaded to a Web browser from the vnc http server which starts with vncserver using port 580*n*, where the *n* is the X screen number of vncserver. For example, if the vncserver is working on X screen number 1, the port is 5801.

In this chapter, we use the vncserver on Linux and the Java applet on Web browser.

1. Start the vnc server by typing the command vncserver as illustrated in Figure 2-4. For DB2 installation, the user must be root.

```
linux:~ # vncserver

You will require a password to access your desktops.

Password:
Verify:

New 'X' desktop is linux:1

Creating default startup script /root/.vnc/xstartup
Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/linux:1.log

linux:~ #
```

Figure 2-4 Start the vnc server

If this is the user's first time use of vncserver, you will get the password prompt and some control files are created under **`.vnc/`** directory of the user's home directory. The password is used when connecting from a client. Remember the X desktop number (1) that is needed to connect from a client.

2. Connect from the client using a Web browser. The URL is " **`http://`*`(Linux server address)`*`:5801/`**", in this case. The password page is displayed as illustrated in Figure 2-5. Type in the vnc password.

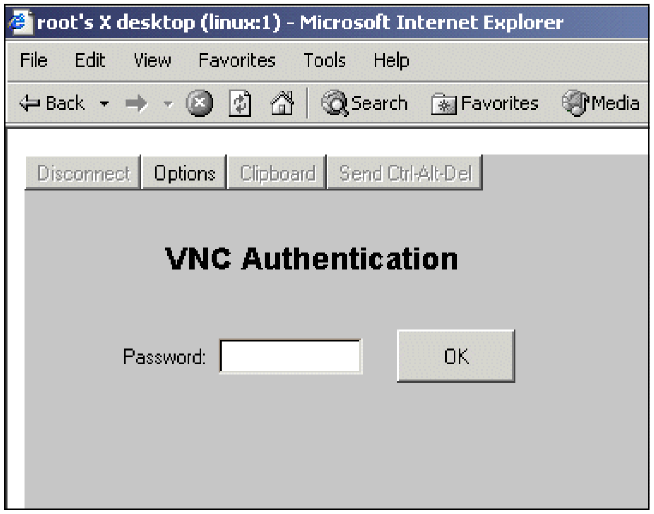
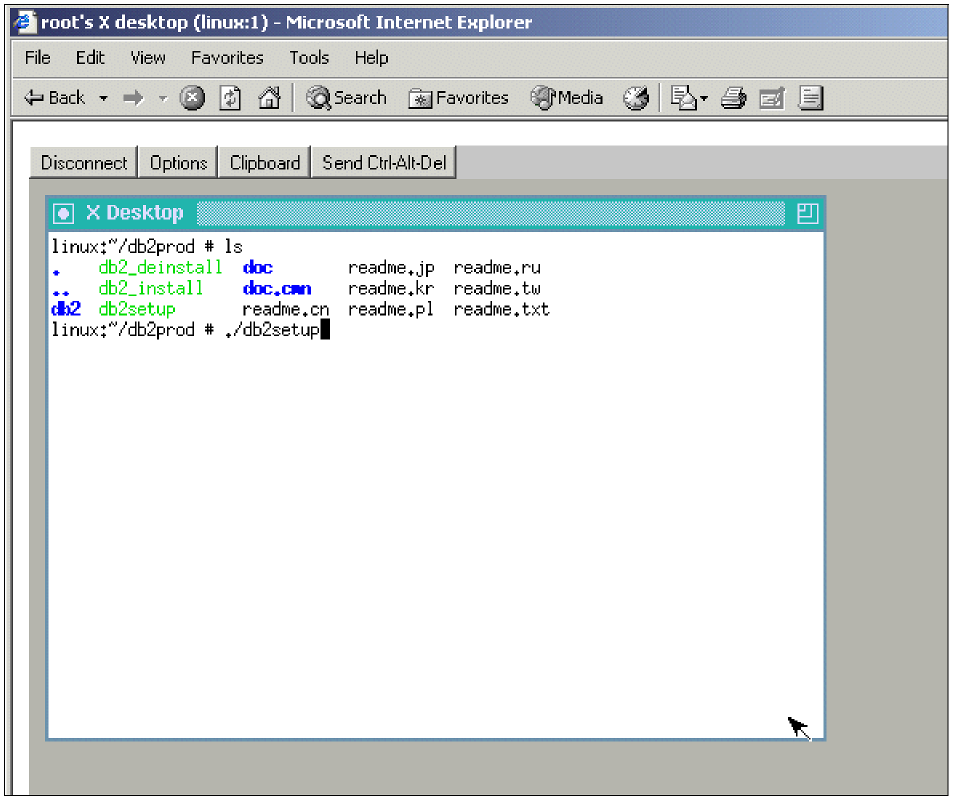

root's X desktop (linux:1) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Disconnect Options Clipboard Send Ctrl-Alt-Del
VNC Authentication
Password:
OK

Figure 2-5 vnc password authentication window

3. The Java applet is downloaded and the X desktop appears in the browser, as illustrated in Figure 2-6.


root's X desktop (linux:1) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Disconnect Options Clipboard Send Ctrl-Alt-Del
X Desktop

```
linux:~/db2prod # ls
.    db2_deinstall  doc       readme.jp  readme.ru
..   db2_install    doc.cmn   readme.kr  readme.tw
db2  db2setup       readme.cn readme.pl  readme.txt
linux:~/db2prod # ./db2setup
```

Figure 2-6 vnc session window

You can operate this window as the X desktop using the mouse and keyboard. The default window manager is **`twm`**.

2.2.3 Installing the DB2

This section provides a step-by-step guide for installing DB2.

1. To install DB2 UDB, mount the CDROM, change the directory to the CDROM, and type **`./db2setup`** from the xterm window of a vnc client. The installation must be done by the root user.

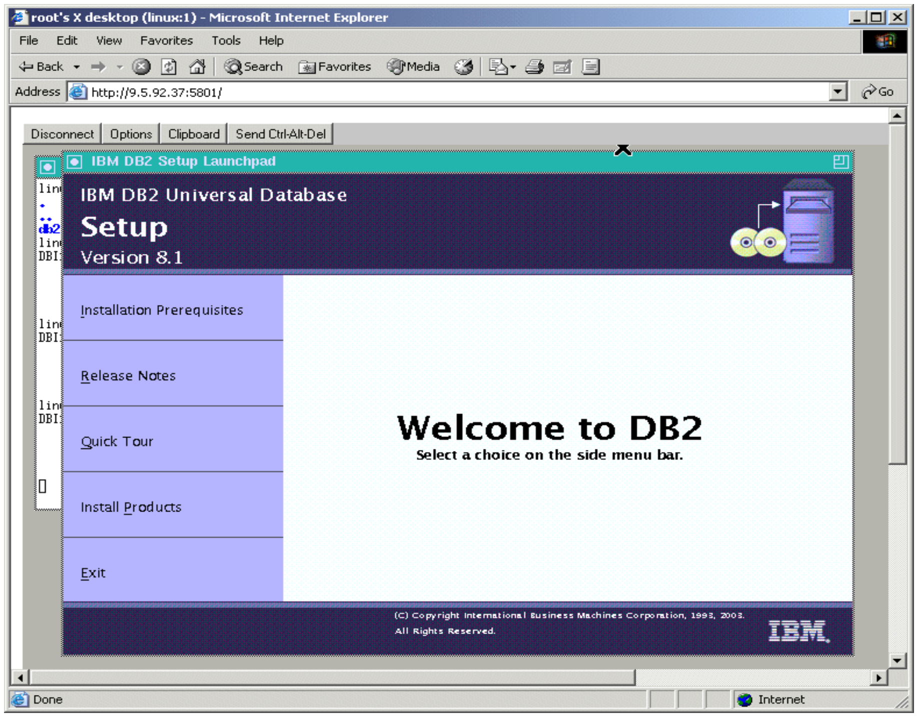
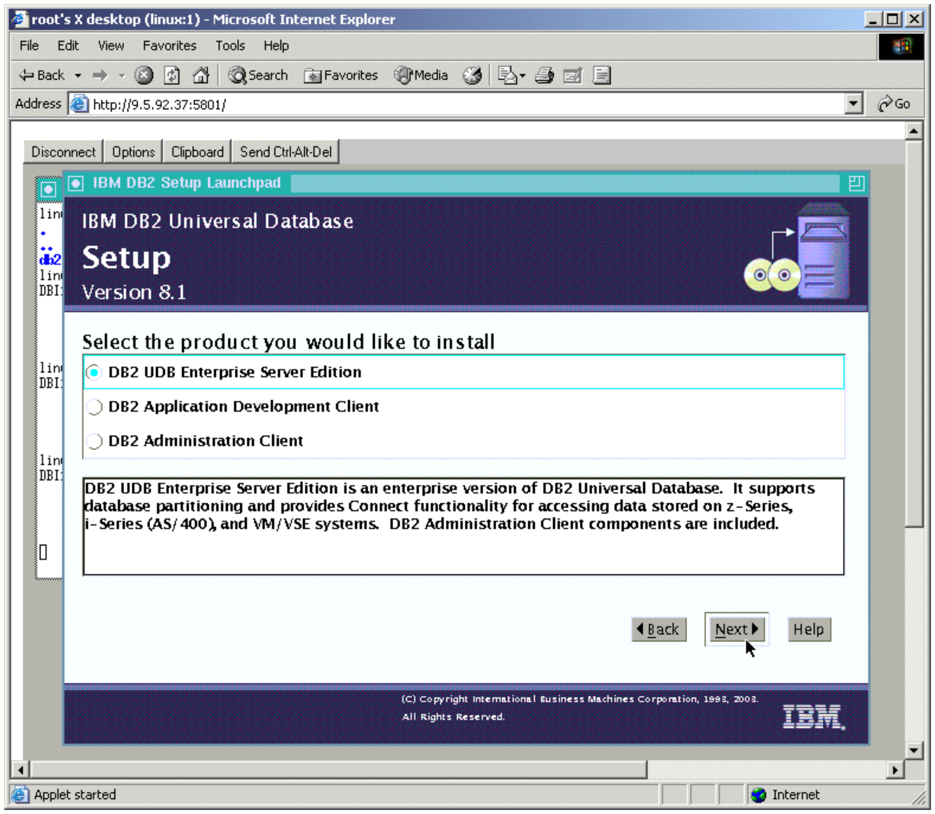

root's X desktop (linux:1) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://9.5.92.37:5801/
Go
Disconnect Options Clipboard Send Ctrl-Alt-Del
IBM DB2 Setup Launchpad
IBM DB2 Universal Database
Setup
Version 8.1
Installation Prerequisites
Release Notes
Quick Tour
Install Products
Exit
Welcome to DB2
Select a choice on the side menu bar.
(C) Copyright International Business Machines Corporation, 1993, 2003.
All Rights Reserved.
IBM
Done
Internet

Figure 2-7 DB2 setup welcome window

After a short time, the window will appear as illustrated in Figure 2-7. Click **Install products**.

Typically, the IBM Developer Kit for Java 1.3.1 is not installed during the standard installation of SLES8. It will be installed with DB2.

2. Select **DB2 UDB Enterprise Server Edition** as illustrated in Figure 2-8 and click **Next**. This step generally takes 1 to 2 minutes.


root's X desktop (linux:1) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://9.5.92.37:5801/
Go
Disconnect Options Clipboard Send Ctrl-Alt-Del
IBM DB2 Setup Launchpad
IBM DB2 Universal Database
Setup
Version 8.1
Back Next Help
(C) Copyright International Business Machines Corporation, 1993, 2003.
All Rights Reserved.
Applet started
Internet

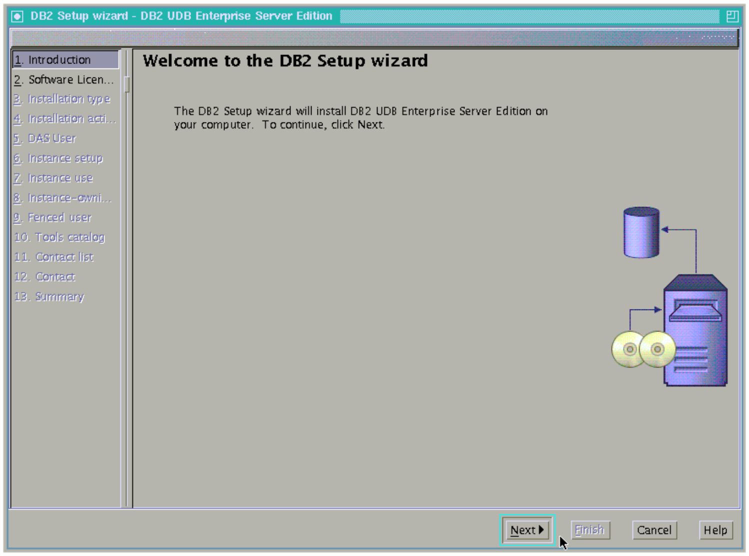
Select the product you would like to install

[✓ DB2 UDB Enterprise Server Edition]
DB2 Application Development Client
DB2 Administration Client

DB2 UDB Enterprise Server Edition is an enterprise version of DB2 Universal Database. It supports database partitioning and provides Connect functionality for accessing data stored on z-Series, i-Series (AS/400), and VM/VSE systems. DB2 Administration Client components are included.

Figure 2-8 Selection of the product to install

3. The setup wizard appears as illustrated in Figure 2-9. Click **Next**.

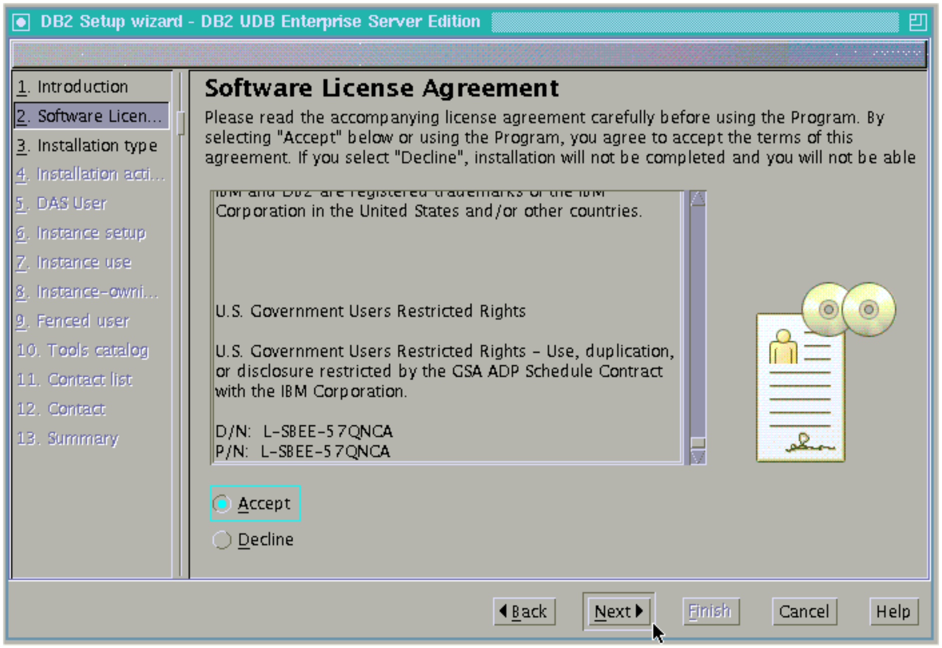


Welcome to the DB2 Setup wizard

The DB2 Setup wizard will install DB2 UDB Enterprise Server Edition on your computer. To continue, click Next.

Figure 2-9 The DB2 setup wizard

4. In the licence agreement window as illustrated in Figure 2-10, select **Accept** and click **Next**.



Software License Agreement

Please read the accompanying license agreement carefully before using the Program. By selecting "Accept" below or using the Program, you agree to accept the terms of this agreement. If you select "Decline", installation will not be completed and you will not be able

IBM and DB2 are registered trademarks of the IBM Corporation in the United States and/or other countries.

U.S. Government Users Restricted Rights

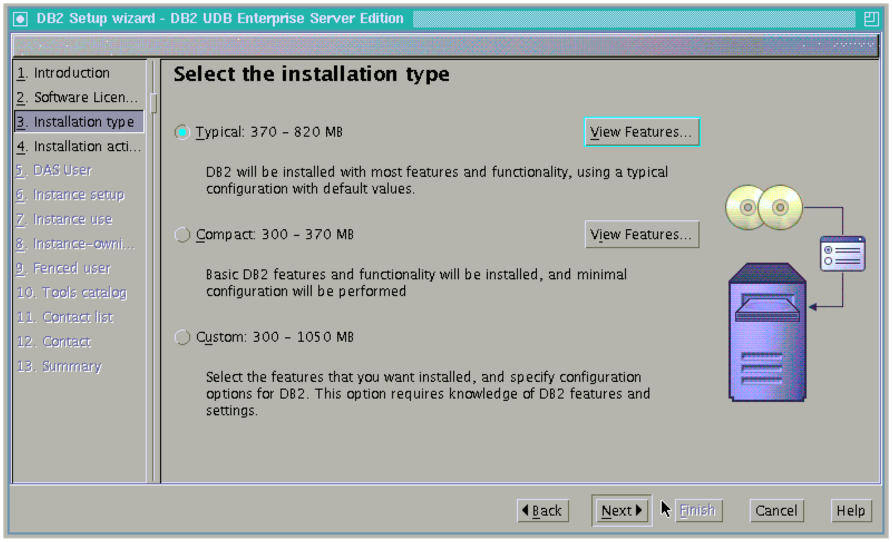
U.S. Government Users Restricted Rights - Use, duplication, or disclosure restricted by the GSA ADP Schedule Contract with the IBM Corporation.

D/N: L-SBEE-57QNCA
P/N: L-SBEE-57QNCA

[✓ Accept] Decline

Figure 2-10 License agreement

5. Select **Typical** as illustrated in Figure 2-11 and click **Next**. We used “Typical” this time.



Select the installation type

[✓ Typical: 370 - 820 MB]

DB2 will be installed with most features and functionality, using a typical configuration with default values.

Compact: 300 - 370 MB

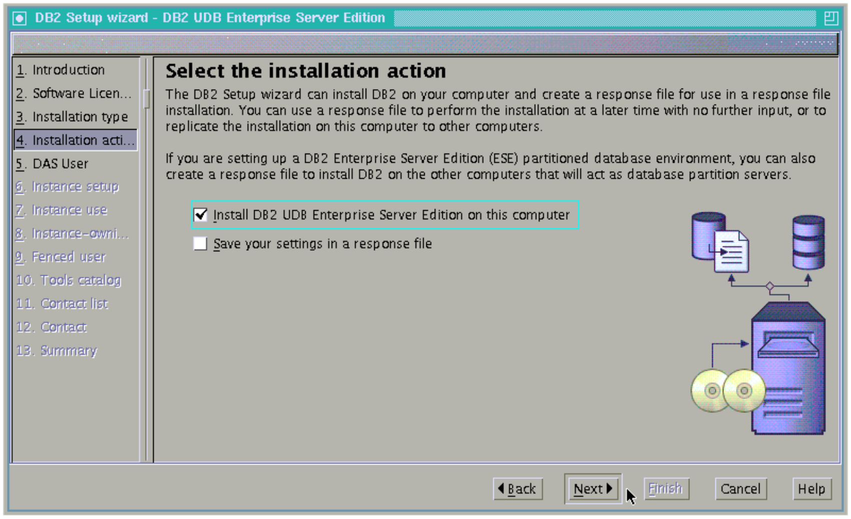
Basic DB2 features and functionality will be installed, and minimal configuration will be performed

Custom: 300 - 1050 MB

Select the features that you want installed, and specify configuration options for DB2. This option requires knowledge of DB2 features and settings.

Figure 2-11 Installation type selection

6. Check **Install DB2 UDB Enterprise Server Edition on this computer** as illustrated in Figure 2-12 and click **Next**. Using the response file option, you can make the file that can be used to do the "silent installation" later. We do not select it now, because we will install the product immediately.


DB2 Setup wizard - DB2 UDB Enterprise Server Edition

Select the installation action

The DB2 Setup wizard can install DB2 on your computer and create a response file for use in a response file installation. You can use a response file to perform the installation at a later time with no further input, or to replicate the installation on this computer to other computers.

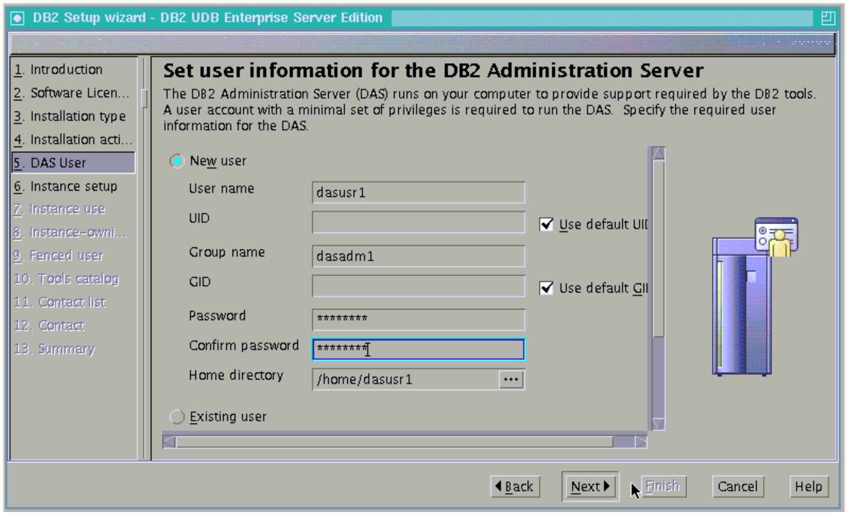
If you are setting up a DB2 Enterprise Server Edition (ESE) partitioned database environment, you can also create a response file to install DB2 on the other computers that will act as database partition servers.

[✓] Install DB2 UDB Enterprise Server Edition on this computer

[ ] Save your settings in a response file

Figure 2-12 Select the installation action

7. DB2 Administration Server (DAS) provides the various administration facilities to the DB2 tools client, such as DB2 Control Center as illustrated in Figure 2-13. Use default values (**New user**) and set the password only. Type twice to confirm. Then click **Next**.


DB2 Setup wizard - DB2 UDB Enterprise Server Edition

1. Introduction
2. Software Licen...
3. Installation type
4. Installation acti...
5. DAS User
6. Instance setup
7. Instance use
8. Instance-owni...
9. Fenced user
10. Tools catalog
11. Contact list
12. Contact
13. Summary

Set user information for the DB2 Administration Server

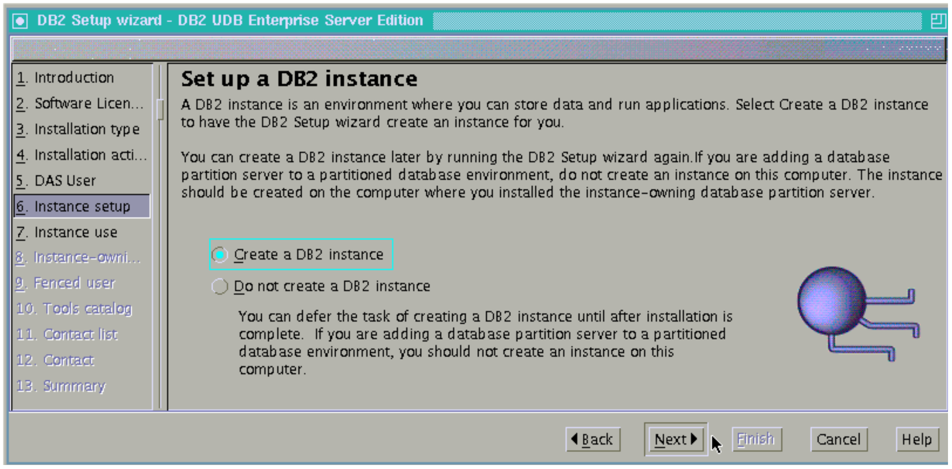
The DB2 Administration Server (DAS) runs on your computer to provide support required by the DB2 tools. A user account with a minimal set of privileges is required to run the DAS. Specify the required user information for the DAS.

New user
User name: dasusr1
UID:
[✓] Use default UID
Group name: dasadm1
GID:
[✓] Use default GID
Password: ********
Confirm password: ********
Home directory: /home/dasusr1

Existing user

Figure 2-13 Setup the DAS user

8. We can create a DB2 instance during the installation. We use this instance as the test bench throughout the book. Select **Create a DB2 Instance** as illustrated in Figure 2-14 and click **Next**.


DB2 Setup wizard - DB2 UDB Enterprise Server Edition

1. Introduction
2. Software Licen...
3. Installation type
4. Installation acti...
5. DAS User
6. Instance setup
7. Instance use
8. Instance-owni...
9. Fenced user
10. Tools catalog
11. Contact list
12. Contact
13. Summary

Set up a DB2 instance

A DB2 instance is an environment where you can store data and run applications. Select Create a DB2 instance to have the DB2 Setup wizard create an instance for you.

You can create a DB2 instance later by running the DB2 Setup wizard again.If you are adding a database partition server to a partitioned database environment, do not create an instance on this computer. The instance should be created on the computer where you installed the instance-owning database partition server.

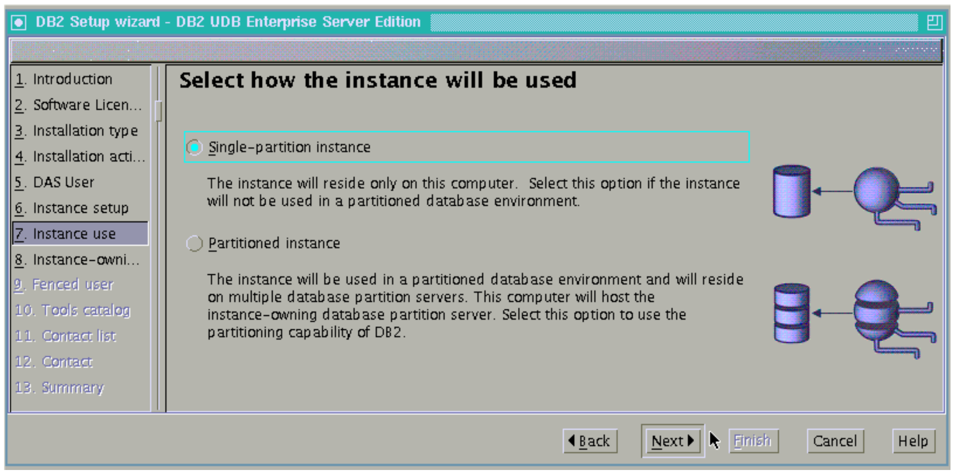
Create a DB2 instance

Do not create a DB2 instance

You can defer the task of creating a DB2 instance until after installation is complete. If you are adding a database partition server to a partitioned database environment, you should not create an instance on this computer.

Figure 2-14 DB2 instance creation selection

9. Select **Single-partition instance** as illustrated in Figure 2-15. Then click **Next**.



Select how the instance will be used

[✓ Single-partition instance]

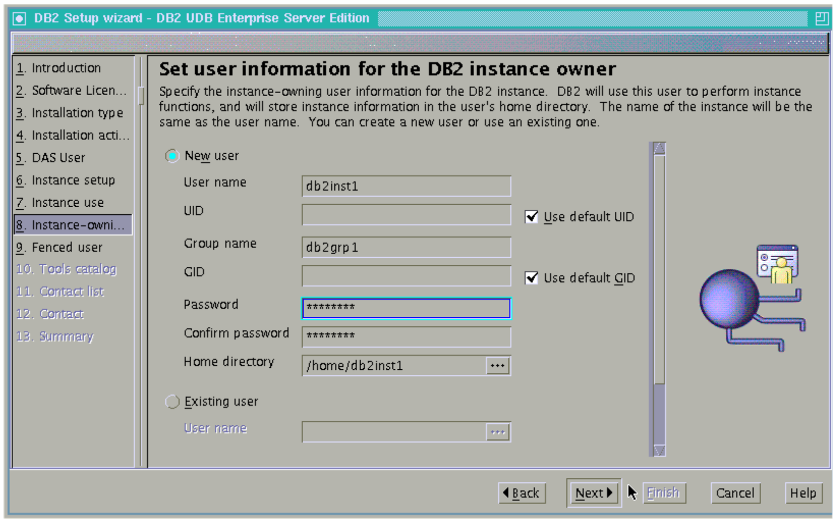
The instance will reside only on this computer. Select this option if the instance will not be used in a partitioned database environment.

Partitioned instance

The instance will be used in a partitioned database environment and will reside on multiple database partition servers. This computer will host the instance-owning database partition server. Select this option to use the partitioning capability of DB2.

Figure 2-15 Select partitioning

10. In the window as illustrated in Figure 2-16, you can set the instance owner information. The instance will have the same name as the instance owner name and the instance directory will be created under the instance owner's home directory. Use all default values (**New user**). You only need to key in the password. Then click **Next**.



Set user information for the DB2 instance owner

Specify the instance-owning user information for the DB2 instance. DB2 will use this user to perform instance functions, and will store instance information in the user's home directory. The name of the instance will be the same as the user name. You can create a new user or use an existing one.

[✓ New user]

User name	db2inst1	
UID		✓ Use default UID
Group name	db2grp1	
GID		✓ Use default GID
Password	********	
Confirm password	********	
Home directory	/home/db2inst1	

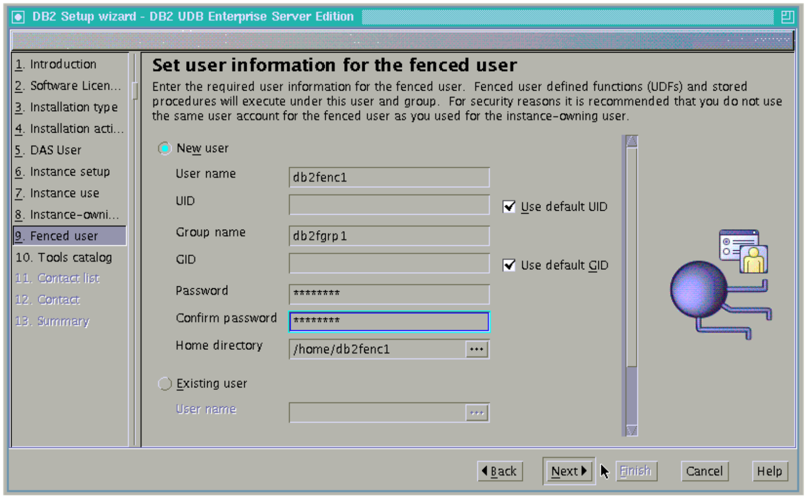
Existing user

User name

Figure 2-16 Set the instance owner

11. For security reasons, the DB2 user defined functions (UDF) or stored procedures which are defined as "fenced" can run as different users from the

usual DB2 user. You have to create or specify the user here. Use defaults (**New user**) and set the password as illustrated in Figure 2-17. Then click **Next**.



DB2 Setup wizard - DB2 UDB Enterprise Server Edition

Set user information for the fenced user

Enter the required user information for the fenced user. Fenced user defined functions (UDFs) and stored procedures will execute under this user and group. For security reasons it is recommended that you do not use the same user account for the fenced user as you used for the instance-owning user.

[✓ New user]

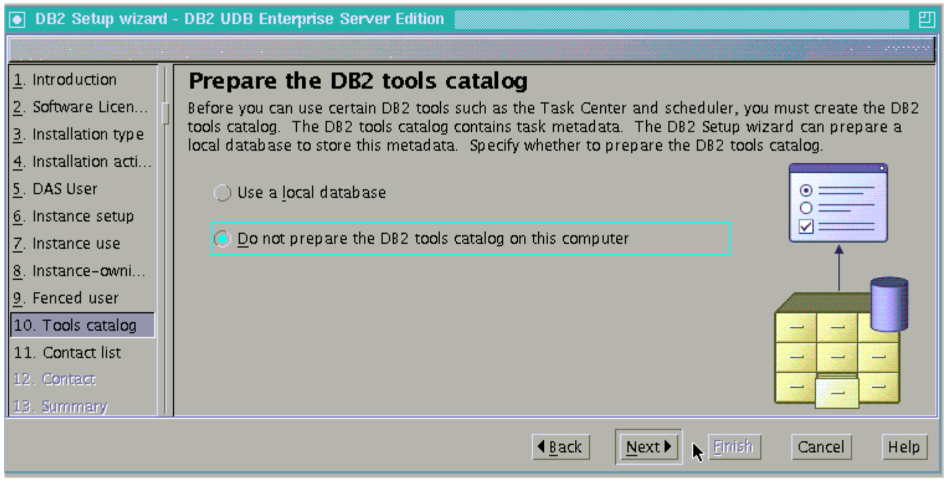
User name	db2fenc1	
UID		[✓] Use default UID
Group name	db2fgrp1	
GID		[✓] Use default GID
Password	********	
Confirm password	********	
Home directory	/home/db2fenc1	

Existing user

User name

Figure 2-17 Set the fenced UDF user

12.From this point, you will work with DB2 tools catalog, administration contact information, and health monitor notification information. The use of these are out of the scope of this redbook. For now, we will just take the default options and move on. Select **Do not prepare the DB2 tools catalog on this computer** as illustrated in Figure 2-18, and then click **Next**.



DB2 Setup wizard - DB2 UDB Enterprise Server Edition

Prepare the DB2 tools catalog

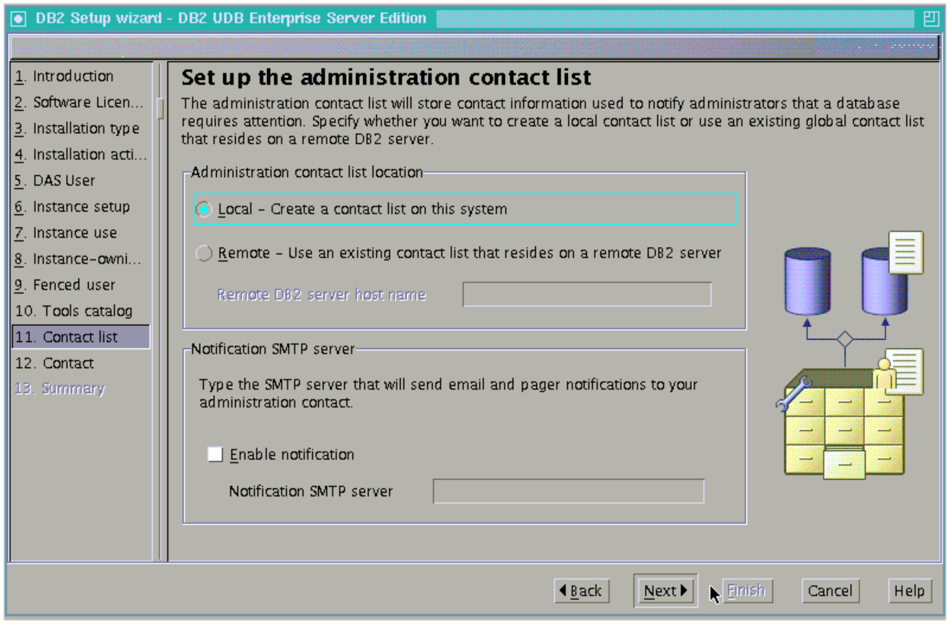
Before you can use certain DB2 tools such as the Task Center and scheduler, you must create the DB2 tools catalog. The DB2 tools catalog contains task metadata. The DB2 Setup wizard can prepare a local database to store this metadata. Specify whether to prepare the DB2 tools catalog.

Use a local database

[✓ Do not prepare the DB2 tools catalog on this computer]

Figure 2-18 Prepare the DB2 tools catalog

13.Select **Local - Create a contact list on this system** as illustrated in Figure 2-19, and then click **Next**.



Set up the administration contact list

The administration contact list will store contact information used to notify administrators that a database requires attention. Specify whether you want to create a local contact list or use an existing global contact list that resides on a remote DB2 server.

Administration contact list location

[✓ Local - Create a contact list on this system]

Remote - Use an existing contact list that resides on a remote DB2 server

Remote DB2 server host name

Notification SMTP server

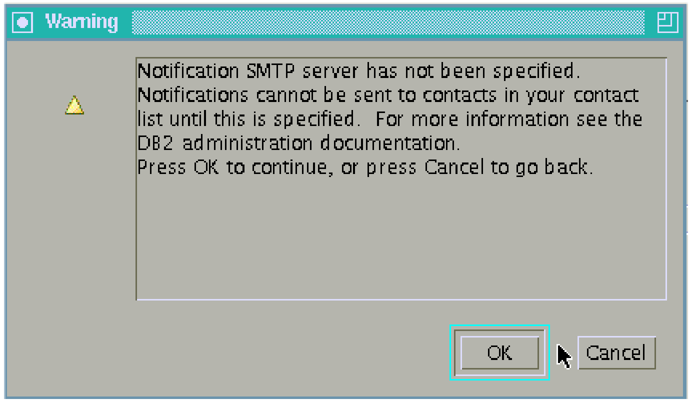
Type the SMTP server that will send email and pager notifications to your administration contact.

Enable notification

Notification SMTP server

Figure 2-19 Set up the administration contact list

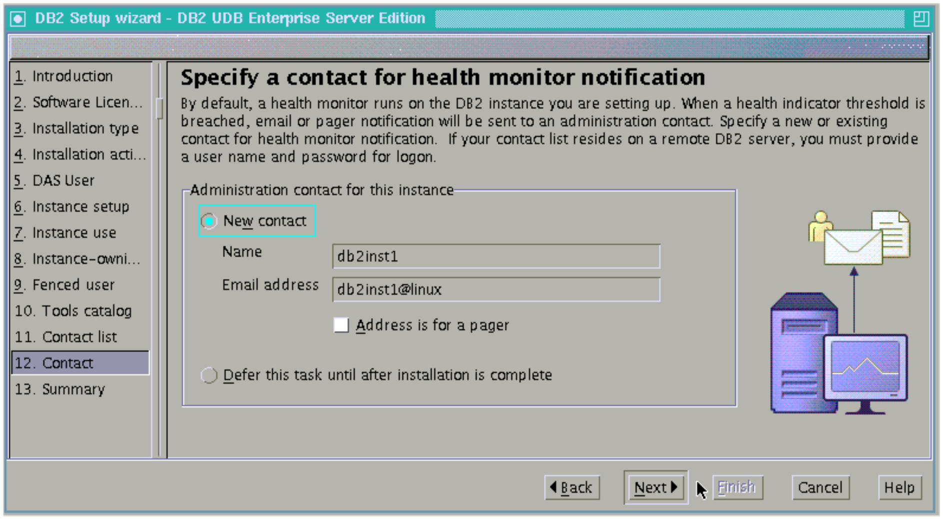
14.Ignore the warning message as shown in Figure 2-20. Click **OK**.



Notification SMTP server has not been specified. Notifications cannot be sent to contacts in your contact list until this is specified. For more information see the DB2 administration documentation.
Press OK to continue, or press Cancel to go back.

Figure 2-20 Warning error for the administration contact list

15.Select **New contact**, use default values as illustrated in Figure 2-21, and then click **Next**.



Specify a contact for health monitor notification

By default, a health monitor runs on the DB2 instance you are setting up. When a health indicator threshold is breached, email or pager notification will be sent to an administration contact. Specify a new or existing contact for health monitor notification. If your contact list resides on a remote DB2 server, you must provide a user name and password for logon.

Administration contact for this instance

[● New contact]

Name: db2inst1

Email address: db2inst1@linux

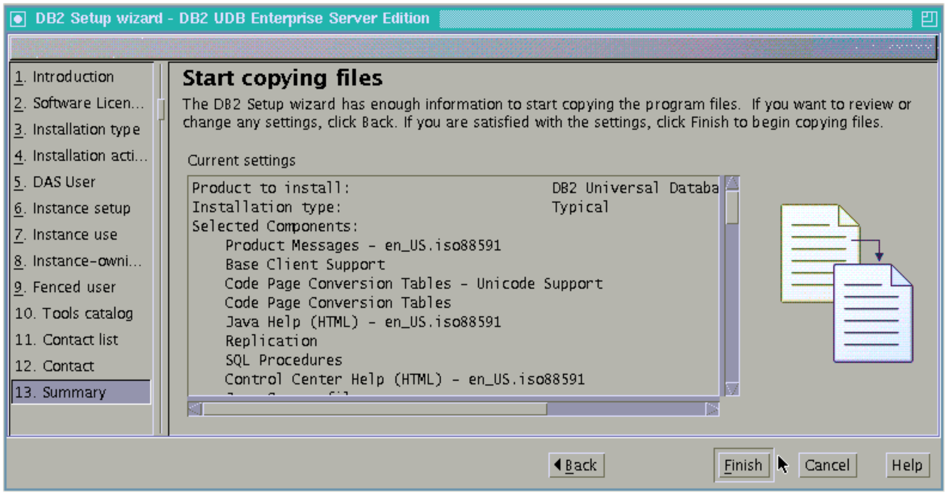
☐ Address is for a pager

○ Defer this task until after installation is complete

Figure 2-21 Health monitor notification information

16. The window shown in Figure 2-22 is the summary information of the DB2 you will install. Click **Finish** to start copying the files from the CDROM to the disk.

.



Start copying files

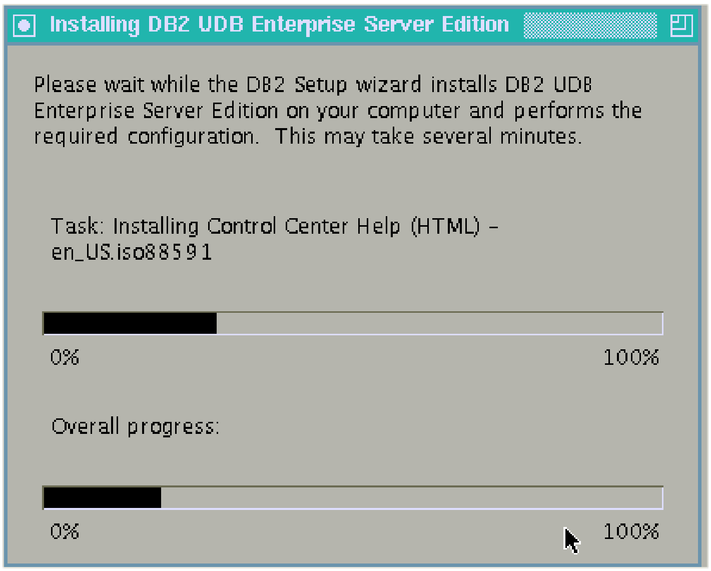
The DB2 Setup wizard has enough information to start copying the program files. If you want to review or change any settings, click Back. If you are satisfied with the settings, click Finish to begin copying files.

Current settings

```
Product to install:                         DB2 Universal Databa
Installation type:                          Typical
Selected Components:
    Product Messages - en_US.iso88591
    Base Client Support
    Code Page Conversion Tables - Unicode Support
    Code Page Conversion Tables
    Java Help (HTML) - en_US.iso88591
    Replication
    SQL Procedures
    Control Center Help (HTML) - en_US.iso88591
```

Figure 2-22 The summary information

17. It depends on the processor power but the setup will take a few minutes while the system displays the window, as illustrated in Figure 2-23.


Installing DB2 UDB Enterprise Server Edition

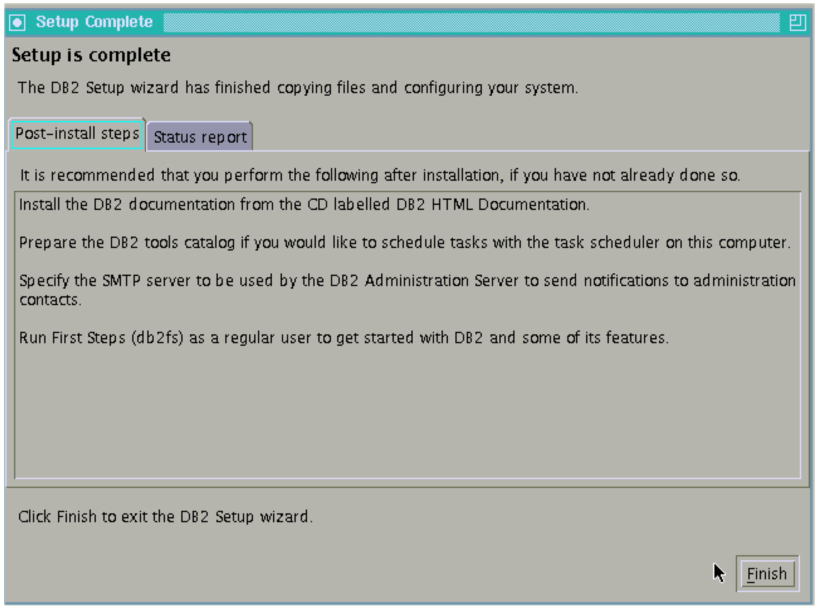
Please wait while the DB2 Setup wizard installs DB2 UDB Enterprise Server Edition on your computer and performs the required configuration. This may take several minutes.

Task: Installing Control Center Help (HTML) - en_US.iso88591

Overall progress:

Figure 2-23 Copying files

18. Figure 2-24 is the window which shows the completion of the installation. Check the **Status Report** tab and click **Finish**.


Setup Complete

Setup is complete

The DB2 Setup wizard has finished copying files and configuring your system.

It is recommended that you perform the following after installation, if you have not already done so.

Install the DB2 documentation from the CD labelled DB2 HTML Documentation.

Prepare the DB2 tools catalog if you would like to schedule tasks with the task scheduler on this computer.

Specify the SMTP server to be used by the DB2 Administration Server to send notifications to administration contacts.

Run First Steps (db2fs) as a regular user to get started with DB2 and some of its features.

Click Finish to exit the DB2 Setup wizard.

Figure 2-24 Completion of the installation

The database objects created

During the installation, the following database objects are created:

- The DB2 Administration Server (DAS)

  The user who runs DAS is **`dasusr1`** and its group is **`dasadm1`**.

- A DB2 instance **`db2inst1`**

  The instance owner user is **`db2inst1`** and its group is **`db2grp1`**. The instance directory is **`/home/db2inst1`**.

- A user **`db2fenc1`**

  This user runs fenced UDFs or stored procedures. Its group **`db2fgrp1`** is also created.

We will use some of these objects and names on the sample programs illustrated in Chapter 3, “Connectivity with DB2 UDB for iSeries” on page 35.

2.3 Create and test the sample database

The DB2 product has now been installed. The next task is creating the database. This section describes the process of creating and testing the sample database.

2.3.1 Create the db2 sample database

After the installation is complete, you can run launch the DB2 First Steps (db2fs) tool which provides the first-step-guide of DB2 UDB. We create a DB2 sample database using db2fs and then test that it works successfully. The sample database is used by the sample programs throughout this book.

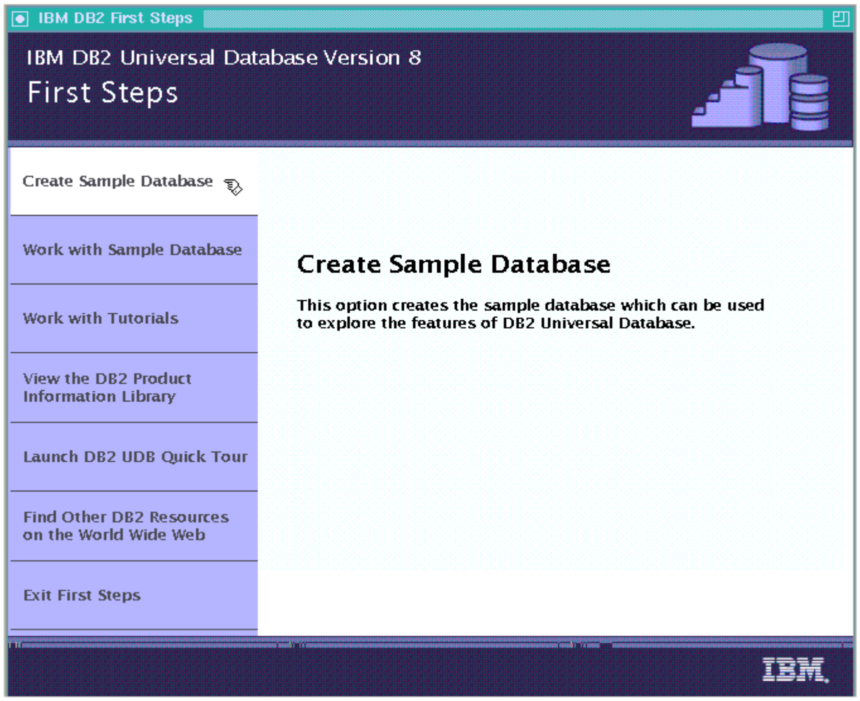
1. The easiest way to start db2fs is to launch it from the vnc client of the user db2inst1 (the instance owner).

```
linux02:~ # su - db2inst1
db2inst1@linux02:~> vncserver

You will require a password to access your desktops.

Password:
Verify:

New 'X' desktop is linux02:2

Creating default startup script /home/db2inst1/.vnc/xstartup
Starting applications specified in /home/db2inst1/.vnc/xstartup
Log file is /home/db2inst1/.vnc/linux02:6.log

db2inst1@linux02:~> logout
linux02:~ #
```

Figure 2-25 Start the vnc server as the instance owner

After switching the user to **`db2inst1`**, start the vncserver as illustrated in Figure 2-25. The desktop number is 2 in this example.

2. After connecting to the db2int1's desktop from a client, type `db2fs` from a xterm window. Then the db2cc main window appears as illustrated in Figure 2-26. To create the sample database, click **Create Sample Database**.

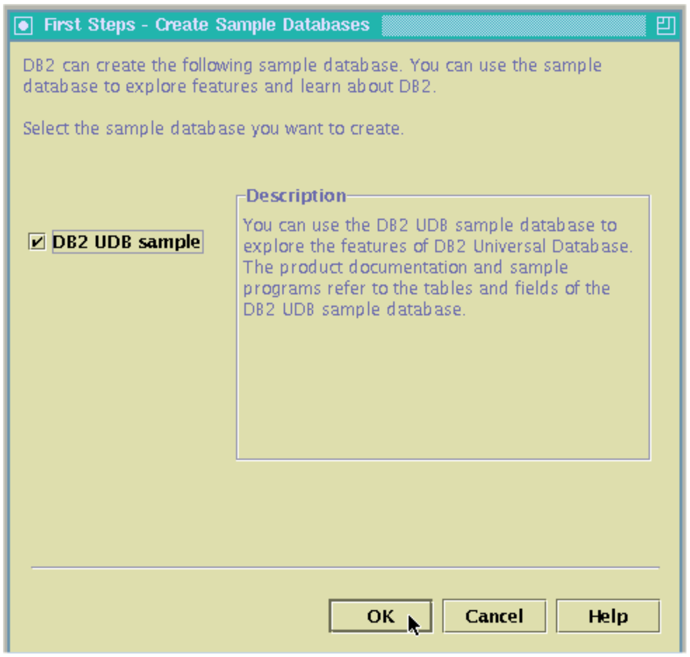

IBM DB2 First Steps
IBM DB2 Universal Database Version 8
First Steps
Create Sample Database
Work with Sample Database
Work with Tutorials
View the DB2 Product Information Library
Launch DB2 UDB Quick Tour
Find Other DB2 Resources on the World Wide Web
Exit First Steps
IBM

Create Sample Database

This option creates the sample database which can be used to explore the features of DB2 Universal Database.

Figure 2-26 DB2 First Steps main window

3. Check **DB2 UDB sample** as illustrated in Figure 2-27 and click **OK**.


First Steps - Create Sample Databases

DB2 can create the following sample database. You can use the sample database to explore features and learn about DB2.

Select the sample database you want to create.

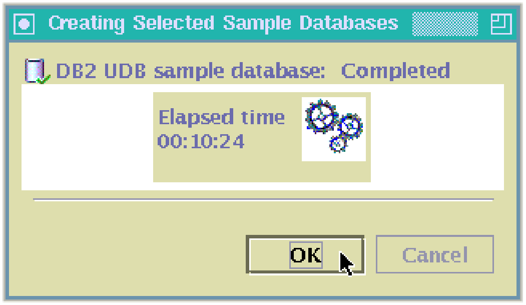
[✓] DB2 UDB sample

Description

You can use the DB2 UDB sample database to explore the features of DB2 Universal Database. The product documentation and sample programs refer to the tables and fields of the DB2 UDB sample database.

Figure 2-27   DB2 sample database creation

4. After a short time, the sample database will be created. During the creation, this window appears, as in Figure 2-28.


Creating Selected Sample Databases

DB2 UDB sample database: Completed

Elapsed time
00:10:24

Figure 2-28   Creating the sample database

This step takes a few minutes depending on the processing power. It took 10 minutes, in our case; the Linux partition has 0.5 processor and the host partition has 0.5 processor.

2.3.2 Test the sample database

After creating the sample database, you can test it by using the DB2 Command Line Processor (CLP). Figure 2-29 shows the procedure.

From a bash prompt of the user db2inst1, type `db2` which starts the CLP. To connect to the database, type the following command and press **Enter**. The name of the database is `sample`. Note that the following db2 commands and SQL statements are case insensitive.

```
connect to sample
```

To execute a simple SQL, type the following SQL statement and press **Enter**:

```
select * from staff where ID > 200
```

`staff` is the name of a table of the sample database. If the SQL worked, you can see the results as in Figure 2-29.

This command disconnects from the database, ends the CLP and terminates the CLP back-end process.

```
terminate
```

```
db2inst1@linux:~> db2
(c) Copyright IBM Corporation 1993,2002
Command Line Processor for DB2 SDK 8.1.2

You can issue database manager commands and SQL statements from the command
prompt. For example:
    db2 => connect to sample
    db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
 ? CATALOG DATABASE for help on the CATALOG DATABASE command
 ? CATALOG          for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => connect to sample

   Database Connection Information

 Database server        = DB2/LINUXPPC 8.1.2
 SQL authorization ID   = DB2INST1
 Local database alias   = SAMPLE

db2 => select * from staff where ID > 200

ID     NAME      DEPT   JOB   YEARS  SALARY    COMM
------ --------- ------ ----- ------ --------- ---------
   210 Lu            10 Mgr       10  20010.00         -
   220 Smith         51 Sales      7  17654.50    992.80
   230 Lundquist     51 Clerk      3  13369.80    189.65
   240 Daniels       10 Mgr        5  19260.25         -
   250 Wheeler       51 Clerk      6  14460.00    513.30
   260 Jones         10 Mgr       12  21234.00         -
   270 Lea           66 Mgr        9  18555.50         -
   280 Wilson        66 Sales      9  18674.50    811.50
   290 Quill         84 Mgr       10  19818.00         -
   300 Davis         84 Sales      5  15454.50    806.10
   310 Graham        66 Sales     13  21000.00    200.30
   320 Gonzales      66 Sales      4  16858.20    844.00
   330 Burke         66 Clerk      1  10988.00     55.50
   340 Edwards       84 Sales      7  17844.00   1285.00
   350 Gafney        84 Clerk      5  13030.50    188.00

  15 record(s) selected.

db2 => terminate
DB20000I  The TERMINATE command completed successfully.
db2inst1@linux:~>
```

Figure 2-29 Test the sample database using DB2 CLP

3

Connectivity with DB2 UDB for iSeries

This chapter discusses connectivity options between the resources on the OS/400 side and the Linux side. Among many possible scenarios, we focus on how Linux applications access DB2 UDB for iSeries using DB2 Connect.

3.1 Connectivity overview

Figure 3-1 shows the possible scenarios when it comes to connectivity between OS/400 resources and Linux resources. We have two LPAR partitions where one partition is for OS/400 and the other is for Linux.

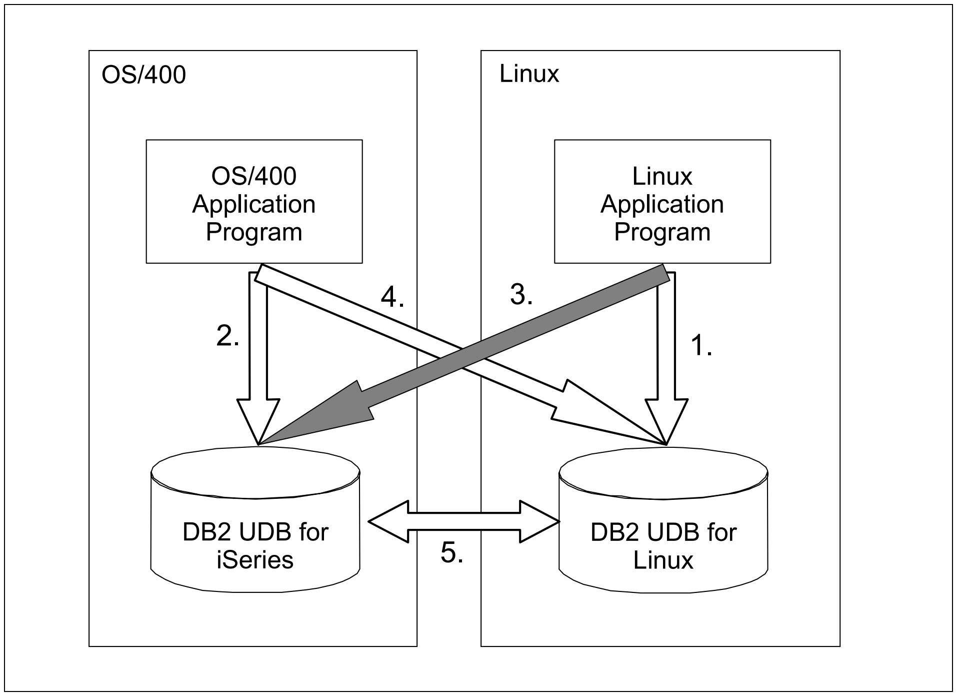


Figure 3-1 OS/400 and Linux DB2 connectivity

We illustrate five possible scenarios, but mainly focus on Scenario 3:

- **Scenario 1** is where Linux applications access their local database, which is DB2 UDB for Linux on iSeries. This is a general applications scenario and there is readily available information for this scenario in the redbook *Up and Running with DB2 for Linux*, SG24-6899.
- **Scenario 2** is the same as Scenario 1, but instead is on the other side of partition: the OS/400 partition. Again, there is enough readily available information for this scenario.
- **Scenario 3** is where Linux applications access DB2 UDB for iSeries using DB2 Connect, which is one of the products offered by DB2 UDB for Linux on iSeries. You can access the DB2 UDB for iSeries from the Linux application transparently as the local Linux database.

- **Scenario 4** accesses DB2 UDB for Linux from OS/400 applications. The OS/400 application program can access the database in Linux via an SQL interface such as CLI (DB2 Call Level Interface), Embedded SQL, JDBC, or SQLJ. You will find some examples in Appendix A, “Accessing DB2 UDB for Linux from OS/400 application” on page 105.
- **Scenario 5** is the database replication between DB2 UDB for iSeries and Linux. This scenario is possible when the Data Propagator is installed on OS/400, and the DB2 UDB for Linux Enterprise Server Edition is installed.

This book focuses on Scenario 3. The various connectivity methods such as CLI, ODBC, Embedded SQL, JDBC, and SQLJ of Scenario 3 are discussed and the configuration samples and sample programs are provided.

3.2 DB2 Connect introduction

Again, our main focus is Scenario 3 and DB2 Connect is the key product facilitating that scenario. Now we introduce DB2 Connect before we discuss actual scenarios of the various access methods.

3.2.1 Overview

DB2 Connect is the product which enables Linux, UNIX or Windows applications to access DB2 UDB for iSeries (OS/400) using DRDA (Distributed Relational Database Architecture). It is an efficient facility to integrate DB2 UDB for iSeries and DB2 UDB for Linux on iSeries.

Although there are other native methods to access DB2 UDB for iSeries from Linux, such as ODBC or JDBC drivers, DB2 Connect is the only way to access DB2 UDB for iSeries from Linux applications using DRDA.

Since it uses DRDA, users can access DB2 UDB for iSeries transparently as a DB2 database. User application programs do not have to be aware that the operating system is OS/400. It means that the same application programming interface is provided to access DB2 UDB for iSeries and DB2 UDB for Linux/UNIX/Windows (LUW).

Figure 3-2 introduces two basic elements of DRDA: Application Requester (AR) and Application Server (AS). Again, the benefit of DRDA is your DB access to applications can be transparent from DB serving platforms. The coordinated functions of AR and AS hide the differences to the applications. When using a distributed relational database, the system on which the application program runs is called the application requester (AR), and the system on which the remote data resides is called the application server (AS). The term *client* is often used interchangeably with AR, and *server* with AS.

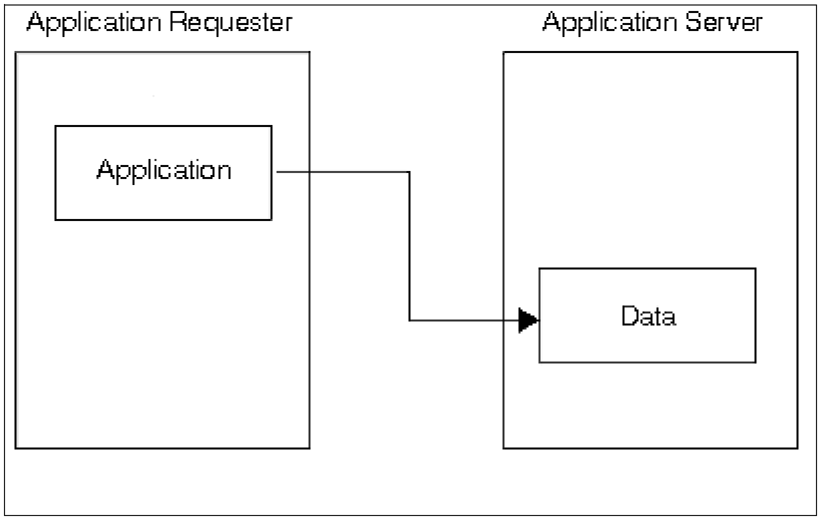


Figure 3-2 A distributed relational database

DB2 UDB for iSeries supports both AS and AR. In case of DB2 UDB for LUW, only Enterprise Edition supports them both. Other editions only support AS and you need to use DB2 Connect in case you need AR functionality.

Therefore, if your Linux applications want to access DB2 UDB for iSeries in a DRDA manner, you will need DB2 Connect on the Linux side. For the reverse, you don't need DB2 Connect on the OS/400 side. Since DB2 UDB for LUW has DRDA AS and DB2 UDB for iSeries has DRDA AR as their standard functions, DB2 Connect is not necessary to access DB2 UDB for LUW.

DB2 Connect is included in DB2 UDB Enterprise Server Edition (ESE), or can be purchased as a single product. Some other developer's editions of DB2 UDB also contain DB2 Connect facility. There are many editions of DB2 Connect which provides various functionality, packaging, or licensing. But from the technical view point, there are only two editions, Personal Edition and Enterprise Edition.

The DB2 Connect Personal Edition supports only the direct connection from the machine DB2 Connect installed to DB2 UDB for iSeries. The Enterprise Edition can support the intermediate connection between the DB2 clients and DB2 UDB for iSeries as a gateway as illustrated in Figure 3-3. The DB2 client which connects to DB2 Connect Enterprise Edition must have DB2 *Client*, formerly known as DB2 Client Application Enabler (CAE) facility.

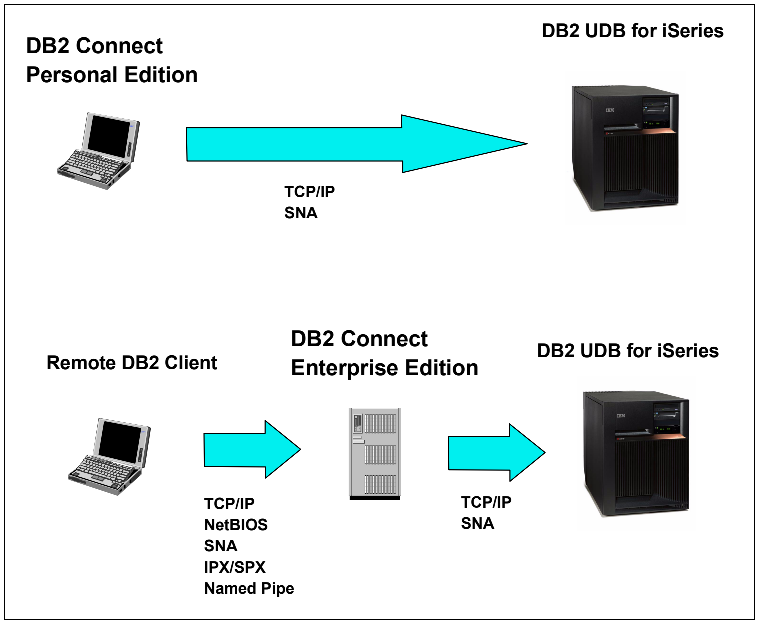


Figure 3-3 DB2 Connect Personal and Enterprise Edition

Note: Here are a couple of points about Figure 3-3:

- DB2 Connect Personal Edition is not available for Linux on iSeries.
- For network protocol, only TCP/IP is supported for Linux on iSeries.

There are yet other editions of DB2 Connect which can be or must be selected according to the user situation. For further information, refer to:

- "Which Edition of DB2 Connect is Right for You?", which can be found at:

`http://www7b.software.ibm.com/dmdd/library/techarticle/0303zikopoulos1/0303zikopoulos1.html`

In this book, we use DB2 Connect which is included in DB2 UDB Enterprise system.

Using DB2 Connect, the database of an application can be easily switched between DB2 UDB for LUW and DB2 UDB for iSeries. Even the concurrent access to both DB2 UDB for iSeries and DB2 UDB for LUW is available using DB2 Connect, in other words, DRDA. Application transparency is the benefit of using DB2 Connect over other connectivity methods such as the Toolbox JDBC driver or the iSeries ODBC Driver for Linux.

DB2 Connect provides various programming interfaces such as CLI, ODBC, JDBC, Embedded SQL, and SQLJ.

3.2.2 DB2 Connect and .NET Framework Application

Microsoft .NET Framework technology is fully supported in the DB2 UDB product family since Version 8.1.2. DB2 UDB provides the .NET application in three different ADO.NET manners to access data:

- DB2 ODBC driver for ODBC.NET
- DB2 OleDb support for OleDb.NET
- DB2 native .NET Framework data providers

Figure 3-4 describes the relationship between ADO.NET and DB2 UDB.

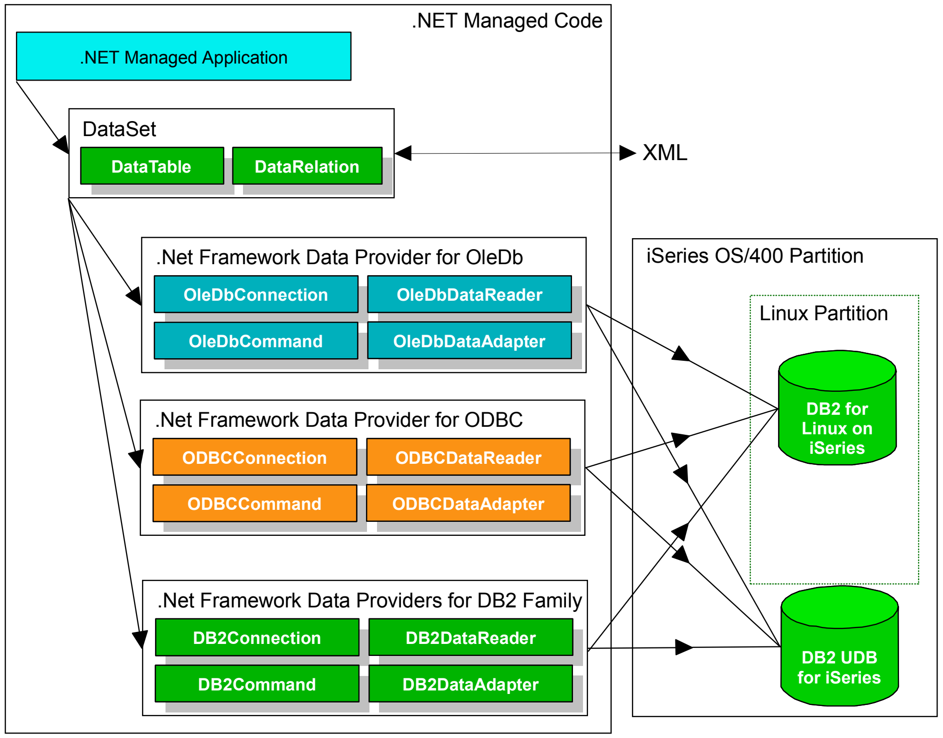


Figure 3-4 ADO.NET applications accessing DB2 UDB

Data Providers

ADO.NET is an improvement to Microsoft ActiveX Data Objects (ADO) programming interface. ADO.NET consists of a set of classes that .NET applications can use to access data in a database. The data model for ADO.NET is primarily used with a disconnected dataset. ADO.NET will, for instance, get data from a data provider, manipulate the data in a disconnected manner, and then send the data back to the data provider to, in turn, update data on the database server. A .NET data provider is used for connecting to a database, executing commands, and retrieving results.

The .NET Framework data providers for OleDb/ODBC are used to access OleDb/ODBC compliant data sources. Both DB2 UDB for Linux on iSeries and DB2 UDB for iSeries can be accessed in these ways. These two data providers are provided by Microsoft .NET by default.

The .NET Framework data provider for DB2 for LUW (including DB2 for Linux on iSeries) is provided along with DB2 Development Client. IBM plans to include a .NET Framework Data Provider for DB2 UDB for iSeries in the iSeries Access for Windows product in the release after V5R2. The .NET Data Provider will allow applications written within the Microsoft® .NET framework to access the DB2 UDB for iSeries. The .NET Data Provider is part of the iSeries Access for Windows Beta. For further information, visit:

http://www-1.ibm.com/servers/eserver/iseries/access/planning.html

Other required components

In the Microsoft ADO.NET structure, some other components are also necessary:

- DB2 Client must be installed when you use ADO.NET to access DB2 UDB for LUW.
- IBM iSeries Access for Windows must be installed when you use ADO.NET to access DB2 UDB for iSeries.
- The driver components, like ODBC drivers, need to be configured properly.

.NET applications accessing DB2 UDB via DB2 Connect

We can also use the DB2 Connect as a gateway to support access from ADO.NET applications. DB2 Connect is configured in a Linux partition. It is a part of DB2 UDB Enterprise Server Edition features or a separate product.

Figure 3-5 describes this alternative way to access DB2 UDB for iSeries via DB2 Connect.

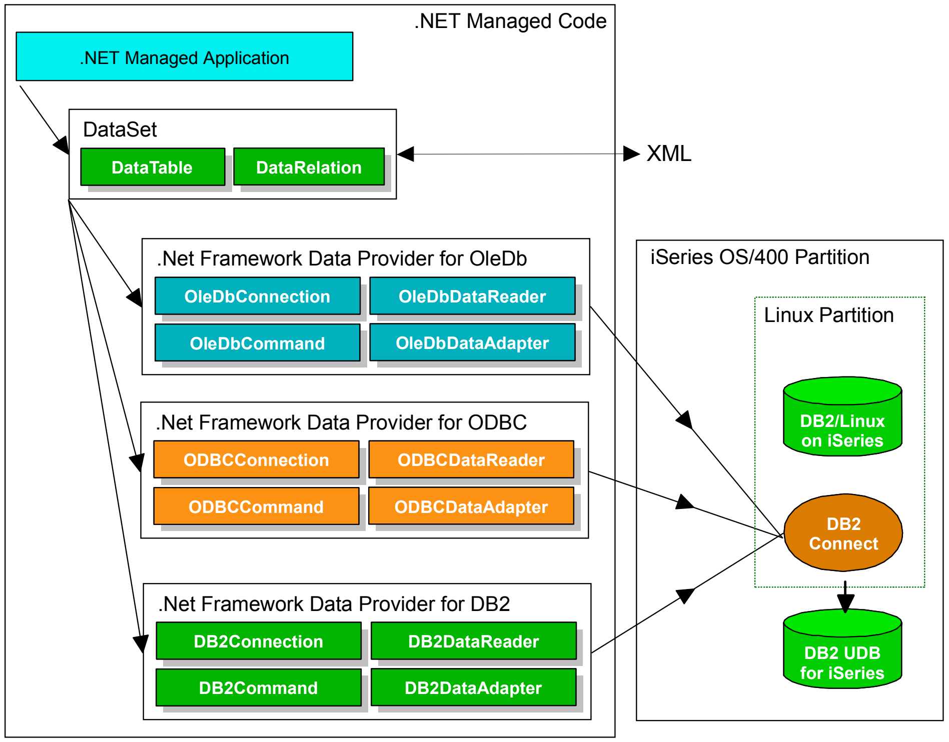


Figure 3-5 .NET applications accessing DB2 UDB for iSeries via DB2 Connect

The advantages of this configuration include:

- A better way to secure you iSeries databases structure
- A better way to manipulate different data sources
- A better way to implement cross platform programming

3.3 How to configure DB2 Connect

This section provides a step-by-step guide for configuring DB2 Connect.

3.3.1 The environment

Figure 3-6 shows the environment of the Linux and OS/400 partition which is used throughout this book. Two LPAR partitions (Linux and OS/400) are connected via the virtual LAN 192.168.1.0 network. The sample database created in Chapter 2 resides in the Linux partition and we will create another sample database in which the schema (library) name will be SAMPLEDB01 in the OS/400 partition.

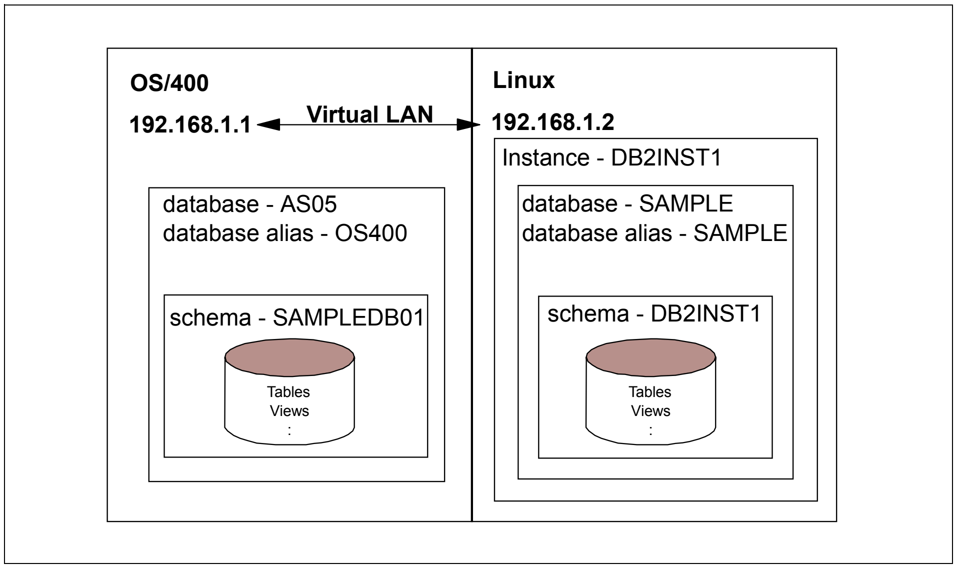


Figure 3-6 The system environment for the sample programs

3.3.2 Configuration parameters

We need to configure the DB2 Connect DRDA environment, where the DB2 Connect in the Linux partition is the AR and the DB2 UDB for iSeries is the AS. Actually this configuration can be done by updating the DB2 node or database directory using DB2 Configuration Assistant (**`db2ca`**). **`db2ca`** is a DB2 GUI tool to add, remove, or alter database configuration.

The configuration information of the DB2 Connect is stored in the *node directory* and the *database directory* of the instance (**`db2inst1`** in our case). We use the settings which are illustrated in Figure 3-7.

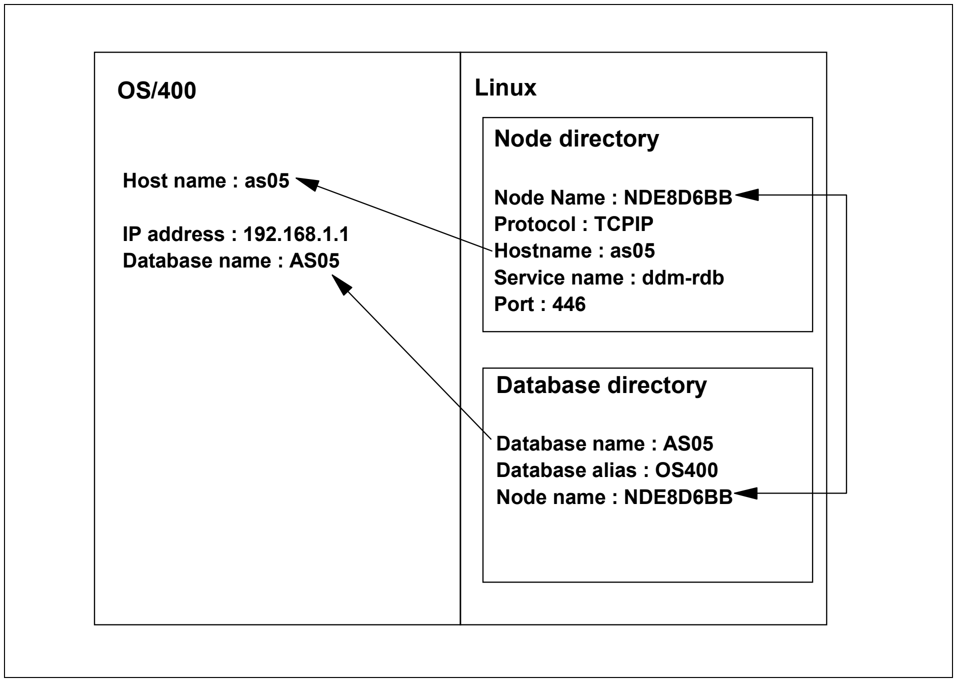


Figure 3-7 DB2 Connect settings

Node name

The node name is the name of the DB2 UDB for iSeries system registered in the node directory of Linux side. The name NDEXXXXX is automatically assigned by the DB2 Configuration Assistant tool. This value must be the same in the node and the database directory but actually it is not related to the OS/400 side definition. That is why the db2ca generates a random name automatically in the Linux side.

Protocol

The protocol is always TCPIP on the Linux platform.

Host name

The hostname specified in the Linux side node directory is the host name or the IP address of the OS/400 partition. In our case, we use as05 as the OS/400 host name which is registered in the `/etc/hosts` file. If you do not want to update the `hosts` file, you can simply use the IP address (192.168.1.1) as the host name.

Service name

The TCP/IP service name of the DRDA DDM service is on the Linux side. The *well known* port of the DRDA DDM service is 446. In the `/etc/services` file, the port 446 is registered as the service name `ddm-rdb`. We use the name `ddm-rdb` and the port number 446 for the Linux configuration.

On the OS/400 side, the same port is registered as the service name `drda` and the OS/400 DDM server is working on the port.

Database name

The database name of the DB2 UDB for iSeries is shown as the OS/400 *LOCAL database name in OS/400 WRKRDBDIRE command. It must match the Linux side database directory entry.

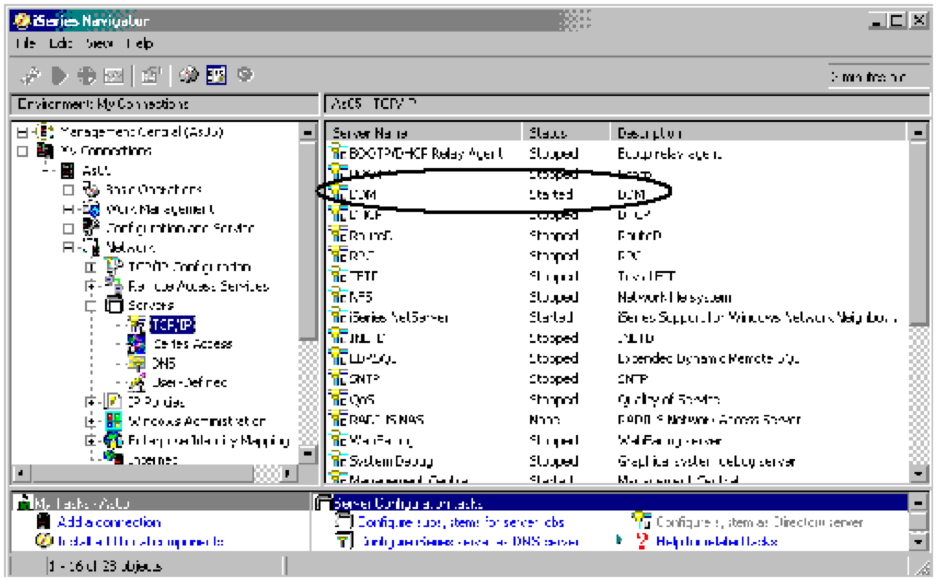
Database alias

The database alias in the database directory is the name which is used in the CONNECT TO statement on the Linux side to connect to the DB2 UDB for iSeries. Though we use the name `OS400` as the alias here, you can register the database name itself as alias.

3.3.3 Configuration on the OS/400 partition

In a typical environment, no special configuration needs to be connected from DB2 Connect. Pay attention to the following points.

- Check the relational database directory entry of the machine (WRKRDBDIRE command). The relational database name where the location is *LOCAL is the name which must be registered in the database directory of the Linux side.
- Confirm that the DDM server is working. If not, start it from the iSeries Navigator or run STRTCPSVR SERVER(*DDM). Figure 3-8 shows the DDM server is working. Select **Network**->**Servers**->**TCP/IP** to get this window from the iSeries Navigator. To start, right-click the server name `DDM` and select **Start**.


iSeries Navigator

Server Name	Status	Description
BOOTP/DHCP Relay Agent	Stopped	Bootp relay agent
[illegible]	Stopped	[illegible]
DDM	Started	DDM
DHCP	Stopped	DHCP
RouteD	Stopped	RouteD
REXEC	Stopped	REXEC
TFTP	Stopped	Trivial FTP
NFS	Stopped	Network File system
iSeries NetServer	Started	iSeries Support for Windows Network Neighborhood
INETD	Stopped	INETD
EDRSQL	Stopped	Extended Dynamic Remote SQL
SNTP	Stopped	SNTP
QoS	Stopped	Quality of Service
RADIUS NAS	None	RADIUS Network Access Server
WebFacing	Stopped	WebFacing server
System Debug	Stopped	Graphical system debug server
Management Central	Started	Management Central

1 - 16 of 23 objects

Figure 3-8   Check the DDM server is working with iSeries Navigator

3.3.4 Configuration on the Linux partition

The DB2 Connect configuration can be done using DB2 Configuration Assistant (**db2ca**) tool in a X window. Or you can also use CATALOG command to update the node or database directory manually. In this example, we use **db2ca**.

To launch the **db2ca**, type **db2ca** and press **Enter** from a terminal emulator on a db2inst1's X desktop.

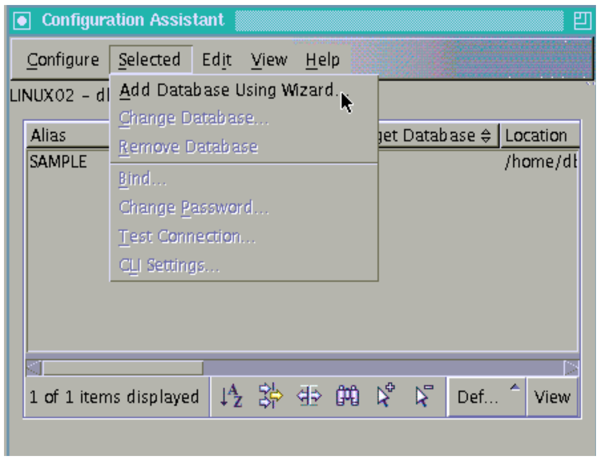
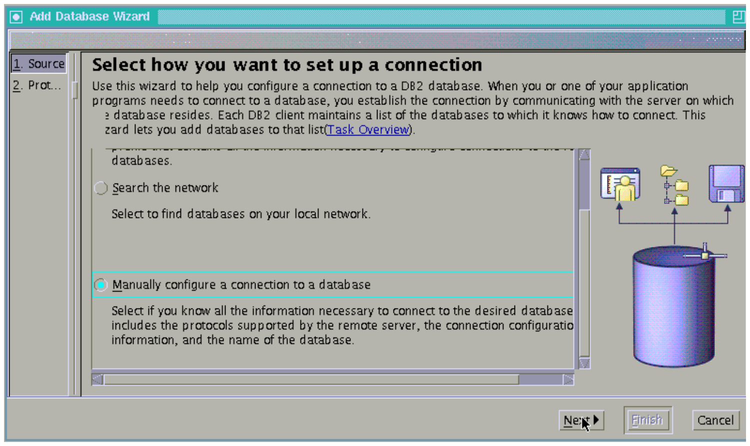


Figure 3-9 DB2 Configuration Assistant main window

Figure 3-9 is the main window of the db2ca. We can see that there is only one “sample” database that is configured in this instance. It was created in Chapter 2. Select **Selected->Add Database Using Wizard**.



Select how you want to set up a connection

Use this wizard to help you configure a connection to a DB2 database. When you or one of your application programs needs to connect to a database, you establish the connection by communicating with the server on which e database resides. Each DB2 client maintains a list of the databases to which it knows how to connect. This zard lets you add databases to that list(Task Overview).

databases.

Search the network

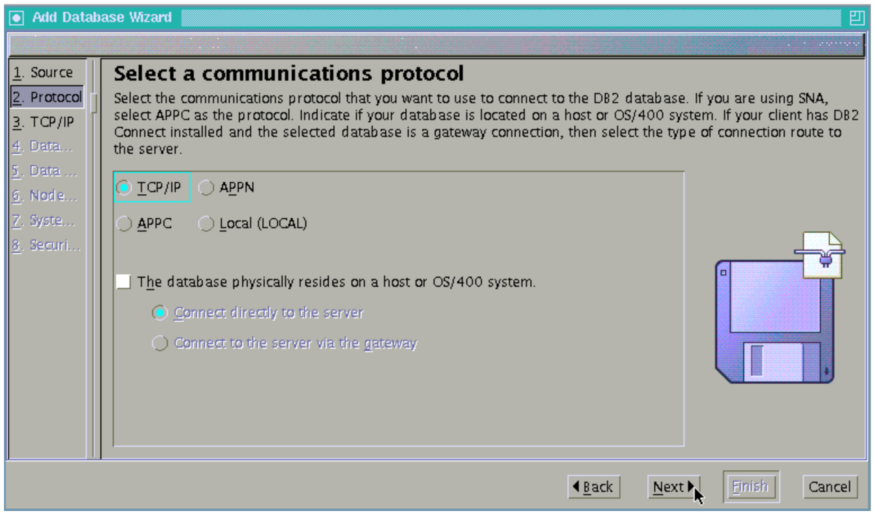
Select to find databases on your local network.

Manually configure a connection to a database

Select if you know all the information necessary to connect to the desired database includes the protocols supported by the remote server, the connection configuratio information, and the name of the database.

Figure 3-10 Add Database Wizard initial window

Select **Manually configure a connection to a database** in the window as seen in Figure 3-10. Click **Next**.



Select a communications protocol

Select the communications protocol that you want to use to connect to the DB2 database. If you are using SNA, select APPC as the protocol. Indicate if your database is located on a host or OS/400 system. If your client has DB2 Connect installed and the selected database is a gateway connection, then select the type of connection route to the server.

[✓ TCP/IP] APPN

APPC Local (LOCAL)

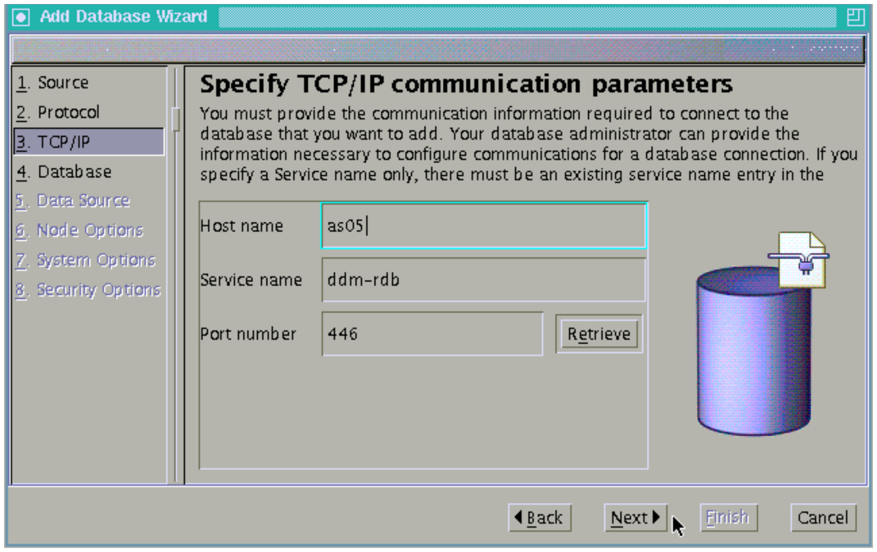
☐ The database physically resides on a host or OS/400 system.

[✓ Connect directly to the server]

Connect to the server via the gateway

Figure 3-11 Selection of the communication protocol

Select **TCP/IP** and click **Next** (Figure 3-11). Do not select the **The database physically resident on a host or OS/400 system**. This option is enabled when DB2 Connect is installed and TCP/IP or APPC is selected as the protocol. By checking the box, the connection will be managed in the Database Connection Service (DCS) directory and DB2 Connect can act as a gateway. It also works in our environment, but we do not use the DCS directory in our scenario.



Specify TCP/IP communication parameters

You must provide the communication information required to connect to the database that you want to add. Your database administrator can provide the information necessary to configure communications for a database connection. If you specify a Service name only, there must be an existing service name entry in the

Host name: as05

Service name: ddm-rdb

Port number: 446

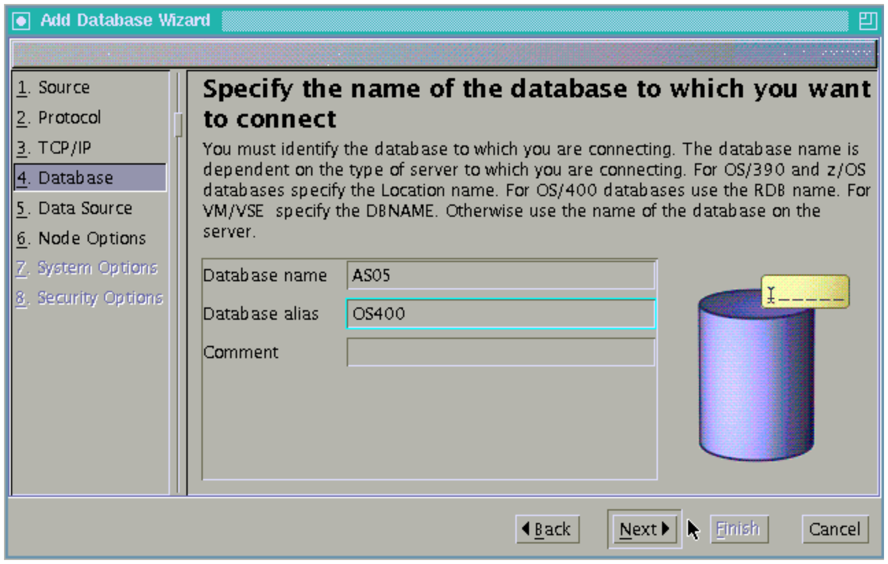
Figure 3-12 Specify the Host name and the port number

Type the host name of the OS/400 partition in the **Hostname** field (Figure 3-12). The host name must be resolved in the `/etc/hosts` file. Then type **`ddm-rdb`** in the Service name field and click **Retrieve** to get the port number 446, which is the port number that the OS/400 DDM server is working on.

Or you can specify only the IP address of the OS/400 and the port number, as follows:

- Host name: 192.168.1.1
- Service name: (blank)
- Port number: 446

Then click **Next**.



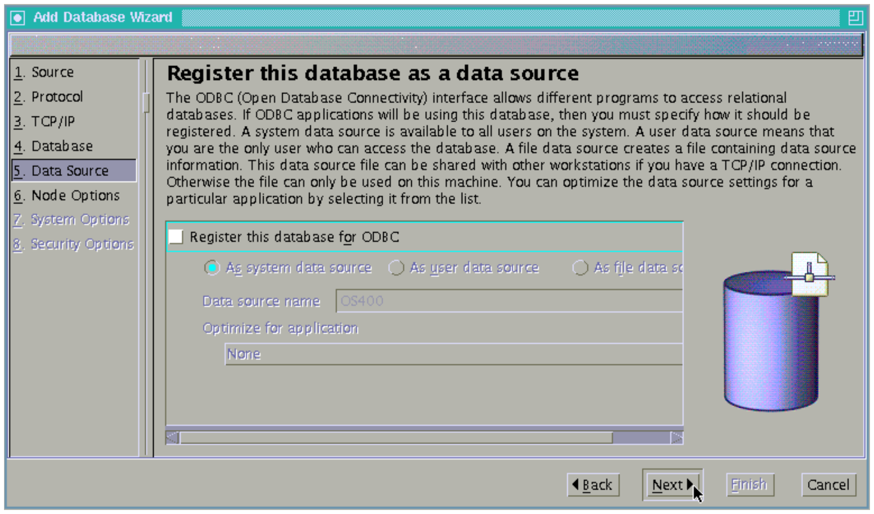
Specify the name of the database to which you want to connect

You must identify the database to which you are connecting. The database name is dependent on the type of server to which you are connecting. For OS/390 and z/OS databases specify the Location name. For OS/400 databases use the RDB name. For VM/VSE specify the DBNAME. Otherwise use the name of the database on the server.

Database name	AS05
Database alias	OS400
Comment	

Figure 3-13 Set the database name of OS/400 partition

Type the database name of the OS/400 partition and the alias (Figure 3-13). The database name must be the same as in the OS/400 RDB directory. The alias is the name you can choose here. Now we set it to `OS400`. Then click **Next**.



Add Database Wizard

1. Source
2. Protocol
3. TCP/IP
4. Database
5. Data Source
6. Node Options
7. System Options
8. Security Options

Register this database as a data source

The ODBC (Open Database Connectivity) interface allows different programs to access relational databases. If ODBC applications will be using this database, then you must specify how it should be registered. A system data source is available to all users on the system. A user data source means that you are the only user who can access the database. A file data source creates a file containing data source information. This data source file can be shared with other workstations if you have a TCP/IP connection. Otherwise the file can only be used on this machine. You can optimize the data source settings for a particular application by selecting it from the list.

Register this database for ODBC

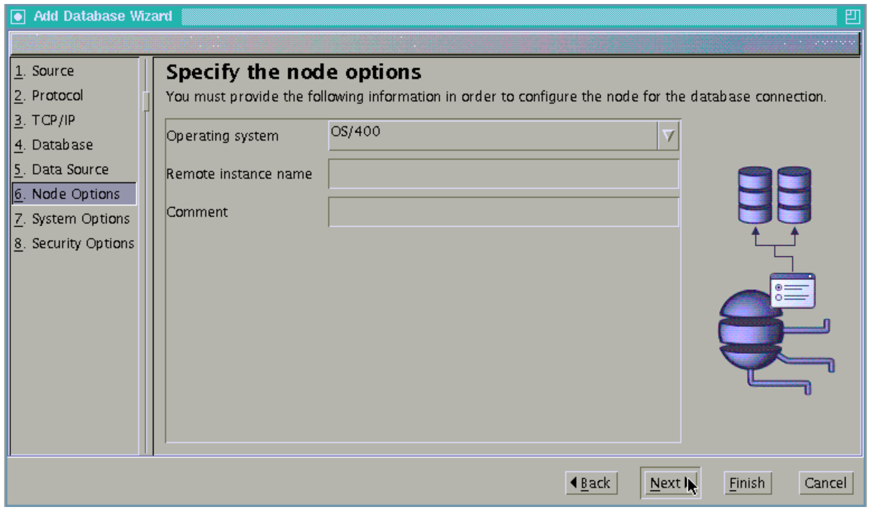
[● As system data source] As user data source As file data sc

Data source name: OS400

Optimize for application: None

Figure 3-14 ODBC configuration

Figure 3-14 shows the window used for **Register this database for ODBC**. Since this ODBC configuration option is not supported on the Linux platform yet, do not select it. Click **Next**.



Add Database Wizard

1. Source
2. Protocol
3. TCP/IP
4. Database
5. Data Source
6. Node Options
7. System Options
8. Security Options

Specify the node options

You must provide the following information in order to configure the node for the database connection.

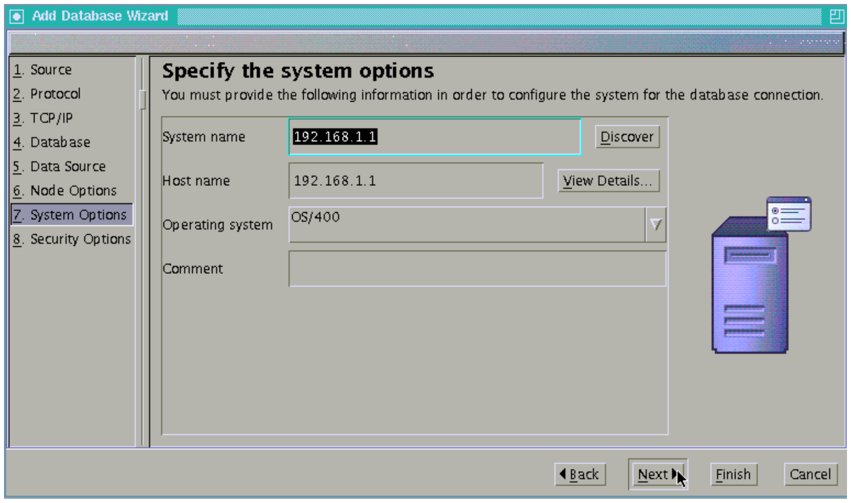
Operating system: OS/400

Remote instance name:

Comment:

Figure 3-15 Specify the Operating System type

Select the operating system OS/400 from the pull down list (Figure 3-15). Leave the remote instance name blank. Click **Next**.



Specify the system options

You must provide the following information in order to configure the system for the database connection.

System name: 192.168.1.1

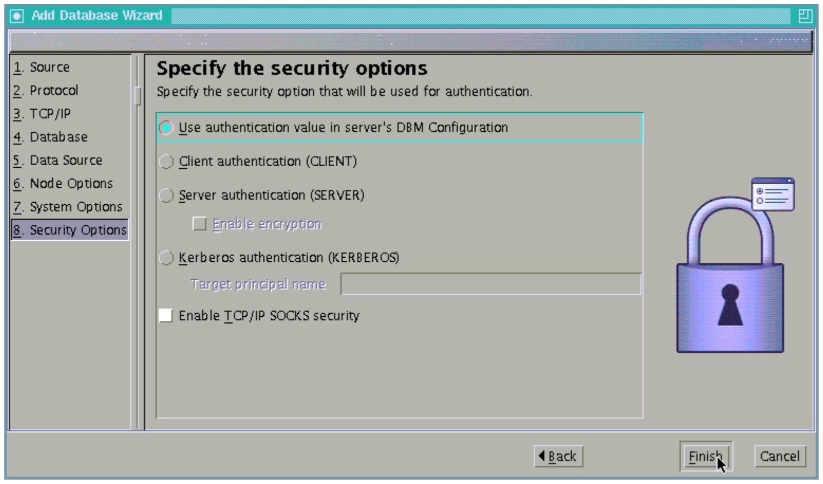
Host name: 192.168.1.1

Operating system: OS/400

Comment:

Figure 3-16 Specify the system options

Figure 3-16 appears. Check the configuration and click **Next**.



Specify the security options

Specify the security option that will be used for authentication.

(●) Use authentication value in server's DBM Configuration

( ) Client authentication (CLIENT)

( ) Server authentication (SERVER)

[ ] Enable encryption

( ) Kerberos authentication (KERBEROS)

Target principal name

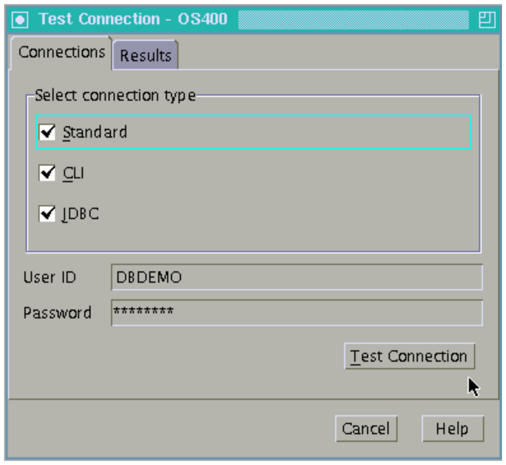
[ ] Enable TCP/IP SOCKS security

Figure 3-17 Specify the security options

As in Figure 3-17, you have to specify the authentication method of the user for this database. You can select the **Use authentication value in server's DBM configuration** or the **Server authentication (SERVER)**. Since the instance db2inst1 DBM is also configured as "SERVER" by default, they are the same as the result. SERVER authentication means that the user profile and the password which are used to connect to the OS/400 are checked as a normal user profile of the OS/400 partition in the OS/400 side. To check the instance DBM parameter, you can use the following command:

```
>db2 get database manager configuration|grep AUTHENTICATION
```

Click **Finish**.

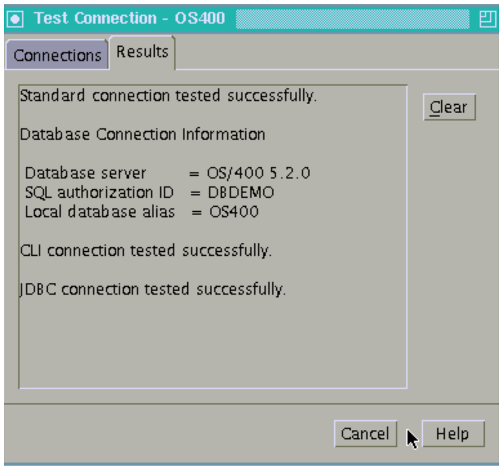


User ID DBDEMO

Password ********

Figure 3-18 The test tool window

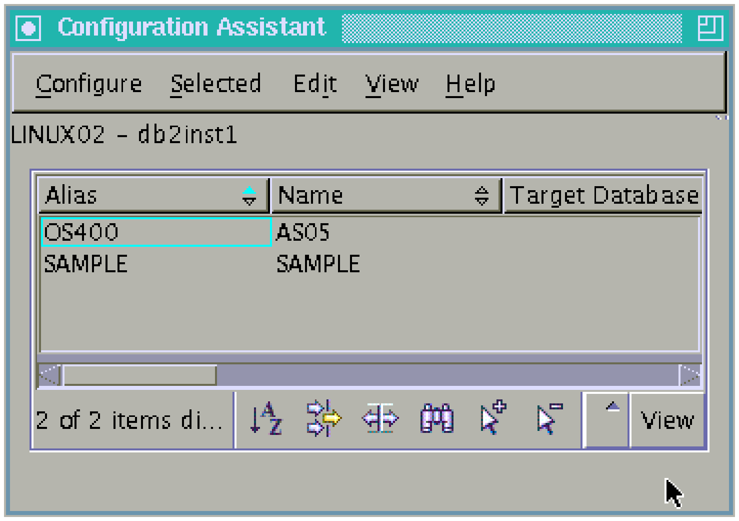
The Test Connection window opens (Figure 3-18). Select all connection types, and type the valid OS/400 user profile name and the password. Then click **Test Connection**.



```
Standard connection tested successfully.

Database Connection Information

 Database server       = OS/400 5.2.0
 SQL authorization ID  = DBDEMO
 Local database alias  = OS400

CLI connection tested successfully.

JDBC connection tested successfully.
```

Figure 3-19 The test result

If the test succeeded, a window as in Figure 3-19 appears. Click **Cancel**.



LINUX02 - db2inst1

Alias	Name	Target Database
OS400	AS05	
SAMPLE	SAMPLE	

Figure 3-20 The db2ca main window after the database is added

Now the database has added successfully. You can check the node or database directory settings as in Figure 3-20. You can use the following commands to check the configuration (Figure 3-21):

```
> db2 list node directory
> db2 list database directory
```

```
db2inst1@linux02:~> db2 list node directory

 Node Directory

 Number of entries in the directory = 1

Node 1 entry:

 Node name                      = NDE8D6BB
 Comment                        =
 Directory entry type           = LOCAL
 Protocol                       = TCPIP
 Hostname                       = as05
 Service name                   = ddm-rdb

db2inst1@linux02:~> db2 list database directory

 System Database Directory

 Number of entries in the directory = 2

Database 1 entry:

 Database alias                       = OS400
 Database name                        = AS05
 Node name                            = NDE8D6BB
 Database release level               = a.00
 Comment                              =
 Directory entry type                 = Remote
 Catalog database partition number    = -1

Database 2 entry:

 Database alias                       = SAMPLE
 Database name                        = SAMPLE
 Local database directory             = /home/db2inst1
 Database release level               = a.00
 Comment                              =
 Directory entry type                 = Indirect
 Catalog database partition number    = 0

db2inst1@linux02:~>
```

Figure 3-21 Checking the node and database directory

There are two databases registered in the instance. The second entry SAMPLE in the database directory is the sample database we created in Chapter 2, “Installation and configuration” on page 7.

Note: Because several of the utilities supplied with DB2 Connect are developed using embedded SQL, they must be bound to an iSeries database server before they can be used with that system. If you do not use the DB2 Connect utilities and interfaces, you do not have to bind them to each of your iSeries database servers. The name of the bind file is `ddcs400.lst` which is provided with DB2 Connect product. For further information, see the manual:

- *IBM DB2 Connect User’s Guide*, SC09-4835

3.4 Introduction to the scenarios

DB2 provides the various programming interfaces to an application program such as DB2 Call Level Interface (CLI), ODBC, Embedded SQL, JDBC or SQLJ. Any of those methods can be used to access DB2 UDB for iSeries.

CLI is the native programming interface of the DB2 family. ODBC is the Microsoft version of the CLI and they are basically the same programming interface. You can use ODBC by adding a little configuration to DB2. Embedded SQL is also the native programming interface which enables the mainframe-like programming style. JDBC is the common programming interface to access from a Java program. DB2 provides 3 types of JDBC driver, Type 2, Type 3, and Type 4. You can use any of these JDBC driver to access DB2 UDB for iSeries. SQLJ is the Java version of the Embedded SQL.

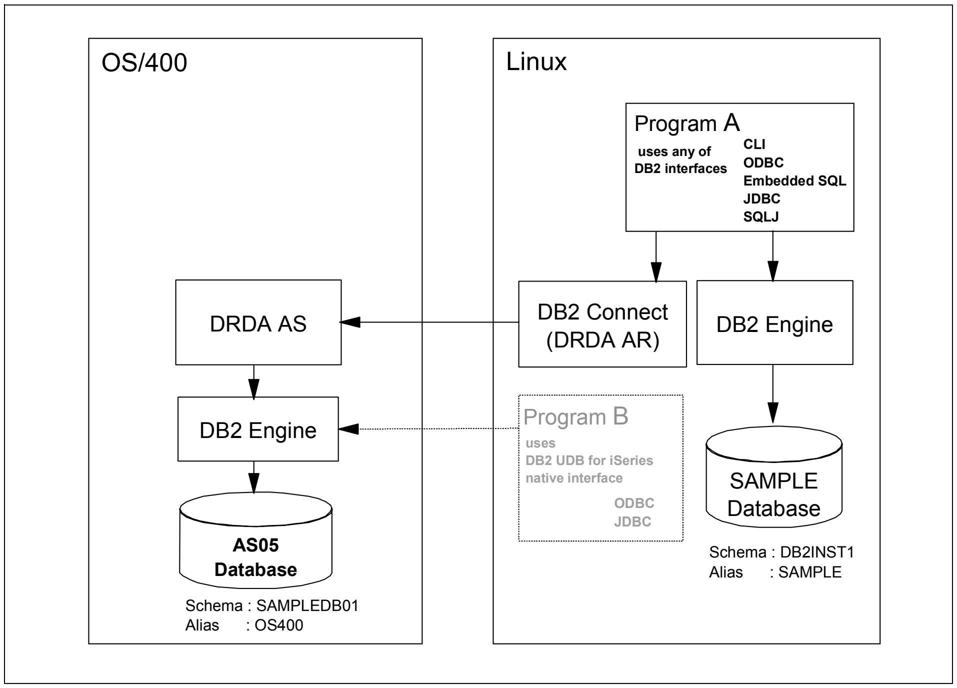


Figure 3-22 Connection methods summary

Figure 3-22 shows the conceptual view of the access methods provided by DB2. Note that the only programming interfaces which are facilitated via DB2 Connect are discussed in this book. They are represented as Program A in the chart.

There are some other native programming interfaces to access DB2 UDB for iSeries from the Linux platform, but are not part of the DB2 UDB for Linux and not discussed in this book. These are:

- iSeries ODBC Driver for Linux (5733-L01)
- IBM Toolbox for Java Type 4 JDBC driver

The program which uses either of these two connection methods is shown as Program B in the chart. This program can access DB2 UDB for iSeries directly but cannot access the Linux local database. If the database access requirement can be restricted only to DB2 UDB for iSeries, these two are good selections.

3.4.1 Create the sample database in OS/400

In this section, we create the DB2 sample database in the OS/400 partition using DB2 Command Line Processor (CLP). CLP is the basic command line interface to operate DB2 databases interactively. It is like the OS/400 STRSQL interface.

The sample database we create here is used by the sample programs of the following sections. Figure 3-23 shows the CLP dialog.

```
db2inst1@linux02:~> db2
(c) Copyright IBM Corporation 1993,2002
Command Line Processor for DB2 SDK 8.1.2

You can issue database manager commands and SQL statements from the command
prompt. For example:
    db2 => connect to sample
    db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
 ? CATALOG DATABASE for help on the CATALOG DATABASE command
 ? CATALOG          for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => connect to OS400 user DBDEMO using PASSWORD

   Database Connection Information

 Database server        = OS/400 5.2.0
 SQL authorization ID   = DBDEMO
 Local database alias   = OS400

db2 => call qsys.create_sql_sample('SAMPLEDB01')

  Return Status = 0

db2 => select * from sampledb01.staff where ID > 200

ID     NAME      DEPT   JOB   YEARS  SALARY    COMM
------ --------- ------ ----- ------ --------- ---------
   210 Lu            10 Mgr       10  20010.00         -
   220 Smith         51 Sales      7  17654.50    992.80
   230 Lundquist     51 Clerk      3  13369.80    189.65
   240 Daniels       10 Mgr        5  19260.25         -
   250 Wheeler       51 Clerk      6  14460.00    513.30
   260 Jones         10 Mgr       12  21234.00         -
   270 Lea           66 Mgr        9  18555.50         -
   280 Wilson        66 Sales      9  18674.50    811.50
   290 Quill         84 Mgr       10  19818.00         -
   300 Davis         84 Sales      5  15454.50    806.10
   310 Graham        66 Sales     13  21000.00    200.30
   320 Gonzales      66 Sales      4  16858.20    844.00
   330 Burke         66 Clerk      1  10988.00     55.50
   340 Edwards       84 Sales      7  17844.00   1285.00
   350 Gafney        84 Clerk      5  13030.50    188.00

  15 record(s) selected.

db2 => terminate
DB20000I  The TERMINATE command completed successfully.
db2inst1@linux02:~>
```

Figure 3-23   Create DB2 sample database in OS/400 partition

All DB2 database objects and DB2 commands are actually case insensitive. In the following sections, we use lower case for the DB2 command and upper case for the database objects. Note that the password of a user is case sensitive in DB2 UDB for Linux.

- To get into the CLP, just type **db2** from a command line of an appropriate db2 user. Then type the following command to connect to the database OS400, which is the alias of the AS05 database:

  ```
  db2 => connect to OS400 user DBDEMO using PASSWORD
  ```

  DBDEMO is the OS/400 user profile and PASSWORD is the password.

- In the following command, `qsys.create_sql_sample` is the stored procedure to create a sample database. The V5R1 or higher version of OS/400 supports the stored procedure. The contents of the sample database are the same as we created in the Linux partition. SAMPLEDB01 is the schema name of the sample database. The OS/400 library SAMPLEDB01 is created:

  ```
  db2 => call qsys.create_sql_sample('SAMPLEDB01')
  ```

  Make sure that SAMPLEDB01 is in uppercase or you will get an error.

- This is the statement to test the database. This SQL statement is the same as the Linux sample database we tested in Chapter 2. The only difference is that we have to explicitly specify the schema name in OS/400. If we do not specify the schema name, the default schema DBDEMO, same name as the user, is assumed and you will get an error.

  ```
  db2 => select * from SAMPLEDB01.STAFF where ID > 200
  ```

- Terminate command to exit from the CLP. It terminates the CLP background process. Though you can use **connect reset** and **quit** to exit from CLP, **terminate** completely ends the CLP, since **quit** does not terminate the background process.

  ```
  db2 => terminate
  ```

3.5 CLI scenario

DB2 Call Level Interface (DB2 CLI) is IBM’s callable SQL interface to the DB2 family of database servers. It is a “C“ and “C++“ application programming interface for relational database access that uses function calls to pass dynamic SQL statements as function arguments. DB2 CLI is based on the Microsoft Open Database Connectivity (ODBC) specification, and the International Standard for SQL/CLI. In addition, some DB2 specific extensions have been added to help the application programmer specifically exploit DB2 features.

Example 3-1 on page 59 (`testCLI.c`) is the simple C program example of the DB2 CLI. This program takes the name of a DB2 sample database, schema name, user and password as the parameters, then accesses the STAFF table, and then print the first six columns to the screen. The SQL statement is hard coded for simplicity and the only rows where the ID column is greater than 280 are printed to save space.

3.5.1 CLI programming

This section provides the analysis of the sample program.

```
#include "sqlcli.h" 1
```

This header file is necessary for CLI. It defines DB2 CLI constants such as SQLINTEGER, data structures, and function prototypes of CLI APIs.

```
strcat(querystring, "SELECT ID, NAME, DEPT, JOB, YEARS, SALARY FROM "); 2
strcat(querystring, argv[2]);
strcat(querystring, ".STAFF WHERE ID > 280");
```

This is the SQL statement to be executed. This program prints the first six columns of the STAFF table to the screen. `argv[2]` is the schema (OS/400 library) name. This program takes the schema name as a parameter to make it available to access both DB2 UDB for iSeries and for Linux by the single program.

The functions such as `SQLAllocEnv()` are the CLI APIs. `SQLConnect()` establishes the connection with the database. `SQLExecDirect()` runs the actual query. `SQLFetch()` gets the result of the query.

Note: The explanation of the CLI APIs is out of the scope of this book. For details, refer to the DB2 manuals:

- *Application Development Guide: Programming Client Applications,* SC09-4826
- *Call Level Interface Guide and Reference, Volumes 1 and 2,* SC09-4849 and SC09-4850

Example 3-1 A sample program of CLI (testCLI.c)

```
/*
Parameters :
              database name   (The alias of the database registered in DB2 directory for CLI
                               The datasource name for ODBC)
              schema          (The schema name of the database object.
                               Linux user name or OS/400 library name)
              validUserID     (The user ID or the user profile of the database.)
              password        (The password)

   Example:   testCLI mydb2 myschema myself mypassword
*/

#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>
#include <sqlcli.h>        /* For CLI  */ 1
/* #include <sql.h> */     /* For ODBC */

void diehappy(char *);
void die(char *);
void die2(char *, SQLRETURN);

/* The main routine */
int main(int argc, char *argv[]) {

#define MAX_CONNECTIONS 5

 /*-------------------------------*/
 /* Preparation                   */
 /*-------------------------------*/
 /* The buffers which holds the data from the table */
 char field1[100];
 char field2[100];
 char field3[100];
 char field4[100];
 char field5[100];
 char field6[100];

 SQLINTEGER retfld1;
 SQLINTEGER retfld2;
 SQLINTEGER retfld3;
 SQLINTEGER retfld4;
 SQLINTEGER retfld5;
 SQLINTEGER retfld6;

 SQLHENV henv;
 SQLHDBC hdbc[MAX_CONNECTIONS];
 SQLHSTMT statement;
 SQLRETURN sqe=0;
SQLRETURN res=0;

 /* Allocate the environment handle */
 if ( SQLAllocEnv(&henv)) die("Alloc Environment");

 /* Allocate the Connection handle */
 if ((sqe=SQLAllocConnect(henv,&hdbc[0]))) die2("Allocate Connect",sqe);

 /* Get the database handle */
 if ( (sqe= SQLConnect(hdbc[0],argv[1],SQL_NTS, argv[3],SQL_NTS, argv[4],SQL_NTS)))
die2("Connect",sqe);

 /* Allocate the statement handle */
 if (SQLAllocStmt(hdbc[0],&statement)) die("Alloc stmt");

 /*-------------------------------*/
 /* The core process              */
 /*-------------------------------*/
 /* Execute the SQL */
 char querystring[200];
 querystring[0] = 0x00;
 strcat(querystring, "SELECT ID, NAME, DEPT, JOB, YEARS, SALARY FROM "); 2
 strcat(querystring, argv[2]);
 strcat(querystring, ".STAFF WHERE ID > 280");
 printf("Running SQL : %s\n", querystring);
 if ((sqe=SQLExecDirect(statement,querystring, SQL_NTS))) die2("exec direct",sqe);

 /* Bind the buffers to the data to be fetched */
 if ((sqe=SQLBindCol(statement, 1 ,SQL_CHAR, field1,100, &retfld1))) die2("Bind field 1", sqe);
 if ((sqe=SQLBindCol(statement, 2, SQL_CHAR, field2,100, &retfld2))) die2("Bind field 2", sqe);
 if ((sqe=SQLBindCol(statement, 3, SQL_CHAR, field3,100, &retfld3))) die2("Bind field 3", sqe);
 if ((sqe=SQLBindCol(statement, 4, SQL_CHAR, field4,100, &retfld4))) die2("Bind field 4", sqe);
 if ((sqe=SQLBindCol(statement, 5, SQL_CHAR, field5,100, &retfld5))) die2("Bind field 5", sqe);
 if ((sqe=SQLBindCol(statement, 6, SQL_CHAR, field6,100, &retfld6))) die2("Bind field 6", sqe);

 /* The loop to fetch the all selected data */
 res = SQLFetch(statement);
 while (SQL_SUCCEEDED(res)) {

   /* Print the data with tab separated */
   printf("%s\t%s\t%s\t%s\t%s\t%s\n",field1,field2,field3,field4,field5,field6);
   res= SQLFetch(statement);  /* Next record */

 }

/*-------------------------------*/
 /* Clean up                      */
 /*-------------------------------*/
 /* Deallocate the statement handle */
 if (SQLFreeStmt(statement,SQL_DROP)) die("Free Statement with Drop");

 /* Deallocate the connection handle */
 if ((sqe=SQLDisconnect(hdbc[0] ))) die2("Disconnect ",sqe);
 if ((sqe=SQLFreeConnect(hdbc[0]))) die2("Free Connection",sqe);

 /* Deallocate the environment handle */
 if (SQLFreeEnv(henv)) die("Free Environment");

 diehappy("Got to end");

 return 0;
}

/* dieXXX() are the functions to print the ending status */

void diehappy(char *reason) {
  printf("%s\n",reason);
}

void die(char *reason) {
  printf("Failure because: %s\n",reason);
  exit(1);
}

void die2(char *reason,SQLRETURN code) {
  printf("Error code: %d\n",code);
  die(reason);
}
```

3.5.2 Compile and run

Figure 3-24 shows the compile and execution of the program.

The following command compiles and links the `testCLI.c` using the GNU C compiler with the include files and with the DB2 libraries.

```
> gcc -o testCLI -I/home/db2inst1/sqllib/include -L/home/db2inst1/sqllib/lib
-ldb2 testCLI.c
```

`-o` option specifies the name of the executable module created.

`-I` option specifies the include directory. The C include files and the header files are located in the `sqllib/include` directory of the instance directory. Actually, the directory is the symbolic link to the DB2 product directory.

`-l` option specifies the library to be linked to this program. When `-ldb2` is specified, the library file named `libdb2.so` is searched and linked.

`-L` option specifies the directory where the library file is. The library file of DB2 are located in the `sqllib/lib` directory of the instance directory. Actually, the directory is the symbolic link to the DB2 product directory.

You can also use the **`bldapp`** script provided by DB2 product to compile the program, which is located in the `/home/db2inst1/sqllib/samples/cli/` directory. It is a short script and you can modify it if needed.

To run the program to the AS05 database, type the following command. The database alias of the OS/400 DB2 `OS400` is specified as well as in the CLP sample.

```
> ./testCLI OS400 SAMPLEDB01 DBDEMO password
```

This program can also access the Linux local DB2 database. The only difference is the schema name. To run it to the local SAMPLE database, type the following command:

```
> ./testCLI SAMPLE DB2INST1 DB2INST1 password
```

```
db2inst1@linux02:~/redbook> gcc -o testCLI -I /home/db2inst1/sqllib/include -L
/home/db2inst1/sqllib/lib -ldb2 testCLI.c
db2inst1@linux02:~/redbook> ./testCLI OS400 SAMPLEDB01 DBDEMO password
Running SQL : SELECT ID, NAME, DEPT, JOB, YEARS, SALARY FROM SAMPLEDB01.STAFF WHERE
ID > 280
290     Quill   84      Mgr     10      19818.00        84
300     Davis   84      Sales   5       15454.50        84
310     Graham  66      Sales   13      21000.00        66
320     Gonzales        66      Sales   4       16858.20        66
330     Burke   66      Clerk   1       10988.00        66
340     Edwards 84      Sales   7       17844.00        84
350     Gafney  84      Clerk   5       13030.50        84
Got to end
db2inst1@linux02:~/redbook> ./testCLI SAMPLE DB2INST1 DB2INST1 password
Running SQL : SELECT ID, NAME, DEPT, JOB, YEARS, SALARY FROM DB2INST1.STAFF WHERE ID
> 280
290     Quill   84      Mgr     10      19818.00        84
300     Davis   84      Sales   5       15454.50        84
310     Graham  66      Sales   13      21000.00        66
320     Gonzales        66      Sales   4       16858.20        66
330     Burke   66      Clerk   1       10988.00        66
340     Edwards 84      Sales   7       17844.00        84
350     Gafney  84      Clerk   5       13030.50        84
Got to end
```

Figure 3-24 Compile and run the testCLI.c program

3.6 ODBC scenario

Practically, ODBC and CLI are the same programming interface. DB2 UDB CLI provides full Level 1 (ODBC) support, plus many Level 2 functions. For the most part, ODBC is a super set of the ANSI and ISO SQL CLI standard. The DB2 CLI driver also acts as an ODBC driver when loaded by an ODBC driver manager. It conforms to ODBC 3.51. In a basic level, the CLI program can also run as an ODBC program by recompiling it. The name of APIs or the parameters of the function of CLI and ODBC are common. For further discussion of ODBC and CLI, refer to:

- *Call Level Interface Guide and Reference*, SC09-4849

As illustrated in Figure 3-25, there are two major components of the ODBC system, the ODBC driver manager and the ODBC driver.

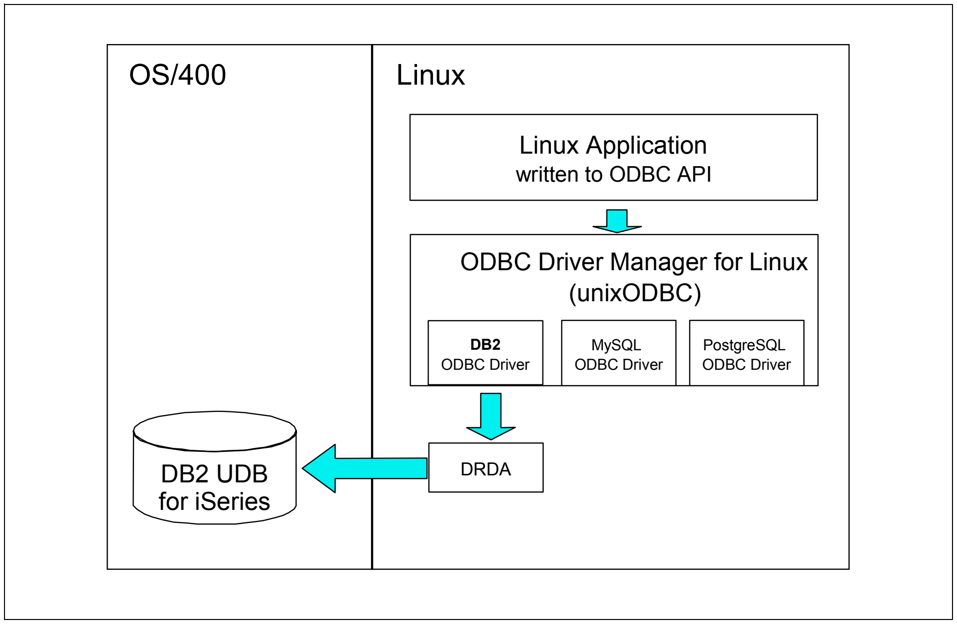


Figure 3-25 ODBC driver and ODBC driver manager

The ODBC driver manager is a management facility which mediates the application program and each ODBC driver. The ODBC driver is the RDBMS dependent module to enable access to each database via ODBC.

On the Windows platform, the ODBC driver manager is provided as part of the operating system, but on Linux there is no system-provided ODBC driver manager. Although there are a number of open source or commercial ODBC driver managers on the Linux platform, the open source unixODBC (`http://www.unixODBC.org/`) is the formally supported ODBC driver manager of DB2 UDB and DB2 Connect on Linux.

The ODBC driver to access DB2 is provided as a part of DB2 UDB or DB2 Connect product. Actually, the CLI library `libdb2.so` itself can be used as the ODBC driver for DB2.

3.6.1 ODBC setup

First, we have to setup the ODBC environment, such as the ODBC driver manager and data sources. SLES8 includes unixODBC-2.2.2-57 package and unixODBC-devel-2.2.2-57 package. We use the unixODBC-2.2.2-57 as the driver manager and unixODBC-devel-2.2.2-57 as the development package. The *devel* package is necessary to compile the ODBC application program.

The unixODBC usually has some other GUI applications, such as ODBCConfig (Configuration tool) or Datamanager (Test tool) but they are not included in SLES8. If you want to use these, you need to download the additional (or the full) packages and install (or make) them. For unixODBC, refer to the Web site:

http://www.unixodbc.org

In this section, we directly update the ODBC configuration files by a text editor.

ODBC Driver Manager

In the default installation of SLES8, the unixODBC driver manager configuration file is the text file `/etc/unixODBC/odbcinst.ini`, which is an empty file. We need to add the lines for the ODBC driver that comes with DB2. In this section, we refer to that driver as the DB2 ODBC Driver.

To configure the ODBC driver as the root user, using a text editor, add a new stanza as illustrated in Figure 3-26.

```
[DB2DRIVER]
Description = DB2 ODBC Driver
Driver      = /home/db2inst1/sqllib/lib/libdb2.so
```

Figure 3-26 Driver manager configuration

The library file `libdb2.so` is the DB2 ODBC Driver installed with the DB2 product. Actually it is the file we used as the CLI library in 3.5.2, “Compile and run” on page 61. The DB2 CLI library also can be used as the DB2 ODBC driver.

We named the driver “DB2DRIVER” which is the name of the stanza. You can choose any name here, but the name must be used in the datasource configuration file.

ODBC data sources

The data source configuration file is `/etc/unixODBC/odbc.ini`, for the system data sources, and `~/.odbc.ini` for the user data sources (~ is the user’s home directory). We use the system data source in our scenario.

To add two databases, AS05 on the OS/400 and SAMPLE on the Linux as the ODBC system data source, edit the `/etc/unixODBC/odbc.ini` as the root user.

In our example (Figure 3-27), we add two stanzas: `[OS400]` for the remote OS/400 database and `[SAMPLE]` for the local Linux database. These names must be the same as the database alias registered in the DB2 database directory. The driver name `DB2DRIVER` must be the same as the stanza name in the odbcinst.ini file in Figure 3-26.

```
[OS400]
Description = OS/400 Host partition AS05
Driver      = DB2DRIVER

[SAMPLE]
Description = DB2 local sample database
Driver      = DB2DRIVER
```

Figure 3-27 System data sources configuration

That is all you need to do for the setup. As illustrated in the above example, configuration is quite simple because all the connection information is held in the DB2 node or database directory. You just need a few stanzas created in both the odbcinst.ini and odbc.ini files for the entry of DB2 Connect databases.

Now you are ready to test the ODBC connection over the DB2 ODBC Driver. You can use “isql” to test the ODBC connection.

Note: Be reminded that there are two common ODBC drivers for accessing DB2 UDB for iSeries. One is DB2 ODBC Driver, which we have mentioned so far, and comes with DB2 product; and the other is iSeries ODBC Driver for Linux (5733-L01), which you can download from the following Web site:

http://www-1.ibm.com/servers/eserver/iseries/linux/odbc/

The iSeries ODBC Driver for Linux is the native ODBC driver that only supports database access to DB2 UDB for iSeries. As a reference, we provide some key elements of the configuration for this driver:

On odbcinst.ini file:

```
[iSeries Access ODBC Driver]
Description = iSeries Access for Linux ODBC Driver
Driver          = /opt/ibm/iSeriesODBC/lib/libcwbodbc.so
Setup           = /opt/ibm/iSeriesODBC/lib/libcwbodbc.so
FileUsage               = 30
```

On odbc.ini file:

```
[OS400NATIVE]
Description             = iSeries Access ODBC Driver
Driver          = iSeries Access ODBC Driver
System          = 192.168.1.1
UserID          =
Password                =
```

Note that the iSeries ODBC Driver for Linux is a completely separate driver to DB2 ODBC Driver, which comes with DB2 Connect, and it does not use DRDA at all. Though, both ODBC drivers can coexist because the ODBC driver configuration files, such as odbcinst.ini and odbc.ini files, can have entries for both drivers. At run time, it's just up to the individual application to pick a particular data source, which then uses selected ODBC driver.

3.6.2 isql

isql is the command line tool like db2 CLP accompanied with the unixODBC package. Figure 3-28 shows the operation of isql. After typing the ODBC data source name, user, and password from a command prompt of user db2inst1, you can issue SQL query to the data source.

```
db2inst1@linux02:~/redbook> isql OS400 DBDEMO password
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL> select * from SAMPLEDB01.STAFF where ID > 280
+-------+----------+-------+------+-------+----------+----------+
| ID    | NAME     | DEPT  | JOB  | YEARS | SALARY   | COMM     |
+-------+----------+-------+------+-------+----------+----------+
| 290   | Quill    | 84    | Mgr  | 10    | 19818.00 |          |
| 300   | Davis    | 84    | Sales| 5     | 15454.50 | 806.10   |
| 310   | Graham   | 66    | Sales| 13    | 21000.00 | 200.30   |
| 320   | Gonzales | 66    | Sales| 4     | 16858.20 | 844.00   |
| 330   | Burke    | 66    | Clerk| 1     | 10988.00 | 55.50    |
| 340   | Edwards  | 84    | Sales| 7     | 17844.00 | 1285.00  |
| 350   | Gafney   | 84    | Clerk| 5     | 13030.50 | 188.00   |
+-------+----------+-------+------+-------+----------+----------+
7 rows returned
SQL> quit
db2inst1@linux02:~/redbook>
```

Figure 3-28 isql operation sample

3.6.3 ODBC programming

The sample program we use to test ODBC is testCLI.c which is the same program used in the CLI section (Example 3-1 on page 59). The only line you have to change is:

```
#include "sqlcli.h" 1
```

It must be changed to:

```
#include <sql.h>
```

Where, the `/usr/include/sql.h` is the file installed as part of the unixODBC-devel package. No other change is needed to compile the CLI program as the ODBC program. The APIs and the parameters are the same.

3.6.4 Compile and run

Use the following command to create the executable program whose name is `testODBC`.

```
gcc -lodbc -o testODBC testCLI.c
```

The library file to be linked to ODBC program is `/usr/lib/libodbc.so`. Since the directory `/usr/lib` is the system default library search path, you do not have to specify the `-L` option. Also, the include file sql.h is in the system default include path, and the `-I` option is not needed.

Figure 3-29 shows the compile and run result of the program. It is completely the same as in the CLI section. You can also connect to the local DB2 sample database by specifying the database name SAMPLE and the schema name DB2INST1.

```
db2inst1@linux02:~/redbook> gcc -lodbc -o testODBC testCLI.c
db2inst1@linux02:~/redbook> ./testODBC OS400 SAMPLEDB01 DBDEMO password
Running SQL : SELECT ID, NAME, DEPT, JOB, YEARS, SALARY FROM SAMPLEDB01.STAFF WHERE
ID > 280
290     Quill   84      Mgr     10      19818.00        84
300     Davis   84      Sales   5       15454.50        84
310     Graham  66      Sales   13      21000.00        66
320     Gonzales        66      Sales   4       16858.20        66
330     Burke   66      Clerk   1       10988.00        66
340     Edwards 84      Sales   7       17844.00        84
350     Gafney  84      Clerk   5       13030.50        84
Got to end
db2inst1@linux02:~/redbook> ./testODBC SAMPLE DB2INST1 DB2INST1 password
Running SQL : SELECT ID, NAME, DEPT, JOB, YEARS, SALARY FROM DB2INST1.STAFF WHERE ID
> 280
290     Quill   84      Mgr     10      19818.00        84
300     Davis   84      Sales   5       15454.50        84
310     Graham  66      Sales   13      21000.00        66
320     Gonzales        66      Sales   4       16858.20        66
330     Burke   66      Clerk   1       10988.00        66
340     Edwards 84      Sales   7       17844.00        84
350     Gafney  84      Clerk   5       13030.50        84
Got to end
db2inst1@linux02:~/redbook>
```

Figure 3-29 Compile and run the testCLI.c as an ODBC program

3.7 Embedded SQL scenario

Embedded SQL is a programming interface which enables a program to access database using static SQL. The static SQL means that the exact SQL statements issued by a program are defined at the coding time. The programmer *embeds* the SQL statement into the source code.

For CLI, ODBC, or JDBC, the technique is called *dynamic SQL*. Using the dynamic SQL, the SQL statements can be provided at run time. For example, in dynamic SQL, the SQL statements can be provided as program parameters, data field read from a file, or user input.

3.7.1 Embedded SQL programming

Example 3-2 shows the simple embedded SQL program `testESQL.sqc`. The program is written in C, but the extension is .sqc because an embedded SQL program must be precompiled by a DB2 tool to make `.c` file.

`testESQL.sqc` does the same work as `testCLI.c`. as shown in "CLI scenario" on page 58. It accesses the OS400 database and prints the six columns of the SAMPLEDB01.STAFF table to the screen.

The statements beginning with "`EXEC SQL`" are the embedded SQL statements, which are translated to C code including various DB2 function calls by the precompiler. For example, the single line is translated as illustrated in Example 3-3 on page 72 by the precompiler:

```
EXEC SQL CONNECT TO :dbname USER :user USING :password; 1
```

The variables such as `:dbname` or `:user` are called "host variables" and can be passed by the program at run time.

Example 3-2 Embedded SQL sample program (testESQL.sqc)

```
/*
  testESQL.sqc a simple sample program of DB2 Embedded SQL which access Linux or OS/400
DB2
              Parameters :
              databasename   (The alias of the database registered in DB2 directory)
              validUserID    (The user ID or the user profile of the database.)
              password       (The password)

   Example :   testESQL mydb2 myself mypassword
*/

#include <stdio.h>
#include <sql.h>
#include <sqlenv.h>
#include <sqlda.h>
#include <sqlca.h>
#include <string.h>

EXEC SQL INCLUDE SQLCA;

int main(int argc, char *argv[]) {

 /*---------------------------------*/
 /* Preparation                   */
 /*-------------------------------*/
 /* The host variables which hold the data from the table */
 EXEC SQL BEGIN DECLARE SECTION;
 char    dbname[10];
 char    user[10];
 char    password[10];

 double  field1;
 char    field2[100];
 double  field3;
 char    field4[100];
 double  field5;
 double  field6;
 EXEC SQL END DECLARE SECTION;

 printf("Embedded SQL sample\n");

 EXEC SQL WHENEVER SQLERROR GO TO error;

 strcpy(dbname, argv[1]);
 strcpy(user, argv[2]);
 strcpy(password, argv[3]);
 EXEC SQL CONNECT TO :dbname USER :user USING :password; 1

 printf("Connected.\n");

 EXEC SQL DECLARE C1 CURSOR FOR
      SELECT ID, NAME, DEPT, JOB, YEARS, SALARY FROM SAMPLEDB01.STAFF
      WHERE ID > 280 ;

 EXEC SQL OPEN C1;
 EXEC SQL FETCH C1 INTO :field1, :field2, :field3, :field4, :field5, :field6;
 while(SQLCODE == 0) {
   printf("%.0f\t%s\t%.0f\t%s\t%.0f\t%.0f\n",field1,field2,field3,field4,field5,field6);
   EXEC SQL FETCH C1 INTO :field1, :field2, :field3, :field4, :field5, :field6;
 }
 EXEC SQL CLOSE C1;

 printf("Got to end\n");

 EXEC SQL COMMIT;
 EXEC SQL CONNECT RESET;

 return 0;

 error:
 EXEC SQL WHENEVER SQLERROR CONTINUE;
 EXEC SQL ROLLBACK;
 EXEC SQL CONNECT RESET;

 exit(1);
}
```

Example 3-3 Embedded SQL code translated by the precompiler

```
/*
EXEC SQL CONNECT TO :dbname USER :user USING :password;
*/

{
  sqlastrt(sqla_program_id, &sqla_rtinfo, &sqlca);
  sqlaaloc(2,3,1,0L);
    {
      struct sqla_setdata_list sql_setdlist[3];
      sql_setdlist[0].sqltype = 460; sql_setdlist[0].sqllen = 10;
      sql_setdlist[0].sqldata = (void*)dbname;
      sql_setdlist[0].sqlind = 0L;
      sql_setdlist[1].sqltype = 460; sql_setdlist[1].sqllen = 10;
      sql_setdlist[1].sqldata = (void*)user;
      sql_setdlist[1].sqlind = 0L;
      sql_setdlist[2].sqltype = 460; sql_setdlist[2].sqllen = 10;
      sql_setdlist[2].sqldata = (void*)password;
      sql_setdlist[2].sqlind = 0L;
      sqlasetdata(2,0,3,sql_setdlist,NULL,0L);
    }
  sqlacall((unsigned short)29,5,2,0,0L);
  if (sqlca.sqlcode < 0)
  {
    sqlastop(0L);
    goto error;
  }
  sqlastop(0L);
}
```

3.7.2 Compile and run

Figure 3-30 shows the procedure to make a program use embedded SQL. Embedded SQL programs need to be precompiled by a DB2 tool (**db2 prep** command) before compiled by a C compiler. The extension of an embedded SQL program is `.sqc`. The precompiler first analyzes the `.sqc` file and converts the `EXEC SQL` statements, which shows the line is the DB2 embedded SQL command to the related C source codes, which includes DB2 internal APIs. The precompiler makes a C source file in which the extension is `.c`. We have to compile the `.c` file using a C compiler to make the application. At the precompile time, the precompiler checks the SQL statement with actually connecting the database and creates some objects called *SQL package* or *bind file*. The SQL package is an object stored in the database which has all the information needed to execute the SQL statements of the program. The SQL package is created and stored in the database at the precompilation time. Instead, you can create the bind file as a local file. Using the bind file, you can delay the creation of the SQL package until the file is *bound* to a database by using the **db2 bind** command.

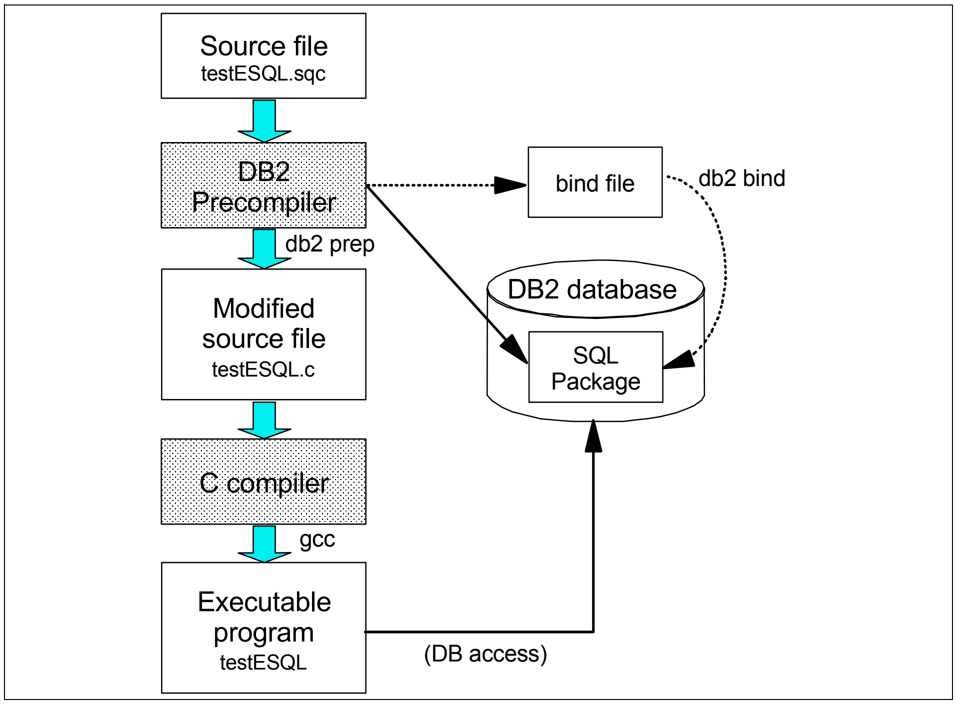


Figure 3-30 Embedded SQL program compiling process

We create the SQL package directly in this scenario and do not use bind file.

To precompile the `testESQL.sqc`, the DB2 CLP and **`db2 prep`** command is used. Since the connection to the database is necessary when precompiling the program, you have to connect to the OS400 database first. Then we can precompile. Figure 3-31 shows this.

```
db2inst1@linux02:~/redbook> db2 connect to OS400 user DBDEMO using PASSWORD

   Database Connection Information

 Database server        = OS/400 5.2.0
 SQL authorization ID   = DBDEMO
 Local database alias   = OS400

db2inst1@linux02:~/redbook> db2 prep testESQL.sqc collection SAMPLEDB01

LINE    MESSAGES FOR testESQL.sqc
------  --------------------------------------------------------------------
        SQL0060W  The "C" precompiler is in progress.
        SQL0091W  Precompilation or binding was ended with "0"
                  errors and "0" warnings.
db2inst1@linux02:~/redbook>db2 terminate
DB20000I  The TERMINATE command completed successfully.
db2inst1@linux02:~/redbook>
```

Figure 3-31 Precompile the embedded SQL program

The parameter `collection SAMPLEDB01` specifies the schema (OS/400 library) in which the SQL package is created. If the `collection` parameter is not specified, the default schema of the user (`DBDEMO`) is assumed.

The precompiler creates the file `testESQL.c`. We, then compile the file using the gcc compiler to create the final product of the process. The compile parameters are the same as for the CLI program. Figure 3-32 shows the compile and run steps.

```
db2inst1@linux02:~/redbook> gcc -o testESQL -I/home/db2inst1/sqllib/include
-L/home/db2inst1/sqllib/lib -ldb2 ./testESQL.c
db2inst1@linux02:~/redbook> ./testESQL OS400 DBDEMO PASSWORD
Embedded SQL sample
Connected.
290     Quill   84      Mgr     10      19818
300     Davis   84      Sales   5       15454
310     Graham  66      Sales   13      21000
320     Gonzales        66      Sales   4       16858
330     Burke   66      Clerk   1       10988
340     Edwards 84      Sales   7       17844
350     Gafney  84      Clerk   5       13030
Got to end
db2inst1@linux02:~/redbook>
```

Figure 3-32 Compile and run the embedded SQL program

You can get the same result from the local SAMPLE database. You have to change the schema name which is embedded in the source code to `DB2INST1`, precompile the `.sqc` file in the environment connected to the `SAMPLE` database, and compile the `.c` program. Figure 3-33 shows the result.

```
db2inst1@linux02:~/redbook> db2 connect to SAMPLE

   Database Connection Information

 Database server        = DB2/LINUXPPC 8.1.2
 SQL authorization ID   = DB2INST1
 Local database alias   = SAMPLE

db2inst1@linux02:~/redbook> db2 prep testESQL.sqc

LINE    MESSAGES FOR testESQL.sqc
------  --------------------------------------------------------------------
        SQL0060W  The "C" precompiler is in progress.
        SQL0091W  Precompilation or binding was ended with "0"
                  errors and "0" warnings.
db2inst1@linux02:~/redbook> db2 terminate
DB20000I  The TERMINATE command completed successfully.
db2inst1@linux02:~/redbook> gcc -o testESQL -I/home/db2inst1/sqllib/include
-L/home/db2inst1/sqllib/lib -ldb2 ./testESQL.c
db2inst1@linux02:~/redbook> ./testESQL SAMPLE DB2INST1 password
Embedded SQL sample
Connected.
290     Quill   84      Mgr     10      19818
300     Davis   84      Sales   5       15454
310     Graham  66      Sales   13      21000
320     Gonzales        66      Sales   4       16858
330     Burke   66      Clerk   1       10988
340     Edwards 84      Sales   7       17844
350     Gafney  84      Clerk   5       13030
Got to end
db2inst1@linux02:~/redbook>
```

Figure 3-33   Precompile, compile, and run to the local database

3.8 JDBC scenario

In this section, we discuss the use of JDBC in the DB2 Connect environment.

3.8.1 JDBC overview

Java Database Connectivity (JDBC) is a mandatory component of the Java programming language as defined in the Java 2, Standard Edition (J2SE) specification. DB2 UDB for Linux on iSeries includes support for JDBC, a vendor-neutral dynamic SQL interface that provides data access to your application through standardized Java methods.

JDBC is similar to DB2 CLI or DB2 ODBC in that you do not have to pre-compile the application code or bind packages to a DB2 database. As an open standard, JDBC applications offer increased portability — a required benefit in today's heterogeneous business infrastructure.

During the execution of a JDBC application the driver will validate SQL statements against the currently connected DB2 database server. Any problems during access will be reported to the application as a Java exception along with the corresponding DB2 SQLSTATE and SQLCODE.

JDBC Versions and JDK

DB2 UDB for Linux on iSeries V8.1.2 supports the JDBC V2.1 specification and portions of JDBC 3.0. Plans are underway to fully support the JDBC 3.0 requirements with the Universal JDBC driver. Each version of JDBC has an associated JDK/JRE.

The JDBC 2.0 and JDBC 2.1 APIs are divided into two packages:

- **java.sql package:** A package which includes the core APIs.
- **javax.sql package:** An optional package, which is used to support connection pooling, distributed transaction, and other advanced functions.

JDBC 2.0 and JDBC 2.1 core APIs are supported by J2SE (Java 2 Standard Edition, implemented in JDK 1.2 and above), and the JDBC 2.0 optional package is supported by J2EE 1.3.

JDBC 3.0 is incorporated in J2SE 1.4 and J2EE 1.4.

IBM JDK 1.3.1 is the currently recommended environment for Java applications. It is provided along with DB2 UDB for Linux on iSeries.

JDBC drivers and JDBC driver types

To enable JDBC applications for DB2, an implementation of the various Java classes and interfaces, as defined in the standard, are required. This implementation is known as a JDBC driver. According to the JDBC specification, there are four types of JDBC driver architectures:

- **Type 1** are drivers that implement the JDBC API as a mapping to another data access API, such as Open Database Connectivity (ODBC). Drivers of this type are generally dependent on a native library, which limits their portability. The JDBC-ODBC Bridge driver is an example of a Type 1 driver.
- **Type 2** are drivers that are written partly in the Java programming language and partly in native code. The drivers use a native client library specific to the data source to which they connect. Again, because of the native code, their portability is limited.
- **Type 3** are drivers that use a pure Java client and communicate with a middleware server using a database-independent protocol. The middleware server then communicates the client's requests to the data source.
- **Type 4** are drivers that are pure Java and implement the network protocol for a specific data source. The client connects directly to the data source.

3.8.2 JDBC drivers provided by DB2 UDB for Linux

DB2 UDB for Linux provides Type 2, Type 3, and Type 4 JDBC drivers. DB2 UDB for Linux Type 2 and Type 3 drivers use the DB2 CLI (Call Level Interface) interface to communicate to DB2 UDB servers. The JDBC Type 2 and Type 3 drivers are provided in the `db2java.zip` file located in the `sqllib/java` directory.

DB2 Version 8 adds a new DB2 JDBC Universal Driver (Type 4), which uses the Distributed Relational Database Architecture™ (DRDA) protocol for client/server communications. This new driver is provided in the `db2jcc.jar` file in the `sqllib/java` directory. In DB2 UDB 8.1.2, the DB2 JDBC Universal Driver also can be used as a Type 2 driver.

Type 2 driver (Application driver)

Figure 3-34 illustrates the mechanism in which the type 2 JDBC driver works. The client layer is added for explanation purposes, but the client facility can also reside in Linux.

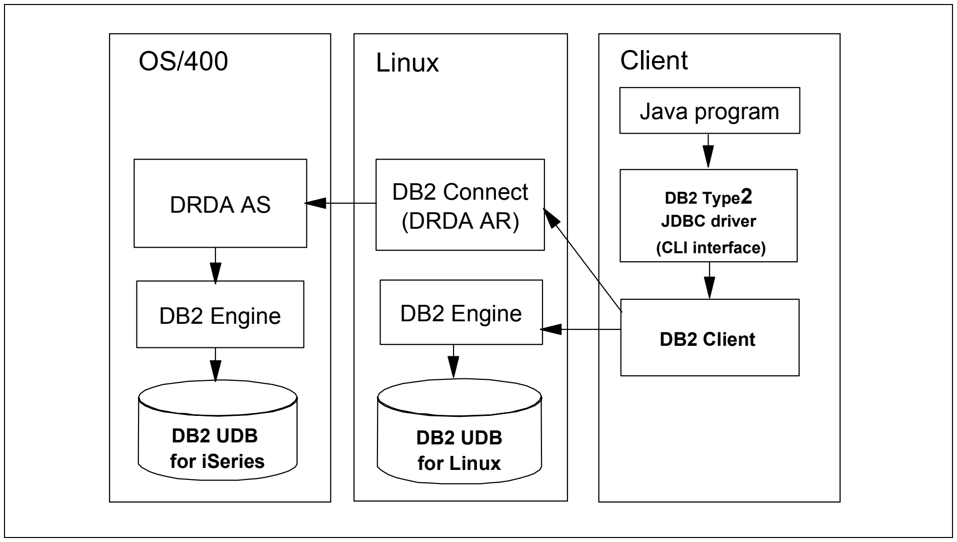


Figure 3-34 Type 2 JDBC driver provided by DB2 UDB for Linux

The type 2 driver, which is called "DB2 JDBC application driver" enables Java applications to make calls to DB2 through JDBC. The Java applications that use this driver must run on a DB2 Client, through which JDBC requests flow to the DB2. DB2 Client is the "native client library" needed for the type 2 driver in this case. When you use the type 2 driver in the Linux, you do not need DB2 Client, because the DB2 UDB for Linux server also has the client facility. If you want to access DB2 UDB for iSeries by this driver, DB2 Connect Version 8 license/installation is required in the Linux partition.

Note: In reality, you don't need DB2 Client product on the client machine as far as your client and Linux partition with DB2 UDB for LUW installed is the same machine (that is, in the same Linux LPAR partition). This is because it is integrated in every edition of DB2 for UDB for LUW available for iSeries.

Type 3 driver (Applet driver or net driver)

Figure 3-35 illustrates the mechanism in which the type 3 JDBC driver works. The client layer is added for explanation purposes but the client facility can also reside in Linux.

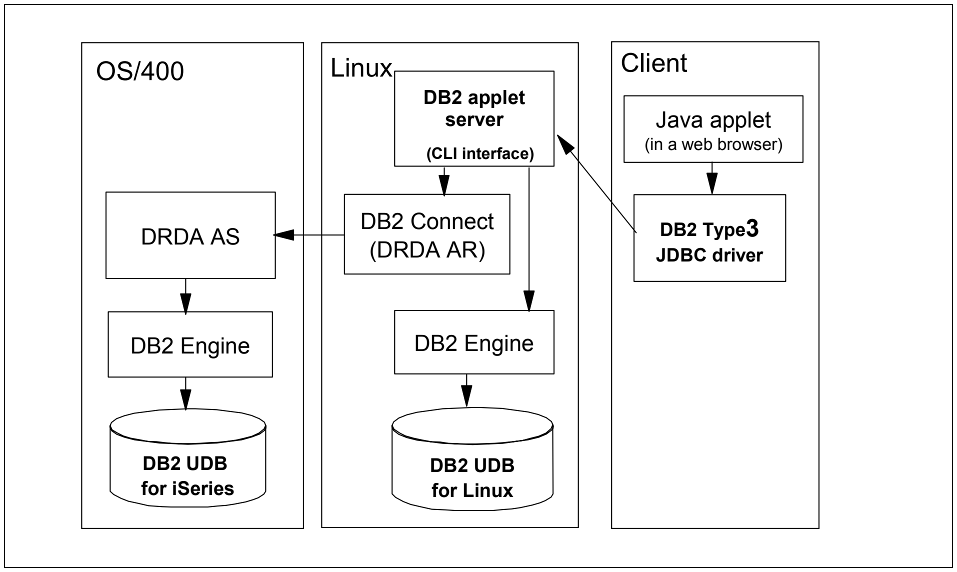


Figure 3-35 Type3 JDBC driver provided by DB2 UDB for Linux

The DB2 JDBC Type 3 driver, also known as the applet or net driver, consists of a JDBC client and a JDBC server. The DB2 JDBC applet driver can be loaded by the Web browser along with the applet or the applet driver can be used in standalone Java applications. When the applet requests a connection to a DB2 database, the applet driver opens a TCP/IP socket to the DB2 JDBC applet server on the machine where the Web server is running.

After a connection is set up, the applet driver sends each of the subsequent database access requests from the applet to the JDBC server through the TCP/IP connection. The JDBC server then makes corresponding DB2 calls to perform the task. Upon completion, the JDBC server sends the results back to the JDBC client through the connection. The JDBC server process is db2jd.

Type 4 driver (DB2 JDBC universal driver)

Figure 3-36 illustrates the mechanism in which the type4 JDBC driver works. The client layer is added for explanation purposes, but the client facility can also reside in Linux.

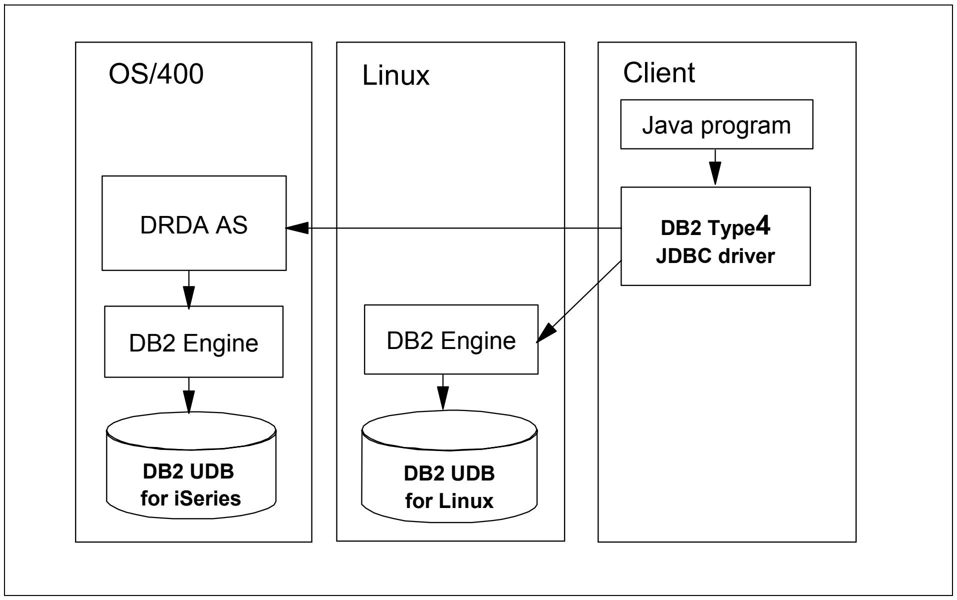


Figure 3-36 Type4 JDBC driver provided by DB2 UDB for Linux

The universal JDBC driver is the new JDBC driver which is provided with DB2 Version 8. This JDBC driver implements DRDA in it and can access all DB2 UDB for LUW directly.

Since type 3 is the applet driver, a typical Java application is supposed to use type 2 or type 4 driver. In this book, only type 2 and type 4 are discussed.

3.8.3 JDBC type 2 driver scenario

The JDBC type 2 driver relies on an OS-specific library (shared library) in the database client-end to communicate with an RDBMS. In the case of Linux application accessing DB2 UDB for iSeries, this library is packaged in the DB2 Client. In other words, you need a DB2 Client installed on the machine where the JDBC type 2 application runs. This scenario is described in Figure 3-37.

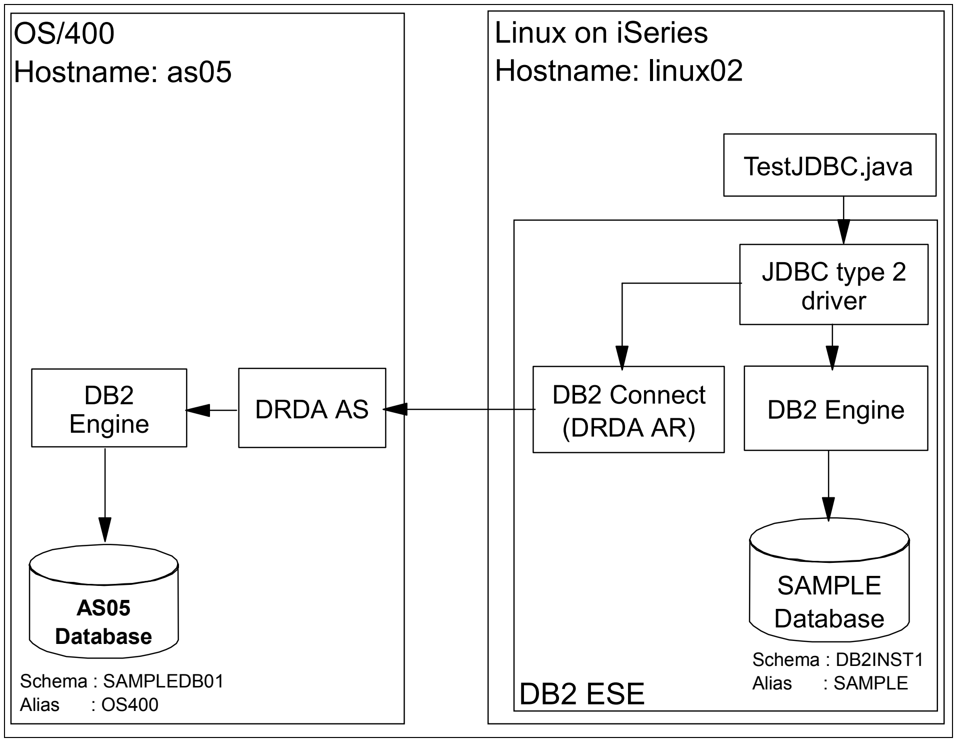


Figure 3-37 DB2 type2 JDBC driver Scenario

Note: On the Linux side, you don't need the DB2 Client, because it is integrated in every edition of DB2 UDB for LUW available for iSeries, including DB2 ESE.

How the JDBC type 2 driver works

The application will load the JDBC driver and the driver will use the shared library to communicate with the DB2 server. Actually, DB2 UDB for Linux on iSeries provides two different type 2 drivers:

- The CLI JDBC driver is provided in the file db2java.zip. The implementation package name is COM.ibm.db2.jdbc.app.DB2Driver. This driver has been used for the current J2EE certifications.
- The Universal JDBC driver, which is introduced as type 4 before, can also be used as the type 2 driver. This driver is provided in the file db2jcc.jar. The implementation package name is com.ibm.db2.jcc.DB2Driver. This driver is new to DB2 UDB V8.1. In the initial implementation (V8.1), this driver was for direct Java connectivity to DB2 servers using a type 4 driver architecture. In

DB2 V8.1.2, you can use this driver in a type 2 architecture with the local DB2 Client. One of the key reasons for using the driver in a type 2 architecture is for local application performance and for distributed transaction support. The Universal JDBC type 2 driver supports distributed transactions using com.ibm.db2.jcc.DB2XADataSource and connection pooling using com.ibm.db2.jcc.DB2ConnectionPoolDataSource.

Both of these JDBC type 2 drivers are supported with WebSphere Application Server. You can also use JDBC type 2 drivers to support Java-based client applications.

Configuration

This section provides a step-by-step process of JDBC type 2 driver configuration.

1. Locate the JDBC drivers path. Usually, all JDBC drivers are in:

   `HOME_DIR/sqllib/java,`

   Here *HOME_DIR* is the instance owner's home directory. In this scenario, the instance name is **`db2inst1`**, thus the full path is:

   `/home/db2inst1/sqllib/java`

   Note: Here are two other directories that may contain the JDBC drivers file:

   - /opt/IBM/db2/V8.1/java
   - /opt/IBM/db2/V8.1/java12

     We recommend that you do *not* to use them. They are the original copies for the creation of DB2 instances.

   - *HOME_DIR*/sqllib/java12

   We recommend that you do *not* to use it, because it is a back-level auxiliary directory for JDBC 1.2 compatibility.

2. Table 3-1 determines the appropriate JDBC driver you can use:

Table 3-1 Implementation Package JDBC drivers on DB2 UDB for Linux on iSeries

Driver Name	Type	Implementation Package
CLI JDBC driver	2	COM.ibm.db2.jdbc.app.DB2Driver
Universal JDBC driver	2 and 4	com.ibm.db2.jcc.DB2Driver

The CLI JDBC driver supports most existing applications, therefore it is recommended to customers considering compatibilities important. The DB2 Universal JDBC driver is recommended for new users.

Make sure the necessary directories are in the correct order in the $CLASSPATH variable. Type the following command:

```
echo $CLASSPATH
```

Example 3-4 illustrates one possible result of this command.

Example 3-4 Content of CLASSPATH Variable

```
db2inst1@linux02:~> echo $CLASSPATH
/home/db2inst1/sqllib/java/db2java.zip:/home/db2inst1/sqllib/java/db2jcc.jar:/h
ome/db2inst1/sqllib/java/sqlj.zip:/home/db2inst1/sqllib/function:/home/db2inst1
/sqllib/java/db2jcc_license_cisuz.jar:/home/db2inst1/sqllib/java/db2jcc_license
_cu.jar:.
```

Note that the result may be long, and in most cases, the default setting can work fine.

To use JDBC type2 driver indirectly access DB2 UDB for iSeries, the following files must be listed in the CLASSPATH variable:

- db2java.zip
- db2jcc.jar
- db2jcc_license_cisuz.jar

3. Determine the version and the location of IBM JDK.

   The JDBC driver in DB2 UDB for Linux on iSeries requires a 1.3.1 environment. A JDK for Linux on iSeries is provided along the DB2 Package. It is very important to verify the IBM JDK installation.

   - Type the following command under a DB2 instance owner to verify that the IBM Java Developer Kit is installed correctly in the Linux Environment:

     ```
     rpm -qa | grep IBMJava
     ```

   - Verify the JRE environment and the version information:

     ```
     echo $JAVA_HOME
     ```

     ```
     java -version
     ```

   - Make sure the file structure of the IBM JDBC products is complete:

     ```
     ls -la $JAVA_HOME
     ```

The output of the above commands should be the same as illustrated in Example 3-5:

Example 3-5 IBM JDK Information

```
db2inst1@linux02:~> rpm -qa | grep IBMJava
IBMJava2-SDK-1.3.1-5.0
db2inst1@linux02:~> echo $JAVA_HOME
/opt/IBMJava2-ppc-131
```

```
db2inst1@linux02:~> java -version
java version "1.3.1"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.1)
Classic VM (build 1.3.1, J2RE 1.3.1 IBM build cxppc32131-20030618 (JIT enabled:
jitc))
db2inst1@linux02:~> ls -la $JAVA_HOME
total 5503
drwxr-xr-x    8 root     root          248 2003-10-27 12:58 .
drwxr-xr-x   15 root     root          384 2003-10-27 12:57 ..
-rwxr-xr-x    1 root     root          492 2003-07-02 05:36 COPYRIGHT
drwxr-xr-x    3 root     root         1488 2003-10-27 12:57 bin
drwxr-xr-x   10 root     root          248 2003-10-27 12:57 demo
drwxr-xr-x    4 root     root          560 2003-10-27 12:57 docs
drwxr-xr-x    2 root     root          248 2003-10-27 12:57 include
drwxr-xr-x    4 root     root           96 2003-10-27 12:58 jre
drwxr-xr-x    2 root     root          152 2003-10-27 12:58 lib
-rw-r--r--    1 root     root      5623476 2003-07-02 05:36 src.jar
db2inst1@linux02:~>
```

4. If you have set the correct directory for $JAVA_HOME in Step 3, then skip this step.

   If not, pay attention to this section, because we will modify it. For example, the $JAVA_HOME variable is set to **`/usr/lib/java`**.

   In SLES 8, the default Java variables are globally defined in:

   `/etc/profile.d/alljava.sh`

   Edit this file according to Example 3-6.

Note: The product information and installation path for IBM JDK1.3.1 may be different from those in future IBM JDK 1.4.1.

Example 3-6 Sample Configuration file: /etc/profile.d/alljava.sh

```
#
#    /etc/profile.d/alljava.sh

#
# send feedback to feedback@suse.de

#
# This script sets some environment variables for default java.
# Affected variables: PATH, JAVA_BINDIR, JAVA_HOME, JRE_HOME,
#                     JDK_HOME, SDK_HOME
#
```

```
if [ -x /opt/IBMJava2-ppc-131/bin/java ] || [ -x /opt/IBMJava2-ppc-131/bin/jre
]
 ; then
  export PATH=$PATH:/opt/IBMJava2-ppc-131/bin
  export JAVA_BINDIR=/opt/IBMJava2-ppc-131/bin
  export JAVA_ROOT=/opt/IBMJava2-ppc-131
  export JAVA_HOME=/opt/IBMJava2-ppc-131
  if [ -x /opt/IBMJava2-ppc-131/jre/bin/java ] ; then
    export JRE_HOME=/opt/IBMJava2-ppc-131/jre
  else
    export JRE_HOME=/opt/IBMJava2-ppc-131
  fi
  unset JDK_HOME
  unset SDK_HOME
  if [ -x /opt/IBMJava2-ppc-131/bin/javac ] ; then
    # it is development kit
    if [ -x /opt/IBMJava2-ppc-131/bin/jre ] ; then
      export JDK_HOME=/opt/IBMJava2-ppc-131
    else
      export JDK_HOME=/opt/IBMJava2-ppc-131
      export SDK_HOME=/opt/IBMJava2-ppc-131
    fi
  fi
else
  if [ -x /opt/IBMJava2-ppc-131/jre/bin/java ] ; then
    # it is IBMJava2-JRE or SunJava2-JRE
    export PATH=$PATH:/opt/IBMJava2-ppc-131/jre/bin
    export JAVA_BINDIR=/opt/IBMJava2-ppc-131/jre/bin
    export JAVA_ROOT=/opt/IBMJava2-ppc-131
    export JAVA_HOME=/opt/IBMJava2-ppc-131/jre
    export JRE_HOME=/opt/IBMJava2-ppc-131/jre
    unset JDK_HOME
    unset SDK_HOME
  fi
fi
```

3.8.4 JDBC type 4 driver scenario

The type 4 driver is a pure Java JDBC driver that connects directly to the database server. What we mean by pure Java is that there is no need for a DB2 Client as in the JDBC type 2 driver. In this way, your application really can be portable. As far as your server platform supports Java Virtual Machine (JVM), your application will run with no other prerequisite product. This scenario is described in Figure 3-38.

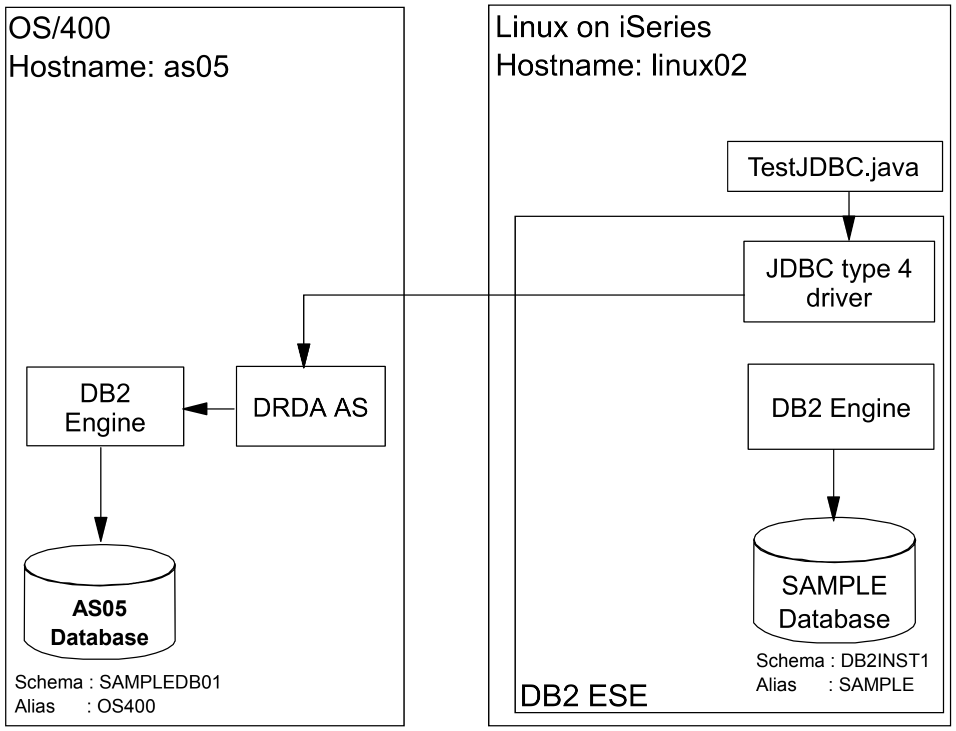


Figure 3-38 Scenario: JDBC type 4 driver

How JDBC type 4 driver works

DB2 UDB for LUW V8.1 introduced a type 4 driver known as the *Universal JDBC driver*. The Universal JDBC driver is provided in the file **`db2jcc.jar`**. The implementation package name is **`com.ibm.db2.jcc.DB2Driver`**.

Notice that the Universal type 2 and Universal type 4 drivers share the same implementation class name. There are several ways to distinguish which driver DB2 will instantiate internally:

1. Use a connection property to determine if the shared library will be used for connectivity (type 2) or if the driver will initiate a direct connection from the Java application (type 4).
2. Use a different connection URL pattern to indicate whether you want type 2 or type 4 behavior.
   - An example of a type 4 URL pattern:
     The string "**`jdbc:db2://as05:446/sample`**" tells the JDBC driver to connect directly from the Java application to the DB2 server to a database called

"sample" within the DB2 instance configured on the DB2 server (hostname is as05) that is listening on port 446.

– An example of a type 2 URL pattern:
  The string "**`jdbc:db2:sample`**" is used. Information about the **`sample`** is available in the DB2 client catalog directory.

The Universal JDBC driver is an abstract JDBC processor that is independent of driver-type connectivity or target platform. The Universal JDBC driver is an architecture-neutral JDBC driver for distributed and local DB2 UDB access. Because the Universal JDBC driver is independent of any particular JDBC driver-type connectivity or target platform, it supports both pure Java connectivity (type 4 driver) or JNI-based connectivity (type 2 driver) in a single driver instance to DB2 UDB. This driver can be used for standalone Java applications or multi-tier applications.

Note: As of DB2 UDB V8.1.2 the Universal JDBC driver requires a license JAR file to be in the CLASSPATH along with the db2jcc.jar file. These files are required:

- For the access to DB2 UDB for LUW servers:

  `db2jcc_license_cu.jar`

- For the access to DB2 UDB for iSeries (provided with DB2 Connect and Enterprise Server Edition):

  `db2jcc_license_cisuz.jar`

JDBC type 4 provides not only the database access functions but also the program communication functions. In terms of performance, therefore, if the Java application is executing on the same machine as the DB2 server, the best performance is with the JDBC type 2 driver, because TCP/IP overhead can be avoided and better throughput can be resulted.

When the DB2 server is on a different machine from the Java application, the performance of the type 4 and type 2 drivers is fairly similar. You should always consider writing more efficient SQL statements, or use SQLJ or stored procedures if you are attempting to improve overall Java application performance.

Configuration

To configure a JDBC type 4 driver, we can follow similar procedures to those in JDBC type 2. Refer to “Configuration” on page 82.

3.8.5 Sample JDBC program

This section introduces a sample program of JDBC. Example 3-7 shows the various JDBC access methods including type 2 and type 4.

Example 3-7 Various JDBC access to OS/400 and Linux DB2

```
/*
  TestJDBC.java a simple sample program of JDBC drivers provided by DB2 which
access Linux or OS/400 DB2
           Parameters :
           driver type     (DB2 JDBC driver type.  "Type2" or "Type4")
           Connection      ("OS400" means to connect to OS/400 via Type4.
                            "Linux" means to access the local Linux database.
                           "OS4002" means to access OS/400 using the universal
driver as Type2.
                            "Linux2" means to access the Local Linux database
using the universal driver as Type2.)
           schema          (The schema (OS/400 library) name)
           user            (OS/400 or DB2 instance user name.)
           password        (The password.)

  Example :   java TestJDBC Type4 OS400 SAMPLEDB01 DBDEMO password
*/

import java.io.*;
import java.sql.*;

public class TestJDBC
{
  public static void main(String argv[])
  {
    try{
      /*-----------------------------------------*/
      // JDBC driver selection
      /*-----------------------------------------*/
      if(argv[0].equals("Type2")){
        Class.forName("COM.ibm.db2.jdbc.app.DB2Driver");   // Type 2 driver
        System.out.println("Type 2 JDBC driver selected.");
      }
      if(argv[0].equals("Type4")){
        Class.forName("com.ibm.db2.jcc.DB2Driver");        // Type 4/2
universal driver
        System.out.println("Type 4/2 universal JDBC driver selected.");
      }

      /*-----------------------------------------*/
      // Connection type and database selection
      /*-----------------------------------------*/
      String url = null;

      /*--- Type 2 driver ---*/
      if(argv[0].equals("Type2")) {

        // Type 2 to local Linux
        if(argv[1].equals("Linux")) url = "jdbc:db2:SAMPLE";  1

        // Type 2 to OS/400 via DB2 Connect
        if(argv[1].equals("OS400")) url = "jdbc:db2:OS400";   2
      }

      /*--- Type 4 driver ---*/
      if(argv[0].equals("Type4")) {

        // Type 4 direct to local
        if(argv[1].equals("Linux")) url = "jdbc:db2://localhost:50001/SAMPLE";
3

        // Type 4 direct to OS/400
        if(argv[1].equals("OS400")) url = "jdbc:db2://192.168.1.1:446/AS05"; 4

        // Use Type 4 driver as type 2 to access local
        if(argv[1].equals("Linux2")) url= "jdbc:db2:SAMPLE"; 5

        // Use Type 4 driver as type 2 to access OS/400
        if(argv[1].equals("OS4002"))url = "jdbc:db2:OS400";  6
      }

      System.out.println("The URL string is : " + url);

      /*------------------------------------------*/
      // Access the table
      /*------------------------------------------*/
      Connection con = DriverManager.getConnection(url, argv[3], argv[4]);
      System.out.println("Database connected.");

      Statement stmt = con.createStatement();
      String sql = "SELECT ID, NAME, DEPT, JOB, YEARS, SALARY FROM " + argv[2]
+ ".STAFF WHERE ID > 280";
      ResultSet rs = stmt.executeQuery(sql);

      while(rs.next())
      {
        String  id     = rs.getString("ID");
        String  name   = rs.getString("NAME");
        String  dept   = rs.getString("DEPT");
        String  job    = rs.getString("JOB");
        String  years  = rs.getString("YEARS");
        String  salary = rs.getString("SALARY");
        System.out.println(id + "\t" + name + "\t" + dept + "\t" + job + "\t" +
years + "\t" + salary);
      }


      stmt.close();

    }catch(Exception e){
      System.out.println("Exception : " + e);
    }
  }
}
```

Type 2 driver

The JDBC driver is *COM.ibm.db2.jdbc.app.DB2Driver.* The URL is *jdbc:db2:(DBalias)*, where the DBalias is the alias of the database, which is cataloged in the database directory of the instance. In the Example 3-7, the URL is as follows:

```
// Type 2 to local Linux
if(argv[1].equals("Linux")) url = "jdbc:db2:SAMPLE";  1

// Type 2 to OS/400 via DB2 Connect
if(argv[1].equals("OS400")) url = "jdbc:db2:OS400";   2
```

“SAMPLE” and “OS400” are the aliases of the Linux and OS/400 database, respectively. In the latter case, DB2 UDB for iSeries is accessed via the DB2 Connect facility.

Type 4 driver

The JDBC driver is *com.ibm.db2.jcc.DB2Driver.* The URL is *jdbc:db2:(Host name or IP address):(DRDA port)/DBalias).* Type 4 driver connects to the database directly. The URL includes the hostname or IP address of the server where the actual DB2 instance resides and the port number of the DRDA service is working. DBalias is the database alias (Linux) or the relational database name (OS/400) of the DB2. In our case, the URL is as follows.

```
// Type 4 direct to local
if(argv[1].equals("Linux")) url = "jdbc:db2://localhost:50001/SAMPLE"; 3

// Type 4 direct to OS/400
if(argv[1].equals("OS400")) url = "jdbc:db2://192.168.1.1:446/AS05"; 4
```

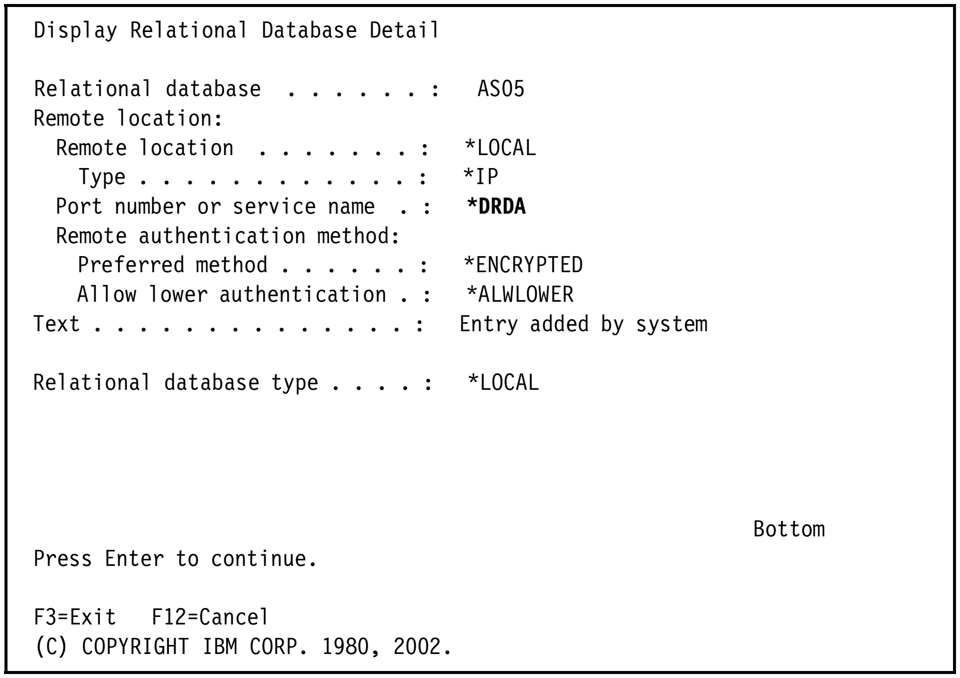
For the Linux DB2, the database is in the same Linux server (localhost) on which the program runs. The DRDA port number of the DB2 UDB for LUW is unique to its database instance and, by default, it is the sequence number beginning with 50001. The procedure to find the certain port is shown in Example 3-8. It shows how to know the SVCENAME parameter from the database instance and how to know the port number used by the service name.

Example 3-8 How to check the Linux DRDA server port number

```
db2inst1@linux02:~/redbook>db2 get database manager configuration|grep
SVCENAME
TCP/IP Service name (SVCENAME)=db2c_db2inst1
db2inst1@linux02:~/redbook>cat /etc/services|grep db2c_db2inst1
db2c_db2inst1 50001/tcp
```

For DB2 UDB for iSeries, the hostname of the iSeries server must be specified in the JDBC URL. In this case, the DB2 Connect installed in the Linux is not used, because the type 4 JDBC driver itself has the DB2 Connect facility. But this does not mean you do not need a DB2 Connect license on the Linux. You do need the DB2 Connect license to connect from Linux to DB2 UDB for iSeries.

By default, the port number of the DRDA service of the iSeries server is 446. To check it, you can use the OS/400 command DSPRDBDIRE RDB(DBname), where the DBname is the *LOCAL database name of the OS/400. Figure 3-39 illustrates the use of this command. The port number *DRDA means the port is 446.



```
Display Relational Database Detail

Relational database  . . . . . . :   AS05
Remote location:
  Remote location  . . . . . . . :   *LOCAL
    Type . . . . . . . . . . . . :   *IP
  Port number or service name  . :   *DRDA
  Remote authentication method:
    Preferred method . . . . . . :   *ENCRYPTED
    Allow lower authentication . :   *ALWLOWER
Text . . . . . . . . . . . . . . :   Entry added by system

Relational database type . . . . :   *LOCAL





                                                                  Bottom
Press Enter to continue.

F3=Exit   F12=Cancel
(C) COPYRIGHT IBM CORP. 1980, 2002.
```

Figure 3-39   How to check DRDA port of DB2 for iSeries

Use type 4 driver as type 2

The universal JDBC driver can be used as a type 2 driver. Sometimes this method is considered for performance reasons. To do it, you only need to specify the universal JDBC driver and specify a type 2 URL as follows:

```
// Use Type 4 driver as type 2 to access local
if(argv[1].equals("Linux2")) url= "jdbc:db2:SAMPLE"; 5

// Use Type 4 driver as type 2 to access OS/400
if(argv[1].equals("OS4002"))url = "jdbc:db2:OS400";  6
```

In this case, the universal JDBC driver uses DB2 Connect explicitly to connect to DB2 for iSeries.

Note: At the time of writing, this scenario did not work as illustrated in Example 3-9 for the portion of `db2inst1@linux:~/redbook>` **`java TestJDBC Type4 OS4002 SAMPLEDB01 DBDEMO password`**. Check the PTF for DB2 before you try this scenario.

Compile and run

To compile the program TestJDBC.java, check to see if the CLASSPATH is set to the JDBC driver first, then javac. Example 3-9 illustrates the procedure to compile and run TestJDBC.java program.

Example 3-9 Compile and run of TestJDBC.java program

```
db2inst1@linux:~/redbook>
db2inst1@linux:~/redbook> echo $CLASSPATH
/home/db2inst1/sqllib/java/db2java.zip:/home/db2inst1/sqllib/java/db2jcc.jar:/h
ome/db2inst1/sqllib/java/sqlj.zip:/home/dbinst1/sqllib/function:/home/db2inst1/
sqllib/java/db2jcc_license_cisuz.jar:/home/db2inst1/sqllib/java/db2jcc_license_
cu.ja:.
db2inst1@linux:~/redbook> javac TestJDBC.java
db2inst1@linux:~/redbook> java TestJDBC Type2 OS400 SAMPLEDB01 DBDEMO password
Type 2 JDBC driver selected.
The URL string is : jdbc:db2:OS400
Database connected.
290     Quill   84      Mgr     10      19818.00
300     Davis   84      Sales   5       15454.50
310     Graham  66      Sales   13      21000.00
320     Gonzales        66      Sales   4       16858.20
330     Burke   66      Clerk   1       10988.00
340     Edwards 84      Sales   7       17844.00
350     Gafney  84      Clerk   5       13030.50
db2inst1@linux:~/redbook> java TestJDBC Type2 Linux db2inst1 db2inst1 password
Type 2 JDBC driver selected.
The URL string is : jdbc:db2:SAMPLE
Database connected.
290     Quill   84      Mgr     10      19818.00
300     Davis   84      Sales   5       15454.50
310     Graham  66      Sales   13      21000.00
320     Gonzales        66      Sales   4       16858.20
330     Burke   66      Clerk   1       10988.00
340     Edwards 84      Sales   7       17844.00
350     Gafney  84      Clerk   5       13030.50
db2inst1@linux:~/redbook> java TestJDBC Type4 OS400 SAMPLEDB01 DBDEMO password
Type 4/2 universal JDBC driver selected.
The URL string is : jdbc:db2://192.168.1.1:446/AS05
Database connected.
290     Quill   84      Mgr     10      19818.00
300     Davis   84      Sales   5       15454.50
310     Graham  66      Sales   13      21000.00
320     Gonzales        66      Sales   4       16858.20
330     Burke   66      Clerk   1       10988.00
340     Edwards 84      Sales   7       17844.00
350     Gafney  84      Clerk   5       13030.50
db2inst1@linux:~/redbook> java TestJDBC Type4 Linux db2inst1 db2inst1 password
Type 4/2 universal JDBC driver selected.
```

```
The URL string is : jdbc:db2://localhost:50001/SAMPLE
Database connected.
290     Quill   84      Mgr     10      19818.00
300     Davis   84      Sales   5       15454.50
310     Graham  66      Sales   13      21000.00
320     Gonzales        66      Sales   4       16858.20
330     Burke   66      Clerk   1       10988.00
340     Edwards 84      Sales   7       17844.00
350     Gafney  84      Clerk   5       13030.50
db2inst1@linux:~/redbook> java TestJDBC Type4 OS4002 SAMPLEDB01 DBDEMO password
Type 4/2 universal JDBC driver selected.
The URL string is : jdbc:db2:OS400
Database connected.
Exception : com.ibm.db2.jcc.c.DisconnectException: Execution failed due to a
distribution protocol error that caused deallocation of the conversation.  The
access relational database command was not issued prior to a command requesting
RDB services.
db2inst1@linux:~/redbook> java TestJDBC Type4 Linux2 db2inst1 db2inst1 password
Type 4/2 universal JDBC driver selected.
The URL string is : jdbc:db2:SAMPLE
Database connected.
290     Quill   84      Mgr     10      19818.00
300     Davis   84      Sales   5       15454.50
310     Graham  66      Sales   13      21000.00
320     Gonzales        66      Sales   4       16858.20
330     Burke   66      Clerk   1       10988.00
340     Edwards 84      Sales   7       17844.00
350     Gafney  84      Clerk   5       13030.50
db2inst1@linux:~/redbook>
```

3.9 SQLJ scenario

In this section, we discuss the use of SQLJ in the DB2 Connect environment.

3.9.1 What is SQLJ ?

SQLJ is embedded SQL in Java and conforms to the ANSI standard database language. The SQLJ implementation can run on any operating system that has JDK. Although it is vendor-independent, we still use IBM JDK 1.3.1 in our environment. Our scenario is illustrated in Figure 3-40.

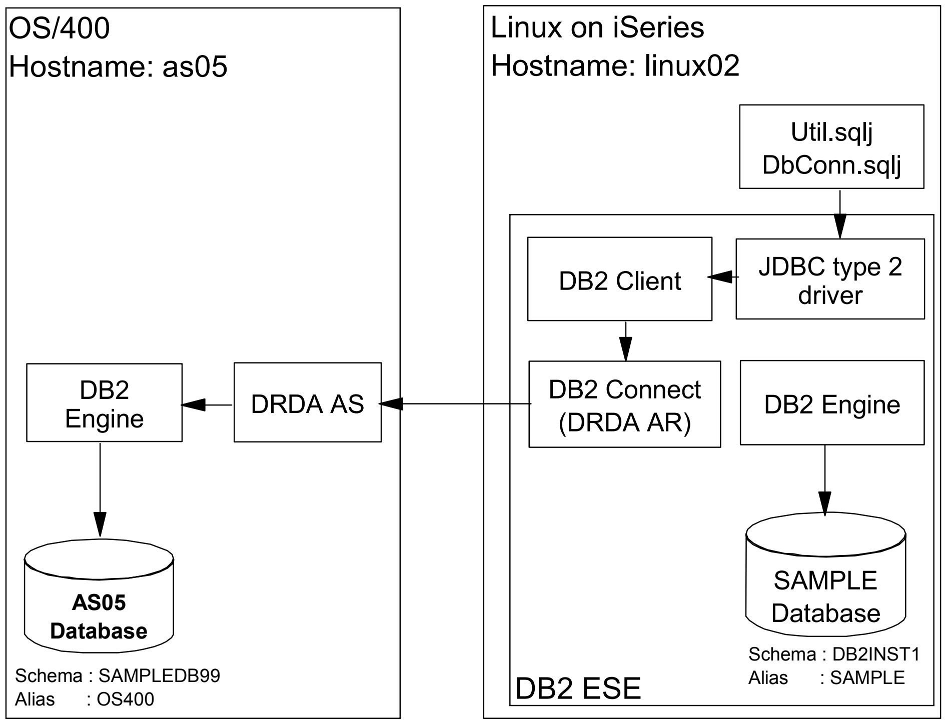


Figure 3-40 SQLJ Scenario

In this scenario, we discuss the SQLJ programming using the JDBC type 2 driver and DB2 Connect.

DB2 UDB for Linux on iSeries provides a Java-based compilation environment. You can use a set of programs to translate, customize and bind the SQLJ source code. The structure of the SQLJ is illustrated Example 3-41.

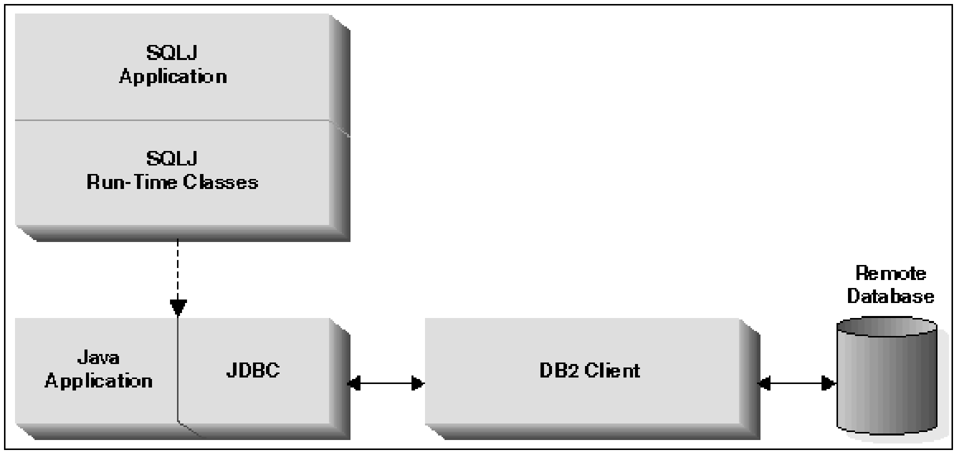


Figure 3-41 How SQLJ works

The SQLJ Run-time classes are the necessary Java packages and classes during SQLJ program compilation. For example, package **`sqlj.runtime.*`** and **`sqlj.runtime.ref.*`**, which are imported at the beginning of **`DbConn.sqlj`** program file, as illustrated at the beginning of the sample code, are shown in Example 3-13 on page 101.

3.9.2 How SQLJ works in Linux on iSeries

Developing applications using SQLJ involves three components: the translator, the profile customizer, and the profile binder. The utilities providing support to the three components are: **`sqlj`**, **`db2sqljcustomize`**, and **`db2sqljbind`**, respectively. The process as illustrated in Figure 3-42.

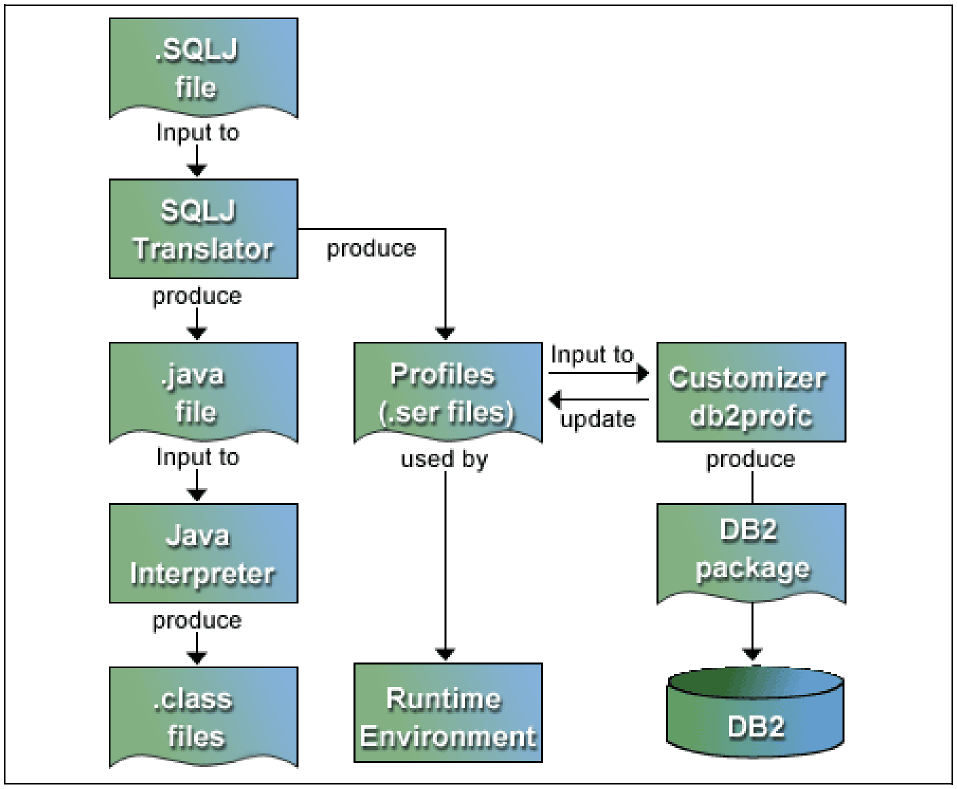


Figure 3-42 From SQLJ Source code to Java Class file

First, invoke the SQLJ translator (sqlj) to read in your SQLJ source file and check for proper SQLJ syntax in the program. The translator generates a Java source file and zero or more SQLJ profiles, and optionally compiles the generated Java source into byte codes (by default) if there are no errors. The generated Java source file replaces embedded SQL with calls to the SQLJ runtime, which executes the SQL operations.

Second, call the SQLJ profile customizer (db2sqljcustomize) to create a DB2 customization for the generated serialized profile. The customizer optionally (by default) performs an online check for SQL statements that can be dynamically prepared. The online check performs syntax, semantic, and schema validation. It also optionally (by default) calls the SQLJ profile binder to bind the DB2 packages.

If you choose not to perform the automatic bind during profile customization, you can invoke the SQLJ profile binder (db2sqljbind) separately to bind the previously customized SQLJ profile to the database.

3.9.3 Configuration

To configure a JDBC driver, we follow these procedures:

1. Create a sample database on iSeries, **`SAMPLEDB99`**.

   In the DB2 Command Line Processor, type the following command:

   ```
   call qsys.create_sql_sample('SAMPLEDB99')
   ```

   For details to create sample database on OS/400, refer to 3.4.1, "Create the sample database in OS/400" on page 56.

   Then from the Linux Partition, you can see the structure of the database on iSeries is:

   – Database Name: **`OS400`**
   – Schemas: **`SAMPLEDB99`**
   – TABLES: **`EMPLOYEE`**, etc.

2. Create a user profile on iSeries, **`SAMPLEDB99`**.

   Sign on as QSECOFR, in the OS/400 command line, type the following command to create user profile:

   ```
   CRTUSRPRF USRPRF(SAMPLEDB99)
   ```

   In this example, the password is **`‘password’`**.

   Make sure **`SAMPLEDB99`** has full access to the Library **`SAMPLEDB99`**.

> **Note:** For programming convenience, we create this user profile, so that the program can find the database objects (tables, views) during database connection without specifying the database objects' schema. Because the database manager can use the connection user name as the default schema.

3. Verify the Database OS400 is properly cataloged.

Example 3-10 Check catalog information using DB2 CLP

```
db2 => list database directory

 System Database Directory

 Number of entries in the directory = 2

Database 1 entry:

 Database alias                       = OS400
 Database name                        = RCHASM05
 Node name                            = NDE578BB
 Database release level               = a.00
 Comment                              =
```

```
 Directory entry type                  = Remote
 Authentication                        = SERVER
 Catalog database partition number     = -1

Database 2 entry:

 Database alias                        = SAMPLE
 Database name                         = SAMPLE
 Local database directory              = /home/db2inst1
 Database release level                = a.00
 Comment                               =
 Directory entry type                  = Indirect
 Catalog database partition number     = 0

db2 =>
```

4. Make a copy of the sample program for verification.

   Type the following commands:

   ```
   mkdir /home/db2inst1/javapgm
   cd /home/db2inst1/javapgm
   cp -R /home/db2inst1/sqllib/samples/java/* ./
   ```

For details for making a copy of the sample programs, refer to “Configuration” on page 87.

5. Configure the SQLJ working environment:

   In **home/db2inst1/javapgm,** use to following command to create a file called **sqlj.properties**

   edit sqlj.properties

   Add the following line to this file in this editor, and then save it. You can also create this file with other text editors.

   – **sqlj.url=jdbc:db2:os400**
   – **sqlj.driver=COM.ibm.db2.jdbc.app.DB2Driver**
   – **sqlj.online=sqlj.semantics.JdbcChecker**
   – **sqlj.offline=sqlj.semantics.OfflineChecker**

6. Compile **Util.sqlj** and **DbConn.sqlj**.

   Use the **sqlj** compiler to make class files.

Example 3-11 Compile the sample Code

```
db2inst1@linux02:~/javapgm/sqlj> sqlj -status Util.sqlj
[Translating 1 files.]
[Reading file Util]
[Translating file Util]
[Compiling 1 Java files.]
```

```
db2inst1@linux02:~/javapgm/sqlj> sqlj -status DbConn.sqlj
[Translating 1 files.]
[Reading file DbConn]
[Translating file DbConn]
[Compiling 1 Java files.]
db2inst1@linux02:~/javapgm/sqlj> ls *.class
Data.class  DbConn.class               SqljException.class
Db.class    DbConn_SJProfileKeys.class  Util_SJProfileKeys.class
```

7. Verify the Connection to database **OS400**.

Example 3-12 Connect to os400 database using SQLJ program

```
db2inst1@linux02:~/javapgm/sqlj> java DbConn os400 sampledb99 password

THIS SAMPLE SHOWS HOW TO CONNECT TO/DISCONNECT FROM DATABASES.

-----------------------------------------------------------
AND JAVA 2 CLASSES:
  Connection
  DriverManager
TO CONNECT TO/DISCONNECT FROM A DATABASE.
  Connect to 'os400' database using JDBC type 2 driver.

  DELETE FROM staff WHERE job = 'Mgr'
  ROLLBACK

  Disconnect from 'os400' database.
db2inst1@linux02:~/javapgm/sqlj>
```

3.9.4 Sample programs

This section analyzes the source code of Util.java. The Java program Util.sqlj provides the fundamental classes for DB2 access. These classes are commonly used for other sample programs in this directory.

Util.java provides three classes:

- Data - Display the data in the table, see 2 in Example 3-13
- Db - Connect to or disconnect from the database, see 3
- JdbcException - Handle Java Exceptions, see 4 i

In this program, three JDBC drivers are implemented in **class Db ()**. However, we omit the content of the class Db() and class Data() to maintain a clearer structure.

Example 3-13 Source code for Util.sqlj

```
import java.lang.*;
import java.sql.*;
import java.math.BigDecimal;
import sqlj.runtime.*; 1
import sqlj.runtime.ref.*; 1

class Data 2
{
Omitted
} // Data

class Db 3
{
Omitted
} // Db

class SqljException extends Exception 4
{
  public SqljException(Exception e)
  {
    super(e.getMessage());
  }

  public void handle()
  {
    try
    {
      System.out.println(getMessage());
      System.out.println();
      System.out.println("--Rollback the transaction-----");
      #sql {ROLLBACK};
      System.out.println("  Rollback done!");
    }
    catch (Exception e)
    {
    };
  } // handle

  public void handleExpectedErr()
  {
    System.out.println();
    System.out.println(
      "**************** Expected Error ******************\n");
    System.out.println(getMessage());
    System.out.println(
      "**************************************************");
  } // handleExpectedErr
} // SqljException
```

8. Source code for Util.java analysis

   The program **DbConn.sqlj** provides examples to connect to database using SQLJ.

   DbConn.sqlj uses 2 classes defined in **Util.sqlj**:

   – Db, see **1** in Example 3-14

   – JdbcException, see **2**

   All statements begin with #sql are to be precompiled. See **3**

Example 3-14 Source code for DbConn.sqlj

```
// Classes used from Util.sqlj are:
//        Db
//        SqljException
//


import java.sql.*;
import sqlj.runtime.*;
import sqlj.runtime.ref.*;

class DbConn
{
  public static void main(String argv[])
  {
    try
    {
      Db db = new Db(argv); 1

      System.out.println();
      System.out.println(
        "THIS SAMPLE SHOWS " +
        "HOW TO CONNECT TO/DISCONNECT FROM DATABASES.");

      System.out.println();
      System.out.println(
        "----------------------------------------------------------\n" +
        "AND JAVA 2 CLASSES:\n" +
        "  Connection\n" +
        "  DriverManager\n" +
        "TO CONNECT TO/DISCONNECT FROM A DATABASE.");

      // connect to the 'sample' database
      db.getDefaultContext();
      // perform an SQL statement
      System.out.println();
      System.out.println("  DELETE FROM staff WHERE job = 'Mgr'");

      #sql {DELETE FROM staff WHERE job = 'Mgr'}; 3

      System.out.println("  ROLLBACK");
      #sql {ROLLBACK}; 3

      // disconnect from the 'samples' database
      db.disconnect();
    }
    catch (Exception e)
    {
      SqljException sqljExc = new SqljException(e); 2
      sqljExc.handle();
    } // try - catch
  } // main
} // DbConn
```

A

Accessing DB2 UDB for Linux from OS/400 application

This appendix provides information to configure the OS/400 DRDA AR and Linux AS. OS/400 CLI and Embedded SQL in RPG sample programs are also provided.

Overview

In this appendix, we show some of the ways to access DB2 UDB for Linux from the OS/400 application.

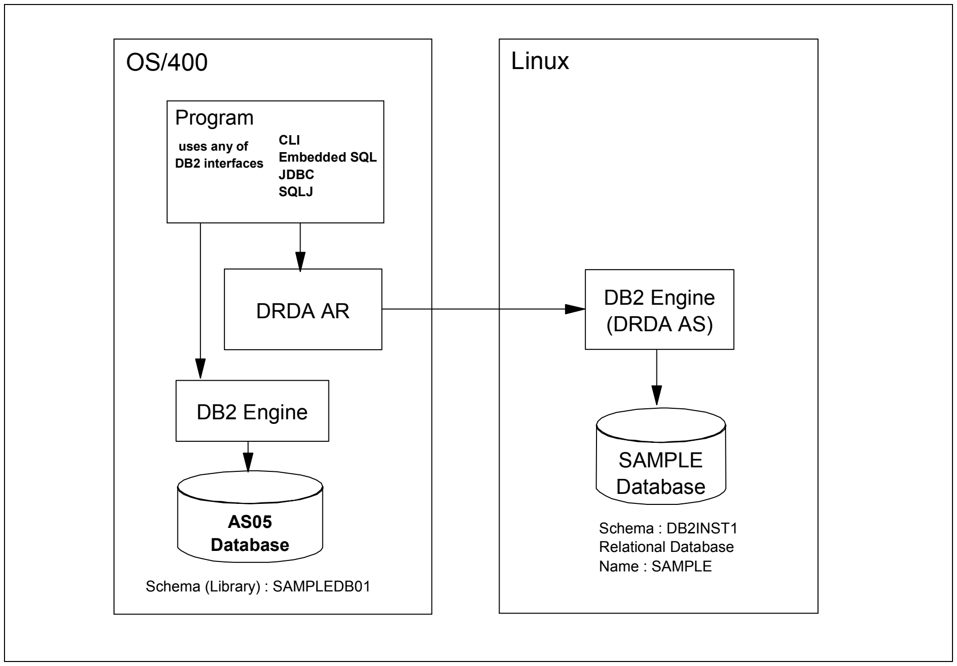


Figure A-1 Accessing DB2 for Linux from OS/400 Application

As illustrated in Figure A-1, OS/400 application can access the Linux DB2 using CLI, Embedded SQL, JDBC, and SQLJ. In this case, DB2 UDB for iSeries is the DRDA AR and DB2 UDB for Linux is the AS. Note that there is no need to install DB2 Connect in the Linux side since DB2 UDB for Linux has the DRDA AS facility as its standard function. DB2 UDB for iSeries also have DRDA AR facility. OS/400 application can access the local DB2 UDB for iSeries using CLI or other methods as well.

Requirements

The OS/400 Interactive SQL and Embedded SQL facility for OS/400 programs are provided by the licence program *DB2 Query Manager and SQL Development Kit for iSeries* (5722-ST1). You have to install the product when using interactive SQL or when compiling embedded SQL programs. At run time of the embeds

SQL application, you do not need the product. CLI method does not requires 5722-ST1 either.

Configuration

In this section, we show the way to configure DRDA environment to enable OS/400 AR.

OS/400 configuration

This section describes how to configure on the OS/400 side.

OS/400 Relational database directory

When accessing from Linux to OS/400, the Linux application finds the OS/400 database using the DB2 node directory and the database directory of the Linux DB2. When accessing from OS/400 to Linux, the OS/400 application uses the OS/400 Relational Database Directory to find Linux DB2.

We need to add the remote database directory entry for the SAMPLE database in the Linux partition using the following command:

```
ADDRDBDIRE RDB(SAMPLE) RMTLOCNAME('192.168.1.2' *IP) PORT(50001)
RMTAUTMTH(*USRIDPWD)
```

Where :

- `SAMPLE` is the alias of the Linux DB2.
- 192.168.1.2 is the IP address of the Linux partition.
- 50001 is the port number which the DRDA server of the Linux DB2 is working.

Generally the “well known” port of the DRDA server is 446 but for DB2 UDB for LUW, the port is different for each instance. Usually the port is the serial number of the created instance that begins with 50001. To check the port number in Linux side, you need to search the service table of `/etc/hosts` file once you get the database manager configuration done as illustrated in Figure A-2. The port name is set in the `SVCENAME` parameter of the database manager configuration of the instance.

```
db2inst1@linux02:~/redbook> db2 get database manager configuration|grep
SVCENAME
 TCP/IP Service name                          (SVCENAME) = db2c_db2inst1
db2inst1@linux02:~/redbook> cat /etc/services|grep db2c_db2inst1
db2c_db2inst1   50001/tcp
db2inst1@linux02:~/redbook>
```

Figure A-2 How to check the Linux side DRDA service port number

You can check the relational database directory configuration using `WRKRDBDIRE` command as shown in Figure A-3.

```
 Work with Relational Database Directory Entries

 Position to  . . . . . .

 Type options, press Enter.
   1=Add   2=Change   4=Remove   5=Display details   6=Print details

         Relational          Remote
 Option  Database            Location                  Text

         SAMPLE              192.168.1.2
AS05            *LOCAL                   Entry added by system









                                                                    Bottom
 F3=Exit   F5=Refresh   F6=Print list   F12=Cancel
 (C) COPYRIGHT IBM CORP. 1980, 2002.
```

Figure A-3 Registration of the Linux database

User profile

To use in this scenario, we create a user profile whose name and password are same as the Linux DB2 instance owner. As we use this user profile to compile the CLE program, we give the `*ALLOBJ` special authority to this user.

```
CRTUSRPRF USRPRF(DB2INST1) PASSWORD(XXXXX) USRCLS(*PGMR) SPCAUT(*ALLOBJ)
CCSID(37)
```

The iSeries server is shipped with a QCCSID value set to 65535. Data tagged with this CCSID is not to be converted by the receiving server. You may not be able to connect to an unlike server when your iSeries server application requester (AR) is using this CCSID. In our environment, the SYSVAL QCCSID is also 65535 then we set the CCSID(37) to this user profile explicitly.

Linux configuration

This section describes how to configure on the Linux side.

Create schema QSQL400

This is the configuration for interactive SQL. When running the OS/400 interactive SQL, an SQL package must be created in the schema named QSQL400 in the Linux database instance. To enable it, we create the schema explicitly in Linux database instance `db2inst1`. Figure A-4 shows this.

```
db2inst1@linux02:~/redbook> db2
(c) Copyright IBM Corporation 1993,2002
Command Line Processor for DB2 SDK 8.1.2

You can issue database manager commands and SQL statements from the command
prompt. For example:
    db2 => connect to sample
    db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
 ? CATALOG DATABASE for help on the CATALOG DATABASE command
 ? CATALOG          for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => connect to SAMPLE

   Database Connection Information

 Database server        = DB2/LINUXPPC 8.1.2
 SQL authorization ID   = DB2INST1
 Local database alias   = SAMPLE

db2 => create schema QSQL400 1
DB20000I  The SQL command completed successfully.
db2 => terminate
DB20000I  The TERMINATE command completed successfully.
db2inst1@linux02:~/redbook>
```

Figure A-4 Create the schema QSQL400

`create schema QSQL400` 1

This command creates the schema.

Interactive SQL

Interactive SQL is the command line SQL processor on OS/400 such like DB2 CLP on the Linux DB2.You can operate any DB2 database interactively using the

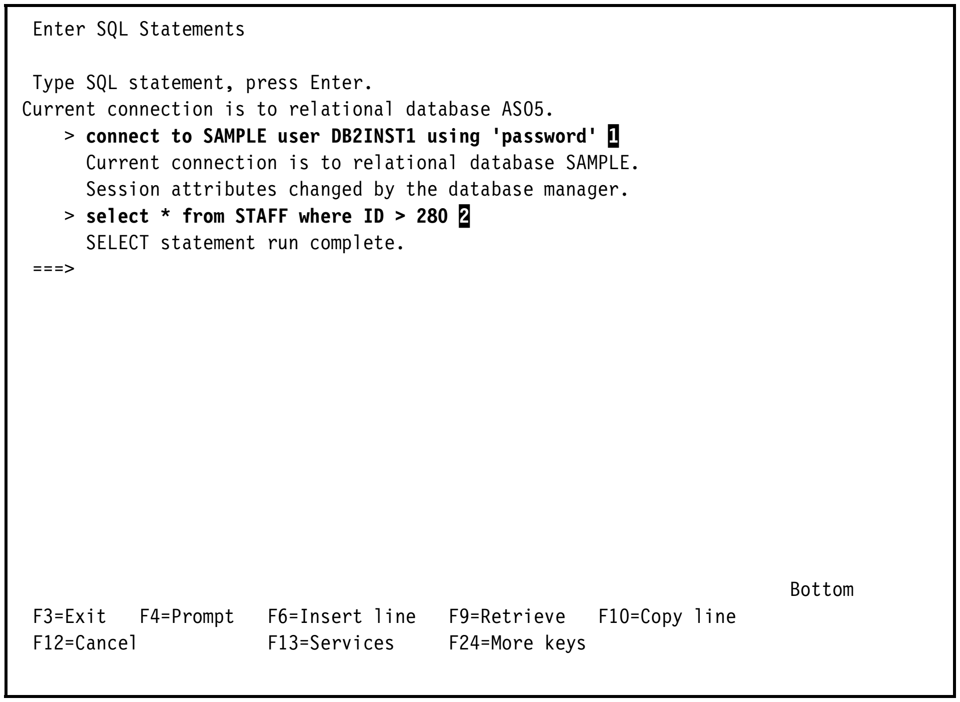
Interactive SQL program.Sign on as DB2INST. To access Linux SAMPLE database from OS/400 Interactive SQL, issue the following command after signing on as the user DB2INST1.

```
STRSQL COMMIT(*CHG) NAMING(*SQL)
```

Then the interactive SQL session opens. Figure A-5 shows the screen.

```
connect to SAMPLE user DB2INST1 using 'password' 1
```

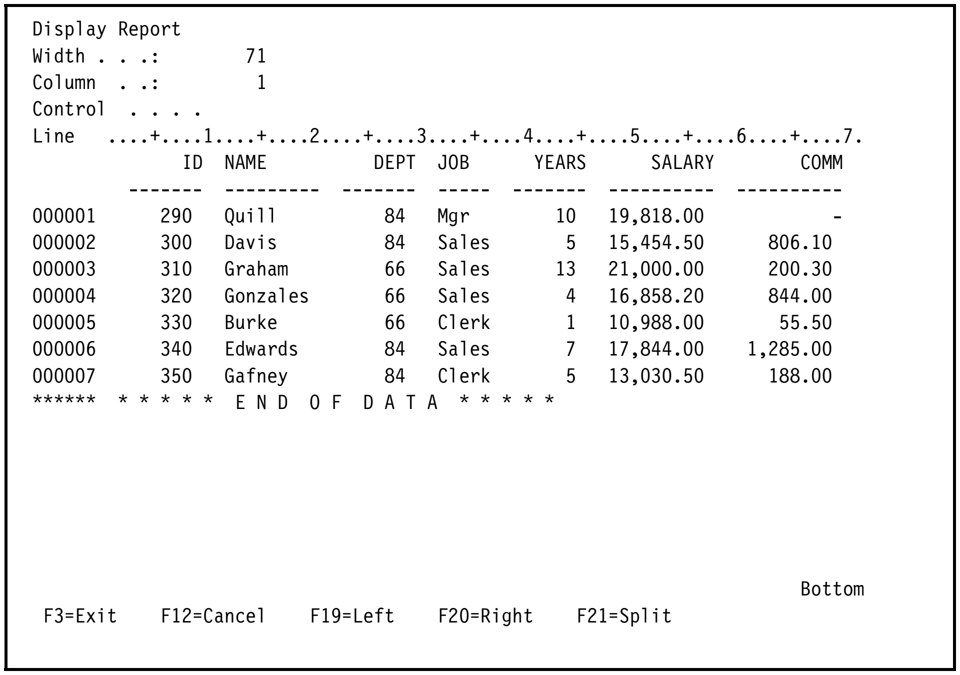
By default, the local database (AS05) is connected. To connect to Linux SAMPLE database, issue the connect command. SAMPLE is the name which we registered in the OS/400 relational database directory.



```
 Enter SQL Statements

 Type SQL statement, press Enter.
Current connection is to relational database AS05.
    > connect to SAMPLE user DB2INST1 using 'password' 1
      Current connection is to relational database SAMPLE.
      Session attributes changed by the database manager.
    > select * from STAFF where ID > 280 2
      SELECT statement run complete.
 ===>















                                                                         Bottom
 F3=Exit   F4=Prompt   F6=Insert line   F9=Retrieve   F10=Copy line
 F12=Cancel            F13=Services     F24=More keys
```

Figure A-5 Connect to Linux SAMPLE database

After connecting to the database, you can run the SQL statements. For example, if you run the `select` statement as in 2, the screen will be changed as in Figure A-6, which shows the data retrieved from the Linux SAMPLE database.



```
Display Report
Width . . .:       71
Column  . .:        1
Control  . . . .
Line   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7.
              ID  NAME          DEPT  JOB      YEARS      SALARY        COMM
         -------  ---------  -------  -----  -------  ----------  ----------
000001       290  Quill           84  Mgr         10  19,818.00            -
000002       300  Davis           84  Sales        5  15,454.50      806.10
000003       310  Graham          66  Sales       13  21,000.00      200.30
000004       320  Gonzales        66  Sales        4  16,858.20      844.00
000005       330  Burke           66  Clerk        1  10,988.00       55.50
000006       340  Edwards         84  Sales        7  17,844.00    1,285.00
000007       350  Gafney          84  Clerk        5  13,030.50      188.00
******  * * * * *  E N D  O F  D A T A  * * * * *





                                                                        Bottom
 F3=Exit    F12=Cancel    F19=Left    F20=Right    F21=Split
```

Figure A-6 The SQL result

CLI scenario

Example A-1 is the OS/400 ILE C program. It is almost same as the program testCLI.c but the only change we needed is the line **1** because the OS/400 include file `sqlcli.h` does not have this entry. `(((res)&(~1))==0)` is the actual macro replacement which the `sqlcli.h` of the DB2 UDB for Linux does.

Example: A-1   QCSRC/TESTCLI

```
Columns . . . :    1 121                              Browse
 SEU==>
 FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
       *************** Beginning of data *********************************************************************
/*
   testCLI.c a simple sample program of DB2 CLI which access Linux or OS/400 DB2
             Parameters :
             databasename   (The alias of the database registered in DB2 directory for CLI
                            The datasource name fot ODBC)
             schema         (The schema name of the database object.
                            Linux user name or OS/400 library name)
             validUserID    (The user ID or the user profile of the database. )
             password       (The password)

   Example :   testCLI mydb2 myschema myself mypassword
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlcli.h>         /* For CLI  */
/* #include <sql.h>    */   /* For ODBC */

void diehappy(char *);
void die(char *);
void die2(char *, SQLRETURN);

/* The main routine */
int main(int argc, char *argv[]) {

#define MAX_CONNECTIONS 5

 /*-------------------------------*/
 /* Preparation                   */
 /*-------------------------------*/
 /* The buffers which holds the data from the table */
 char field1[100];
 char field2[100];
 char field3[100];
 char field4[100];
 char field5[100];
 char field6[100];
 char querystring[200];

 SQLINTEGER retfld1;
 SQLINTEGER retfld2;
 SQLINTEGER retfld3;
 SQLINTEGER retfld4;
 SQLINTEGER retfld5;
 SQLINTEGER retfld6;

 SQLHENV henv;
 SQLHDBC hdbc[MAX_CONNECTIONS];
 SQLHSTMT statement;
 SQLRETURN sqe=0;
 SQLRETURN res=0;

 /* Allocate the environment handle */
 if ( SQLAllocEnv(&henv)) die("Alloc Environment");

 /* Allocate the Connection handle */
 if ((sqe=SQLAllocConnect(henv,&hdbc[0]))) die2("Allocate Connect",sqe);

 /* Get the database handle */
 if ( (sqe= SQLConnect(hdbc[0],argv[1],SQL_NTS, argv[3],SQL_NTS, argv[4],SQL_NTS))) die2("Connect",sqe);

 /* Allocate the statement handle */
 if (SQLAllocStmt(hdbc[0],&statement)) die("Alloc stmt");

 /*-------------------------------*/
 /* The core process              */
 /*-------------------------------*/
 /* Execute the SQL */
 querystring[0] = 0x00;
 strcat(querystring, "SELECT ID, NAME, DEPT, JOB, YEARS, SALARY FROM ");
 strcat(querystring, argv[2]);
 strcat(querystring, ".STAFF WHERE ID > 280");
 printf("Running SQL : %s\n", querystring);
 if ((sqe=SQLExecDirect(statement,querystring, SQL_NTS))) die2("exec direct",sqe);

 /* Bind the buffers to the data to be fetched */
 if ((sqe=SQLBindCol(statement, 1, SQL_CHAR, field1,100, &retfld1))) die2("Bind field 1", sqe);
 if ((sqe=SQLBindCol(statement, 2, SQL_CHAR, field2,100, &retfld2))) die2("Bind field 2", sqe);
 if ((sqe=SQLBindCol(statement, 3, SQL_CHAR, field3,100, &retfld3))) die2("Bind field 3", sqe);
 if ((sqe=SQLBindCol(statement, 4, SQL_CHAR, field4,100, &retfld4))) die2("Bind field 4", sqe);
 if ((sqe=SQLBindCol(statement, 5, SQL_CHAR, field5,100, &retfld5))) die2("Bind field 5", sqe);
 if ((sqe=SQLBindCol(statement, 6, SQL_CHAR, field6,100, &retfld6))) die2("Bind field 6", sqe);


 /* The loop to fetch the all selected data */
 res = SQLFetch(statement);
 /*while (SQL_SUCCEEDED(res)) { */
 while ((((res)&(~1))==0)) {                                            1

   /* Print the data with tab separated */
   printf("%s\t%s\t%s\t%s\t%s\t%s\n",field1,field2,field3,field4,field5,field6);
   res= SQLFetch(statement);  /* Next record */
 }

 /*------------------------------*/
 /* Clean up                    */
 /*------------------------------*/
 /* Deallocate the statement handle */
 if (SQLFreeStmt(statement,SQL_DROP)) die("Free Statement with Drop");

 /* Deallocate the connection handle */
 if ((sqe=SQLDisconnect(hdbc[0] ))) die2("Disconnect ",sqe);
 if ((sqe=SQLFreeConnect(hdbc[0]))) die2("Free Connection",sqe);

 /* Deallocate the environment handle */
 if (SQLFreeEnv(henv)) die("Free Environment");

 diehappy("Got to end");

 return 0;
}


/* dieXXX() are the functions to print the ending status */

void diehappy(char *reason) {
  printf("%s\n",reason);
}

void die(char *reason) {
  printf("Failure because: %s\n",reason);
  exit(1);
}

void die2(char *reason,SQLRETURN code) {
  printf("Error code: %d\n",code);
  die(reason);
}


       ****************** End of data ***********************************************************************
```

Compile and run

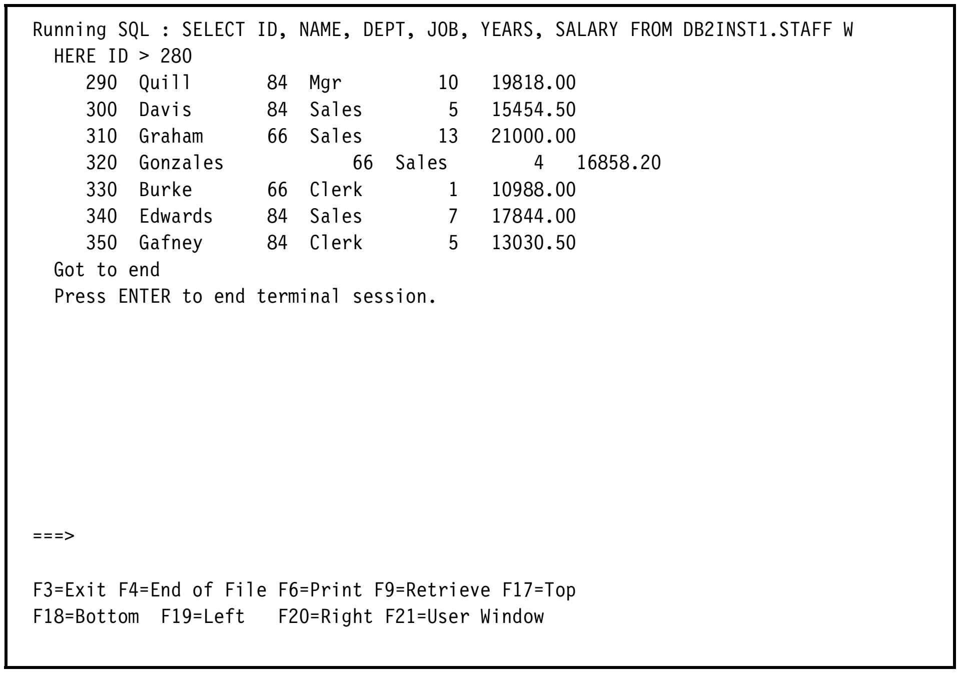
To compile the TESTCLI program, simply use the CRTBNDC command.

```
CRTBNDC PGM(REDBOOK/TESTCLI) SRCFILE(REDBOOK/QCSRC)
```

To run the TESTCLI, signing on as the user DB2INST1, use CALL command with parameters, which are the database names to connect, schema name of the table to access, user name of the DB2 instance, and the password.

```
CALL REDBOOK/TESTCLI PARM('SAMPLE' 'DB2INST1' 'DB2INST1' 'password')
```

Figure A-7 shows the result. The data retrieved from Linux is shown.

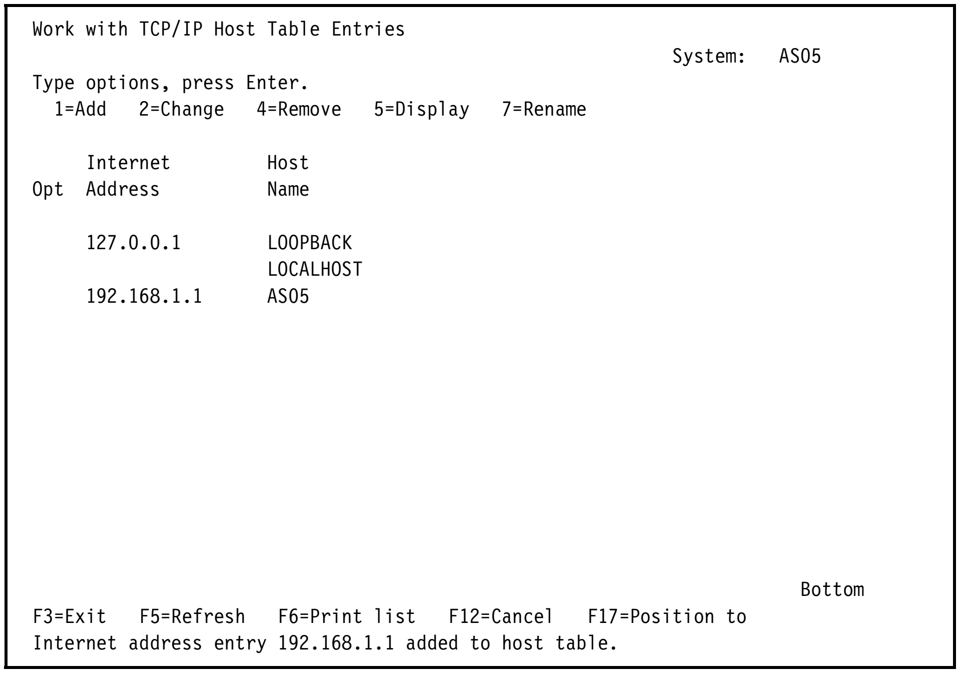


```
Running SQL : SELECT ID, NAME, DEPT, JOB, YEARS, SALARY FROM DB2INST1.STAFF W
  HERE ID > 280
     290  Quill       84  Mgr         10   19818.00
     300  Davis       84  Sales        5   15454.50
     310  Graham      66  Sales       13   21000.00
     320  Gonzales            66  Sales        4   16858.20
     330  Burke       66  Clerk        1   10988.00
     340  Edwards     84  Sales        7   17844.00
     350  Gafney      84  Clerk        5   13030.50
  Got to end
  Press ENTER to end terminal session.











===>


F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
F18=Bottom  F19=Left   F20=Right F21=User Window
```

Figure A-7 The screen after running the TESTCLI program

CLI for OS/400 local DB

The TESTCLI program can also run to the OS/400 local database. To enable it, the local database name must be registered in the TCP/IP host table. Check `CFGTCP` and **10. Work with TCP/IP host table entries**. Figure A-8 shows the settings of our machine. 192.168.1.1 is the virtual LAN IP address of the OS/400 partition.



```
Work with TCP/IP Host Table Entries
                                                            System:   AS05
Type options, press Enter.
  1=Add   2=Change   4=Remove   5=Display   7=Rename

     Internet         Host
Opt  Address          Name

     127.0.0.1        LOOPBACK
                      LOCALHOST
     192.168.1.1      AS05

















                                                                        Bottom
F3=Exit   F5=Refresh   F6=Print list   F12=Cancel   F17=Position to
Internet address entry 192.168.1.1 added to host table.
```

Figure A-8 Check the OS/400 TCP/IP host table

To run TESTCLI for the local database, issue the CALL command from the DB2INST1's 5250 session as follows :

```
CALL REDBOOK/TESTCLI PARM('AS05' 'SAMPLEDB01' 'DB2INST1' 'password')
```

Tip: If the program is terminated in error, the OS/400 job state often goes to some error state of database access. Usually they are remaining opened database connections or transactions which have not committed yet. To clean up the state, you should once sign off and sign on again. COMMIT or ROLLBACK CL commands are also useful. You can issue them from the command line.

Embedded SQL scenario

Example A-2 (TESTESQL) is the OPM RPG simple embedded SQL program which does the same work as `tectESQL.c` in 3.7, "Embedded SQL scenario" on page 69. Instead of showing the retrieved data on the screen, this program makes a spool file. The database name and schema name is hardcoded and this program takes no parameter.

Example: A-2   OPM RPG program to access Linux database (QRPGSRC/TESTESQL)

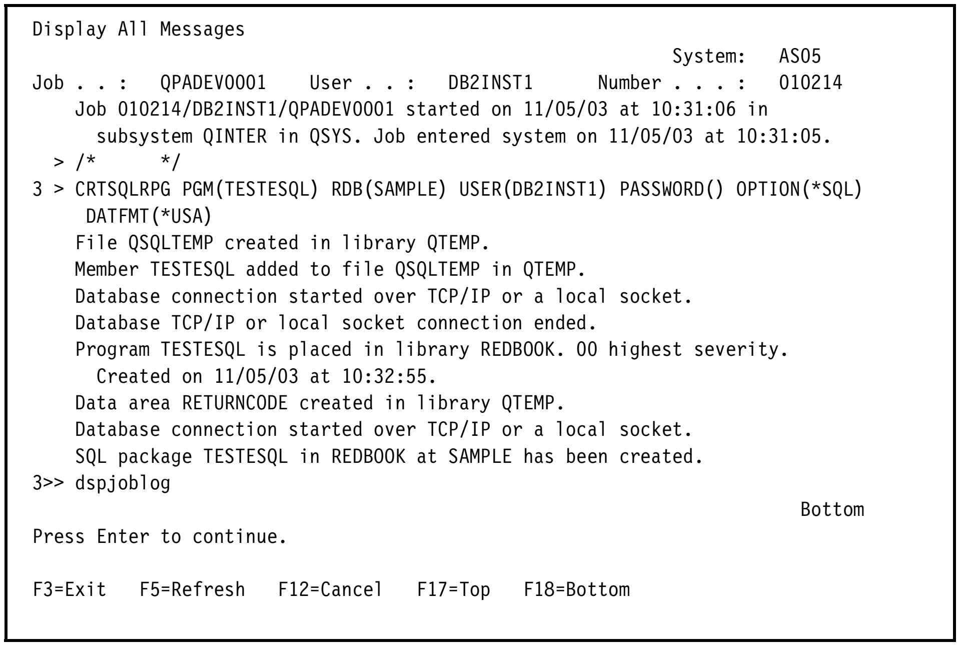
```
*************** Beginning of data ************************************
FQPRINT  O   F     132            PRINTER
C                     Z-ADD0         ID      50
C                     MOVE *BLANK    NAME    9
C                     Z-ADD0         DEPT    50
C                     MOVE *BLANK    JOB     5
C                     Z-ADD0         YEARS   50
C                     Z-ADD0         SALARY  90
C*
C/EXEC SQL WHENEVER SQLERROR GO TO ERROR
C/END-EXEC
C*
C/EXEC SQL DECLARE C1 CURSOR FOR
C+ SELECT ID, NAME, DEPT, JOB, YEARS, SALARY FROM DB2INST1.STAFF
C+ WHERE ID > 280
C/END-EXEC
C*
C/EXEC SQL OPEN C1
C/END-EXEC
C*
C/EXEC SQL WHENEVER NOT FOUND GO TO DONE
C/END-EXEC
C*
C           SQLCOD    DOUNE0
C/EXEC SQL FETCH C1 INTO :ID, :NAME, :DEPT, :JOB, :YEARS, :SALARY
C/END-EXEC
C           SQLCOD    IFEQ 0
C                     EXCPTOUT
C                     ENDIF
C                     ENDDO
C*
C           DONE      TAG
C/EXEC SQL CLOSE C1
C/END-EXEC
C*
C           ERROR     TAG
C                     SETON                      LR
C                     RETRN
C*
OQPRINT  E                OUT
O                         ID    4    5
O                         NAME      17
O                         DEPT  4   19
O                         JOB       30
O                         YEARS 4   38
O                         SALARY2   50
        ****************** End of data **************************************
```

Compile and run

The following command compiles the program TESTESQL and creates an executable module. As well as Linux C embedded SQL scenario, the connection to the target Linux database is also needed to create and store the SQL package for this program when compiling. The USER and PASSWORD are used to connect to the Linux SAMPLE database to create the SQL package.

```
CRTSQLRPG PGM(TESTESQL) RDB(SAMPLE) USER(DB2INST1) PASSWORD('password')
OPTION(*SQL) DATFMT(*USA)
```

Figure A-9 shows the job log of the compile. You can see that the precompiler generates a source member in QTEMP/QSQLTEMP, then the RPG compiler generates the program. The database connection to the SAMPLE database is also held and an SQL package is created and stored. These are similar processes to the Linux case as in 3.7, “Embedded SQL scenario” on page 69.



```
Display All Messages
                                                            System:   AS05
Job . . :   QPADEV0001    User . . :   DB2INST1      Number . . . :   010214
    Job 010214/DB2INST1/QPADEV0001 started on 11/05/03 at 10:31:06 in
      subsystem QINTER in QSYS. Job entered system on 11/05/03 at 10:31:05.
  > /*      */
3 > CRTSQLRPG PGM(TESTESQL) RDB(SAMPLE) USER(DB2INST1) PASSWORD() OPTION(*SQL)
     DATFMT(*USA)
    File QSQLTEMP created in library QTEMP.
    Member TESTESQL added to file QSQLTEMP in QTEMP.
    Database connection started over TCP/IP or a local socket.
    Database TCP/IP or local socket connection ended.
    Program TESTESQL is placed in library REDBOOK. 00 highest severity.
      Created on 11/05/03 at 10:32:55.
    Data area RETURNCODE created in library QTEMP.
    Database connection started over TCP/IP or a local socket.
    SQL package TESTESQL in REDBOOK at SAMPLE has been created.
3>> dspjoblog
                                                                        Bottom
Press Enter to continue.

F3=Exit   F5=Refresh   F12=Cancel   F17=Top   F18=Bottom
```

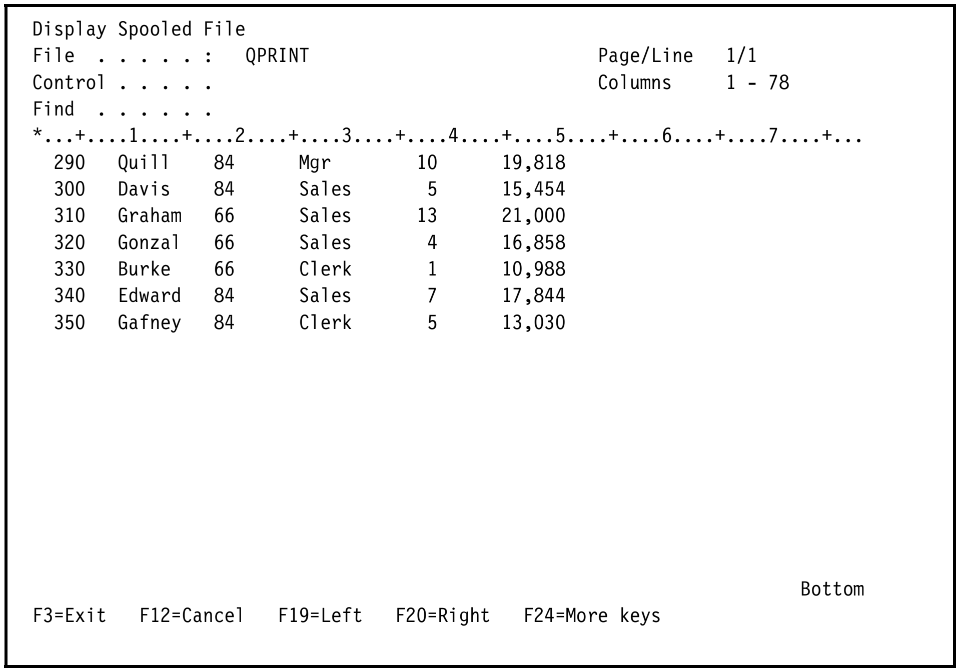
Figure A-9 TESTESQL compile JOBLOG

To run the program, we have to provide the run time user and password information to the program. It can be done by the following command.

```
ADDSVRAUTE USRPRF(DB2INST1) SERVER(SAMPLE) USRID(DB2INST1) PASSWORD(‘password’)
```

To store the password in the system, the SYSVAL QRETSVRSEC must be set to ‘1’.

After the setup, you can run the program simply issuing `CALL TESTESQL`. Figure A-10 shows the spool data created by TESTESQL. These are the data retrieved from Linux DB2.



```
Display Spooled File
File  . . . . . :   QPRINT                           Page/Line   1/1
Control . . . . .                                    Columns     1 - 78
Find  . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
  290   Quill    84      Mgr       10      19,818
  300   Davis    84      Sales      5      15,454
  310   Graham   66      Sales     13      21,000
  320   Gonzal   66      Sales      4      16,858
  330   Burke    66      Clerk      1      10,988
  340   Edward   84      Sales      7      17,844
  350   Gafney   84      Clerk      5      13,030









                                                                       Bottom
F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

Figure A-10   The spool data generated by TESTESQL

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 122. Note that some of the documents referenced here may be available in softcopy only.

- *Up and Running with DB2 for Linux,* SG24-6899
- *WebSphere for Linux on iSeries: Implementation Guide,* SG24-6958
- *Linux on the IBM @server iSeries Server: An Implementation Guide,* SG24-6232
- *Linux Handbook: A Guide to IBM Linux Solutions and Resources,* SG24-7000
- *Linux on iSeries Sales Guide,* SG24-7058
- *Linux for WebSphere and DB2 Servers,* SG24-5850
- *Linux Web Hosting with WebSphere, DB2, and Domino,* SG24-6007
- *LPAR Configuration and Management Working with IBM @server iSeries Logical Partitions,* SG24-6251

Other publications

These publications are also relevant as further information sources:

- *IBM DB2 Universal Database Administration Guide: Planning*, SC09-4822
- *IBM DB2 Universal Database Administration Guide: Implementation*, SC09-4820
- *Application Development Guide: Programming Client Applications,* SC09-4826
- *Call Level Interface Guide and Reference, Volumes 1,* SC09-4849
- *Call Level Interface Guide and Reference, Volumes 2,* SC09-4850

Online resources

These Web sites and URLs are also relevant as further information sources:

- "Which Edition of DB2 Connect is Right for You?"

http://www7b.software.ibm.com/dmdd/library/techarticle/0303zikopoulos1/0303zikopoulos1.html

- The .NET Data Provider is part of the iSeries Access for Windows Beta:

  http://www-1.ibm.com/servers/eserver/iseries/access/planning.html

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

IBM Redbooks

DB2 UDB for Linux on iSeries: Implementation Guide

(0.2”spine)
0.17”<->0.473”
90<->249 pages

DB2 UDB for Linux on iSeries
Implementation Guide

How to install and configure DB2 UDB for Linux on iSeries

Understanding DB2 Connect and its features

Database access scenarios from Linux applications

This IBM Redbook discusses DB2 UDB for Linux on iSeries. We provide the installation and configuration steps for DB2 UDB for Linux on iSeries. We also present connectivity scenarios between OS/400 resources and Linux resources.

With V8.1, DB2 UDB for Linux is available for running in a Linux partition on an IBM @server iSeries system.

Linux is one of the fastest growing operating systems in the industry today. Linux is quickly becoming a key enabler for e-business applications that are demanding more robust local database capabilities.

Applications running in the Linux partition can certainly leverage DB2 UDB for iSeries through a variety of interfaces (DB2 Connect, JDBC, ODBC), and this is an excellent choice to leverage the low administration overhead, autonomic computing, and security benefits provided by DB2's tight integration with OS/400. DB2 for Linux is an excellent choice to support Linux (or other client/server) applications when there is a need to have local data storage within the Linux application environment or to leverage specific features of DB2 UDB V8 such as Federated Database or Microsoft .NET Framework support.

SG24-6081-00 ISBN 0738498122