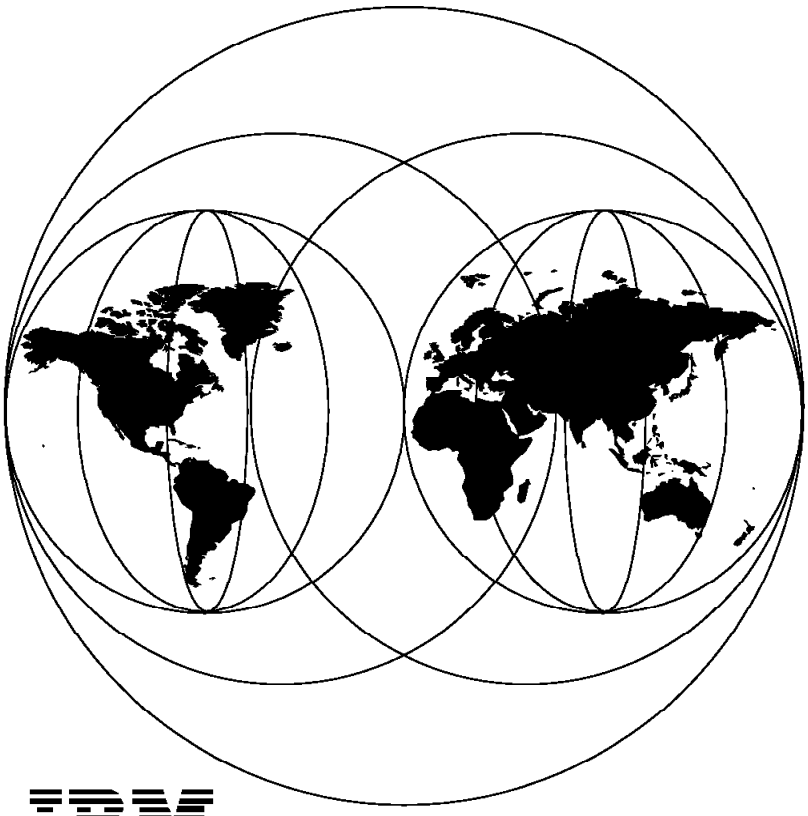
SG24-2006-00

Migrating to DB2 Universal Database Version 5

December 1997



IBM

International Technical Support Organization

Austin Center

International Technical Support Organization

SG24-2006-00

Migrating to DB2 Universal Database Version 5

December 1997

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special Notices” on page 361.

First Edition (December 1997)

This edition applies to DB2 Universal Database Version 5.0 in the Personal, Workgroup, Enterprise and Enterprise - Extended Editions for use with the Windows 95, Windows NT, OS/2, AIX, HP-UX and Solaris Operating Systems where applicable.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 045 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

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

Contents

Figures

Tables

Preface

This redbook is intended to help database administrators plan their migration to DB2 Universal Database from previous versions of DB2. Specifically, it covers migration scenarios on the Windows NT, OS/2 and UNIX platforms including DB2 Parallel Edition. On each platform, a step by step guide to migration is documented.

A detailed explanation of the new features included in DB2 Universal Database is given both to users of DB2 Common Server and DB2 Parallel Edition. An entire chapter is devoted to the GUI Tools which are a significant part of this version of DB2.

Some knowledge of either DB2 Common Server or DB2 Parallel Edition is assumed.

How This Redbook Is Organized

The chapters in this redbook are organized as follows:

- Chapter 1, “DB2 Universal Database Overview” on page 1

  This chapter describes the product packaging in DB2 Universal Database Version 5. It also covers several licensing scenarios.

- Chapter 2, “New Features of DB2 Universal Database Version 5” on page 19

  This chapter gives an overview of the new features included in DB2 Universal Database Version 5. These are the features which are new with respect to DB2 Common Server Version 2.

- Chapter 3, “New Features in DB2 UDB V5 for Users of DB2/PE” on page 79

  This chapter is specifically intended for users of DB2 Parallel Edition Version 1. It gives an overview of the features in DB2 Universal Database which were first introduced in DB2 Common Server.

- Chapter 4, “DB2 Universal Database GUI Tools” on page 137

  This chapter describes the GUI Tools which are included in DB2 Universal Database.

- Chapter 5, “DB2 UDB V5 Migration on OS/2 and Windows NT” on page 199

  This chapter describes the migration process in detail on the Windows NT and OS/2 platforms. Both server and client systems are covered.

- Chapter 6, “DB2 UDB V5 Migration on UNIX” on page 261

  This chapter describes the migration process in detail on UNIX platforms. This includes migrating from a single to a multi-partition database system.

- Chapter 7, “DB2 UDB V5 Migration for DB2/PE Systems” on page 307

  This chapter describes the migration process for users of DB2 Parallel Edition. This includes suggested methods for moving data from SMS to DMS table spaces.

- Appendix A, “db2split.cfg” on page 353

  This appendix details the contents of the db2split configuration file.

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 Austin Center.

This project was designed and managed by:

Jonathan Cook
International Technical Support Organization, Austin Center

The authors of this document are:

Ada Lau Azurza
IBM Peru

Usha Bodalia
IBM United Kingdom

Jonathan Cook
IBM Austin

Richard Hewitt
IBM United Kingdom

Yu-Phing Ong
IBM Australia

Tony Winch
Datasync Consulting, Australia

Takeshi Yoshizawa
IBM Japan

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

Calene Janacek
International Technical Support Organization, Austin Center

Dwaine Snow
IBM Canada

Melanie Stopfer
IBM US Education

My thanks for time and support is also extended to:

Renson Clouden
IBM Canada

Vincent Woo
IBM Canada

Mahesh Garg
IBM Canada

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 373 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com`
  For IBM Intranet users `http://w3.itso.ibm.com`

- Send us a note at the following address:

  `redbook@vnet.ibm.com`

Chapter 1. DB2 Universal Database Overview

The term *universal* in DB2 Universal Database refers to the ability to store all kinds of electronic information. This includes traditional relational data, as well as structured and unstructured binary information, documents and text in many languages, graphics, images, multimedia (audio and video) or information specific to operations like engineering drawings, maps, insurance claims forms, numerical control streams, any type of electronic information.

1.1 DB2 Universal Database Version 5

DB2 Universal Database Version 5 (UDB V5) is actually two successful products merged together: DB2 Parallel Edition V1.2 and DB2 Common Server V2.1.2, as shown in Figure 1. Both were based on DB2/6000 Version 1, but were developed in parallel to address some specific customer needs.

- DB2 Parallel Edition addressed the need to query very large databases running on massively parallel processors and clusters and was developed to optimize parallel query capabilities on IBM's RS/6000 SP hardware.
- DB2 Common Server addressed the needs of the general Structured Query Language (SQL) server market for UNIX, OS/2, and Windows NT.

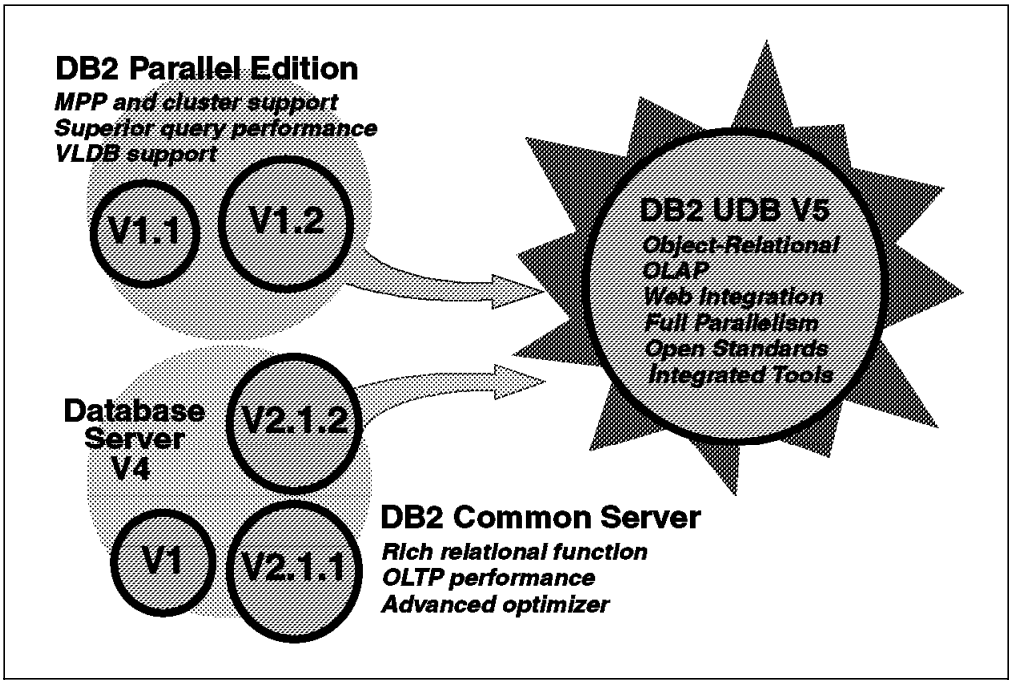


Figure 1. DB2 Universal Database Version 5

The merging of these two products results in a product capable of scaling from laptops to massively parallel systems and of supporting high-volume parallel transactions using the rich function of DB2 Version 2. To this merged function is added additional support for online analytical or Online Analytical Processing (OLAP) processing. There is more web integration in DB2 UDB V5 and Net.Data is packaged with the database server. Also included is more support for industry standards such as greater DCE support, more SQL92 compliance, and integrated tools that include data replication, a governor and a job scheduler.

There is support for previous versions of DB2. The following general rules apply:

- Latest version of the client (version n) will be able to access the previous version of the servers (version n-1).
- Latest version of the server (version n) will support two versions of down-level clients (versions n-1 and n-2).

Version 5 clients can access all common server (Version 2) servers, DDCS Version 2 gateways, and DB2 PE V1 servers.

DB2 UDB V5 servers will support CAE clients from Version 1 and Version 2.

1.2 DB2 Universal Database - The Scalable Database

In Figure 2, all of the DB2 Universal Database server products are shown. DB2 UDB is capable of supporting hardware platforms from laptops to massively parallel systems with hundreds of nodes. This provides extensive and granular growth. There are four different DB2 database server products in Version 5. This group of database servers and related products are the main subject of this book.

For simplicity, we will refer to the DB2 Universal database family of products as DB2.

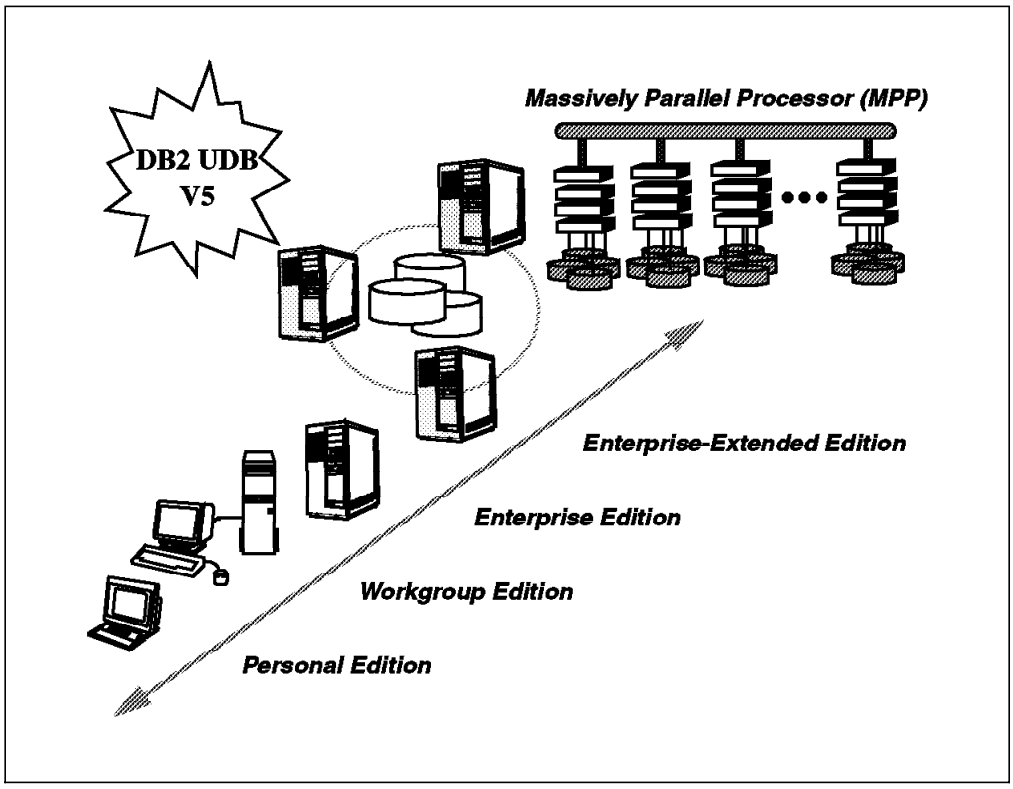


Figure 2. DB2 Universal Database Server Products

1.2.1 DB2 Products

All DB2 products require licensing. There are three main DB2 products, including:

- DB2 Universal Database (UDB) - There are four versions of the DB2 Universal Database, as shown in Figure 2 on page 2: DB2 Personal Edition, DB2 Workgroup Edition, DB2 Enterprise Edition, and DB2 Enterprise-Extended Edition.
- DB2 Connect - Provides the ability to access a host database with DRDA. There are two versions of the product: DB2 Connect Personal Edition and DB2 Connect Enterprise Edition.
- DB2 Developer's Edition - Provides the ability to develop and test a database application for one user. There are two versions of the product: DB2 Personal Developer's Edition (PDE) and DB2 Universal Developer's Edition (UDE).

All DB2 products have a common component called the DB2 Client Application Enabler (CAE). Once a DB2 application has been developed, the DB2 Client Application Enabler component must be installed on each workstation executing the application. Figure 3 shows the relationship between the application, CAE and the DB2 database server. If the application and database are installed on the same workstation, the application is known as a local client. If the application is installed on a workstation other than the DB2 server, the application is known as remote clients.

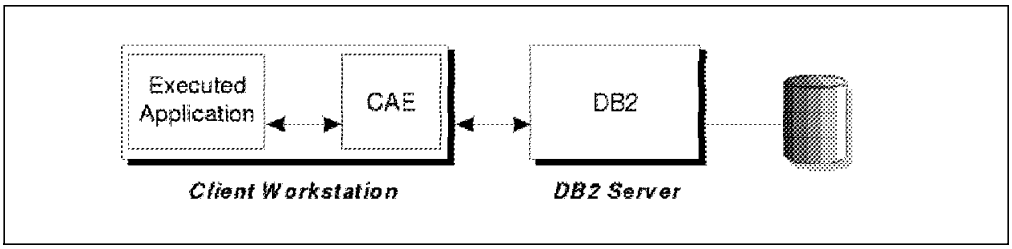


Figure 3. Remote Client Accessing DB2 Server Using CAE

The Client Application Enabler provides functions other than the ability to communicate with a DB2 UDB server or DB2 Connect gateway machine. From the CAE, you can do any of the following:

- Issue an interactive SQL statement using the CAE on a remote client to access data on a remote UDB server.
- Graphically administer and monitor a UDB database server.
- Run applications that were developed to comply with the Open Database Connectivity (ODBC) standard.
- Run Java applications that access and manipulate data in DB2 UDB databases using Java Database Connectivity (JDBC).

There are no licensing requirements to install the Client Application Enabler (CAE) component. Licensing is controlled at the DB2 UDB server.

The CAE installation depends on the operating system on the client machine. For example, if you have a database application developed for AIX, you will need to install the Client Application Enabler for AIX.

There is a different CAE for each supported DB2 client operating system. The supported platforms are OS/2, Windows NT, Windows 95, Windows 3.x, AIX, HP-UX, and Solaris. The CAE component should be installed on all end-user workstations.

A complete set of CAEs is provided with Workgroup Edition, Enterprise Edition, Enterprise-Extended Edition and the DB2 Connect Enterprise products. This set of CAEs is provided on a CD-ROM and it is referred to as the DB2 *CAE Client Pack*.

1.2.2 DB2 Universal Database

DB2 Universal Database, or UDB, is a Relational Database Management System (RDBMS). It can be included in one of the following four database server products:

- DB2 Personal Edition

  An RDBMS engine that will not accept incoming database requests. (Available on the Intel platform only.)

- DB2 Workgroup Edition

  An RDBMS engine that will accept incoming database requests. (Available on the Intel platform only.)

- DB2 Enterprise Edition

  Similar to Workgroup Edition, but also allows remote client applications to access data on a host database.

- DB2 Enterprise - Extended Edition

  Similar to the Enterprise Edition, with additional support for clusters of database servers in a partitioned database environment.

1.2.2.1 DB2 UDB Personal Edition

DB2 UDB Personal Edition provides the same engine functions found in Workgroup, Enterprise and Enterprise-Extended Editions. However, DB2 Personal Edition cannot accept requests from a remote client.

As the product name suggests, DB2 Personal Edition is licensed for one user to create databases on the workstation in which it was installed. DB2 Personal Edition can be used as a remote client to a DB2 UDB server where either Workgroup, Enterprise or Enterprise-Extended is installed, since it contains the Client Application Enabler (CAE) component. Therefore, once the DB2 Personal Edition product has been installed, you can use this workstation as a remote client connecting to a DB2 Server, as well as a DB2 Server managing local databases.

The DB2 Personal Edition product may be appropriate for the following users:

- DB2 mobile users who use a local database and can take advantage of the replication feature in UDB to copy local changes to a remote server
- DB2 end-users requiring access to local and remote databases

Figure 4 on page 5 shows an example of DB2 Personal Edition installation. In this example, the user can access a local database on their mobile workstation (laptop) and access remote databases found on the database server. From the

laptop, the user can make changes to the database throughout the day and replicate those changes as a remote client to a UDB remote server.

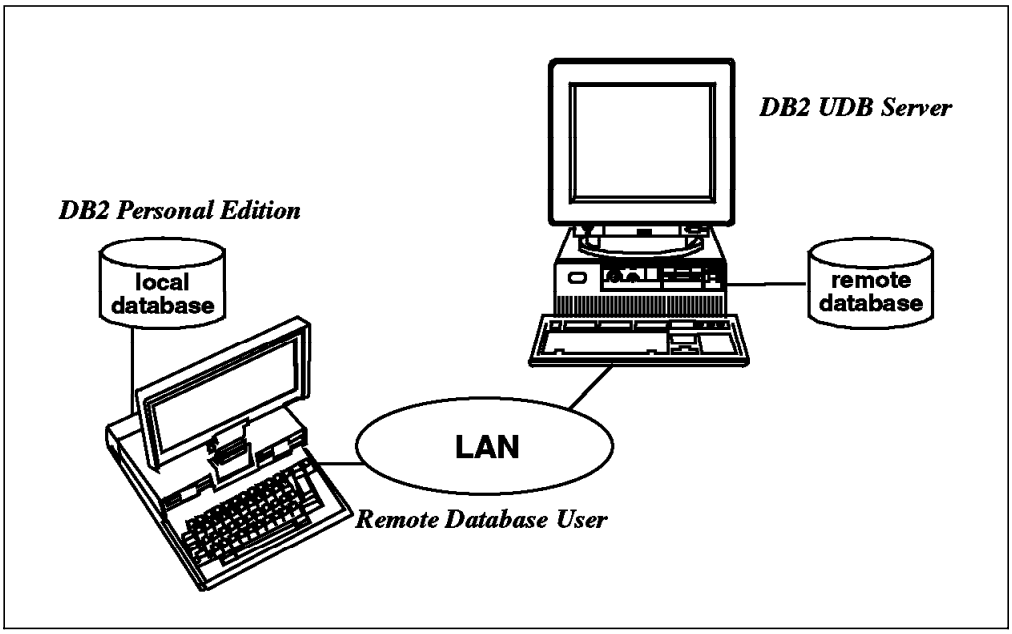


Figure 4. DB2 UDB Personal Edition

DB2 Personal Edition includes graphical tools that enable you to administer, tune for performance, access remote DB2 servers, process SQL queries, and manage other servers from a single workstation.

This product is available for the OS/2, Windows NT, and Windows 95 operating systems. DB2 Personal Edition is licensed for a single user.

1.2.2.2 DB2 Workgroup Edition

DB2 Workgroup contains all DB2 Personal Edition product functions with the added ability to accept requests from remote clients. Like DB2 Personal Edition, DB2 Workgroup Edition is for the Intel platform only. In Figure 4, DB2 Personal Edition is shown as a mobile user that occasionally connects to a LAN. This mobile user can access any of the databases on the workstation where DB2 Workgroup Edition is installed.

DB2 Workgroup Edition is designed for use in a LAN environment. It provides support for both remote and local clients. A workstation with Workgroup Edition installed can be connected to a network and participate in a client/server environment as shown in Figure 5 on page 6.

In Figure 5 on page 6, *application 1* and *application 2* are local database applications. Remote clients can also execute *application 1* and *application 2*, if the proper client/server setup has been performed. A DB2 application does not contain any specific information regarding the physical location of the database. DB2 client applications communicate with DB2 Workgroup Edition using a client/server—supported protocol with the DB2 CAE. Depending on the client and server operating system involved, DB2 Workgroup supports the following protocols: TCP/IP, NetBIOS, IPX/SPX, Named Pipes and APPC.

DB2 Workgroup includes Net.Data for Internet support, as well as the graphical management tools that are also found in DB2 Personal Edition. In addition, the

DB2 Client Pack is shipped with DB2 Workgroup Edition. The DB2 Client Pack contains all of the current DB2 Client Application Enablers.

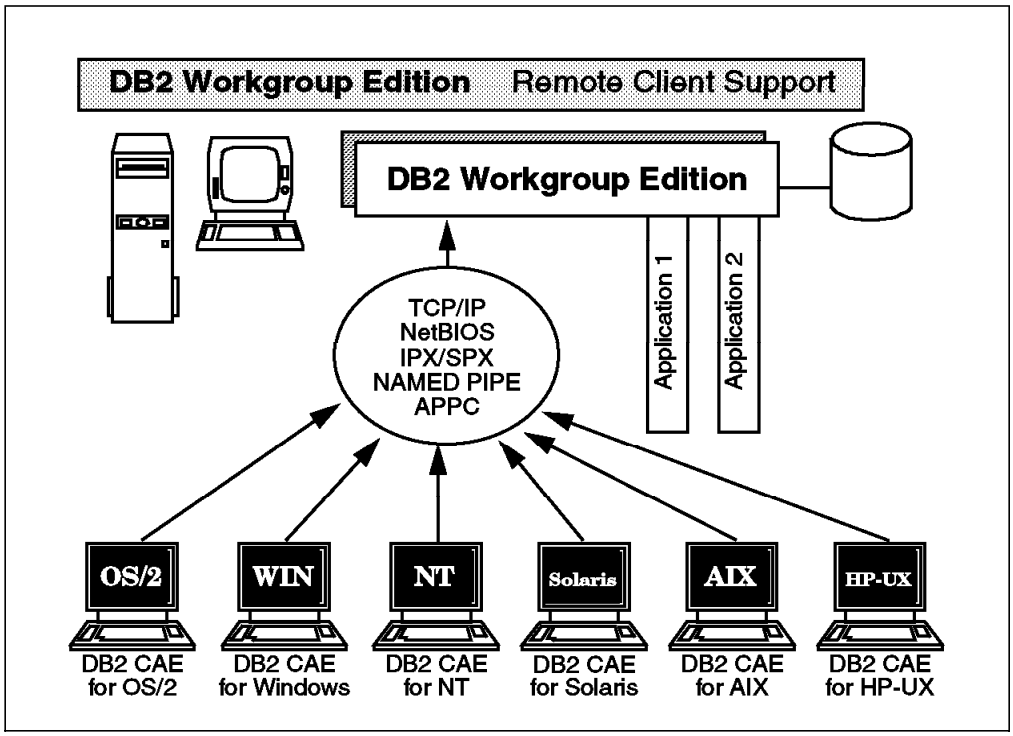


Figure 5. DB2 Workgroup Edition with Remote Clients

DB2 Workgroup Edition is licensed on a per-user basis. The base license is for one concurrent or registered DB2 user. It is available for OS/2 and Windows NT. Additional entitlements (user licenses) are available in multiples of 1, 5, 10 or 50. DB2 Workgroup is licensed for a machine with one-to-four processors only.

Entitlement keys are required for additional users.

1.2.2.3 DB2 Enterprise Edition

DB2 UDB Enterprise Edition includes all of the function provided in the Workgroup Edition, plus support for host database connectivity. It allows access to DB2 databases residing on host systems such as DB2 for OS/390 Version 5.1, DB2 for OS/400, and DB2 Server for VSE/VM Version 5.1. The licensing for DB2 Enterprise Edition is based on the number of users, the number of machines installed and the processor type. The base license is for one concurrent or registered DB2 user. Additional entitlements are available for 1, 5, 10 or 50 users. The base installation of DB2 Enterprise is on a uni-processor. Tier upgrades are available if the machine has more than one processor. With the first tier upgrade, you receive the rights to a free entitlement for 50 users.

The licensing for the gateway capability for access to a host database is for 30 concurrent or registered DB2 users.

DB2 Enterprise Edition is available on OS/2, Windows NT, and the following UNIX platforms: AIX, HP-UX, and Solaris.

- **Sample Scenario Using DB2 Enterprise Edition**

The popularity of the Internet and the World Wide Web has created a demand for web access to enterprise data. The DB2 UDB server product includes all supported DB2 Net.Data products.

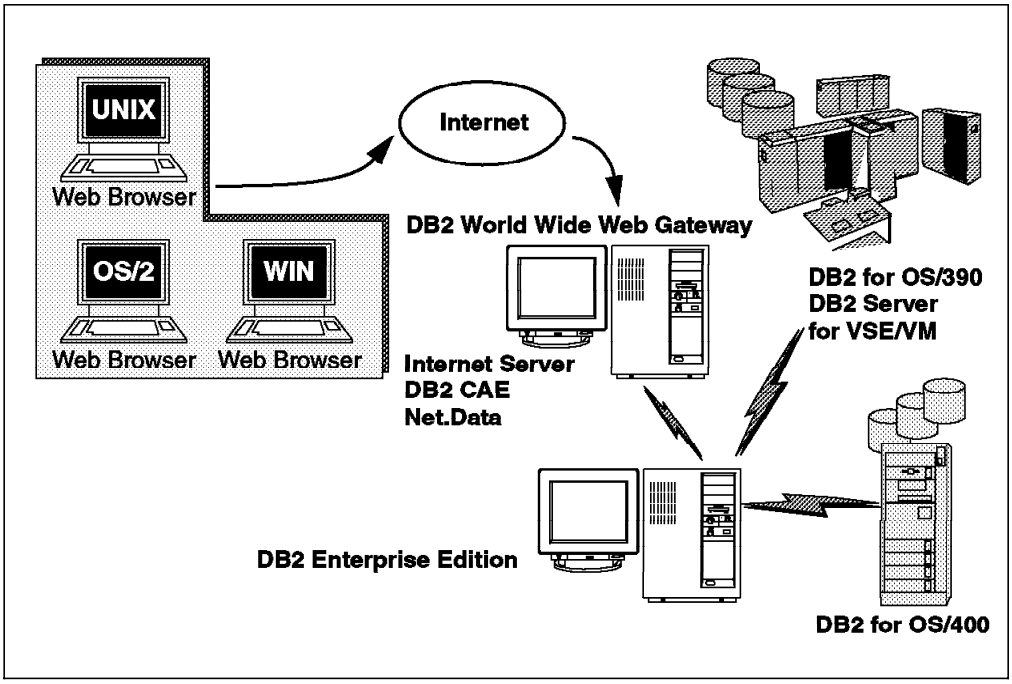


Figure 6. Accessing Data via the Internet

Applications that are built with Net.Data may be stored on a web server and can be viewed from any web browser because they are written in HTML (hypertext markup language). While viewing these documents, users can either select automated queries or define new ones that retrieve specified information directly from a DB2 UDB database. Figure 6 illustrates how DB2 Enterprise Edition acts as the database server and the gateway to access data stored in DB2 for OS/400, DB2 for OS/390 and DB2 Server for VSE/VM. The ability to connect to a host database (DB2 Connect) is built into Enterprise Edition.

1.2.2.4 DB2 Universal Database Enterprise-Extended Edition

This product contains all the features and functions of DB2 Enterprise Edition. It also provides the ability for a database to be partitioned across multiple independent computers of the same platform. To the end-user or application developer, the database appears to be on a single computer. While DB2 Workgroup and DB2 Enterprise Edition can handle large databases, the Enterprise-Extended Edition (EEE) is designed for applications where the database is simply too large for a single computer to handle efficiently. SQL operations can operate in parallel on the individual database partitions, thus increasing the execution of a single query.

- **Sample Scenario Using DB2 Enterprise - Extended Edition**

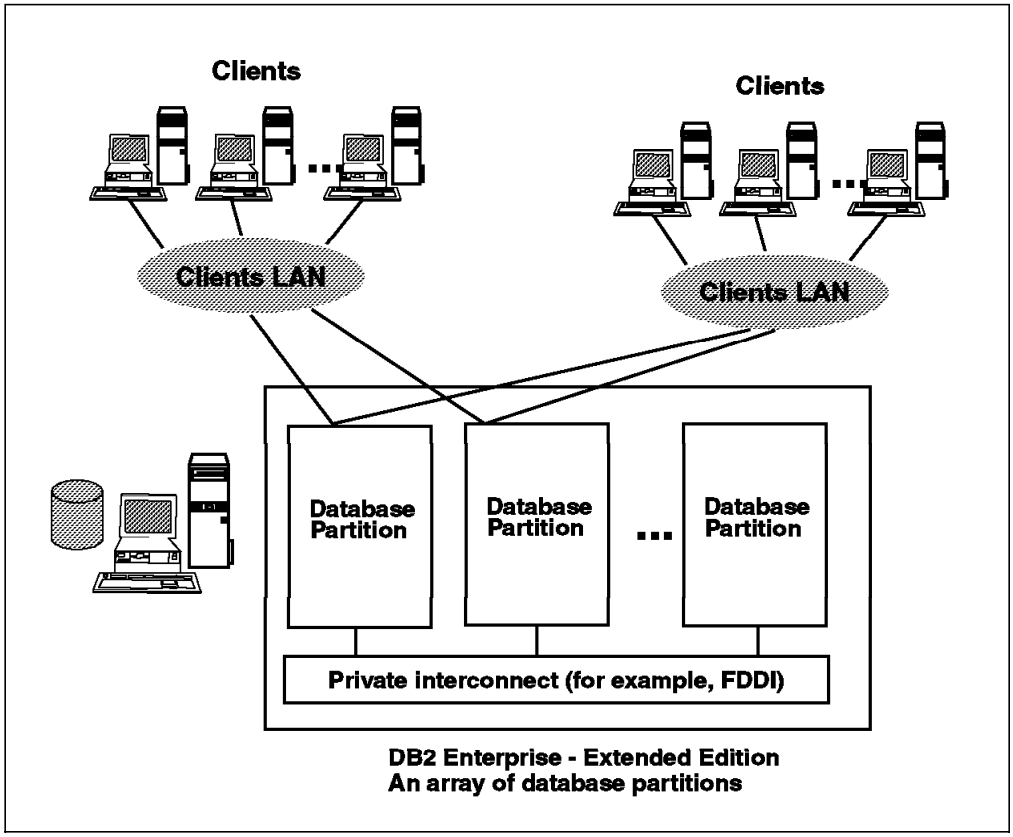


Figure 7. DB2 Enterprise - Extended Edition

DB2 Enterprise-Extended Edition licensing is similar to that of DB2 Enterprise Edition. However, the licensing is based on the number of registered or concurrent users, the type of processor and the number of database partitions. You must have a license for each database partition (node) in your configuration. The base license for DB2 Enterprise-Extended Edition is for machines ranging from a uni-processor up to a 4-way SMP. The base number of users is different in Enterprise-Extended Edition than in Enterprise Edition. The base user license is for one user with an additional 50 users, equaling 51 users for that database partition. The total number of users per database partition also depends on the total number of database partitions. For example, in a system configuration of four nodes, each node or database partition could support 51 X 4, or 204 users. Tier upgrades also are available. The first tier upgrade for a database partition provides the rights to a 50 user entitlement pack for that database partition or node. Additional user entitlements are available for 1, 5, 10 or 50 users.

The licensing for the gateway is the same as that found in Enterprise Edition for 30 users. DB2 Enterprise-Extended Edition (EEE) is currently available on the AIX platform.

1.2.2.5 DB2 Connect

The DB2 Connect product allows clients to access data stored on database servers that implement the Distributed Relational Database Architecture (DRDA). The target database server for a DB2 Connect installation is known as a DRDA Application Server. Figure 8 on page 9 shows the flow of database requests through DB2 Connect.

The most commonly accessed DRDA application server is DB2 for OS/390.

DB2 Connect supports the APPC communication protocol to provide communications support between DRDA Application Servers and DRDA Application Requesters. Also, DB2 for OS/390 Version 5.1 supports TCP/IP in a DRDA environment. Any of the supported network protocols can be used for the DB2 client (CAE) to establish a connection to the DB2 Connect gateway.

The database application must request the data from a DRDA Application Server through a DRDA Application Requester. The DB2 CAE component is not a DRDA Application Requester.

The DB2 Connect product provides DRDA Application Requester functionality.

The DRDA Application Server accessed using DB2 Connect could be any of the following DB2 Servers:

- DB2 for OS/390. Only Version 5.1 or higher supports TCP/IP in a DRDA environment.
- DB2 for OS/400
- DB2 Server for VSE/VM

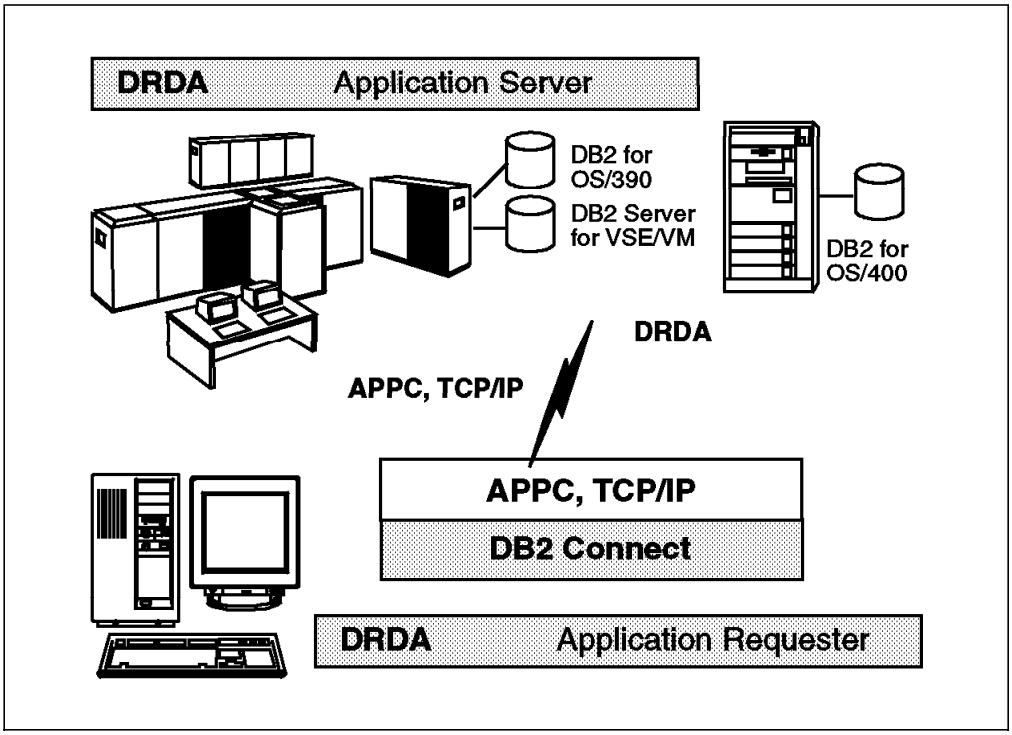


Figure 8. DRDA Application Flow

1.2.2.6 DB2 Connect Personal Edition

DB2 Connect Personal Edition is available only on the Intel platform: OS/2, Windows NT, Windows 95 and Windows 3.x. It provides access to host databases from the workstation where it is installed. Figure 9 on page 10 shows the DRDA flow between the Application Requester and the Application Server using the DB2 Connect Personal Edition product.

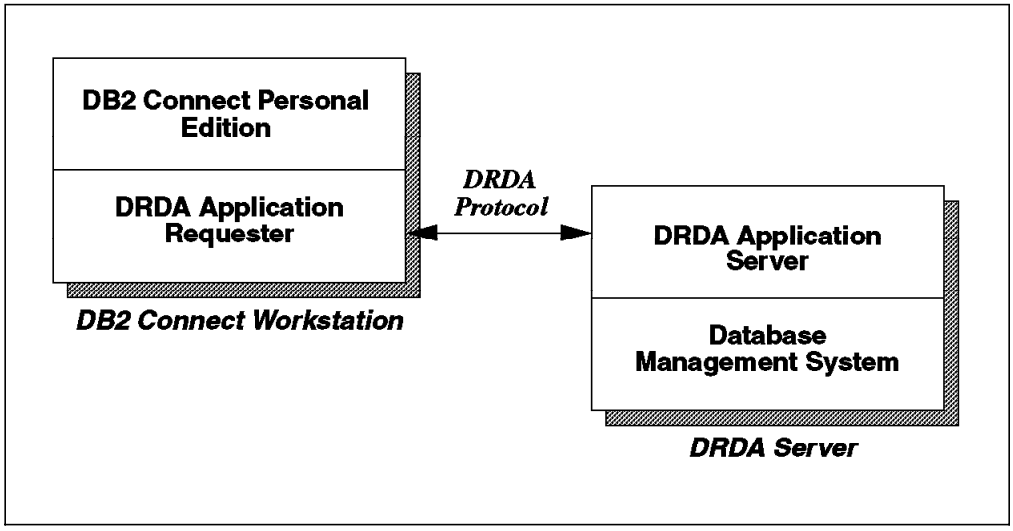


Figure 9. DRDA Flow in DB2 Connect Personal Edition

1.2.2.7 DB2 Connect Enterprise Edition

The DB2 Connect Enterprise Edition product provides the ability for multiple clients to access host data. A DB2 Connect gateway routes each database request from the DB2 clients to the appropriate DRDA Application Server database. Figure 10 shows the addition of a remote client. The remote client communicates with the DB2 Connect workstation using any of the supported communication protocols.

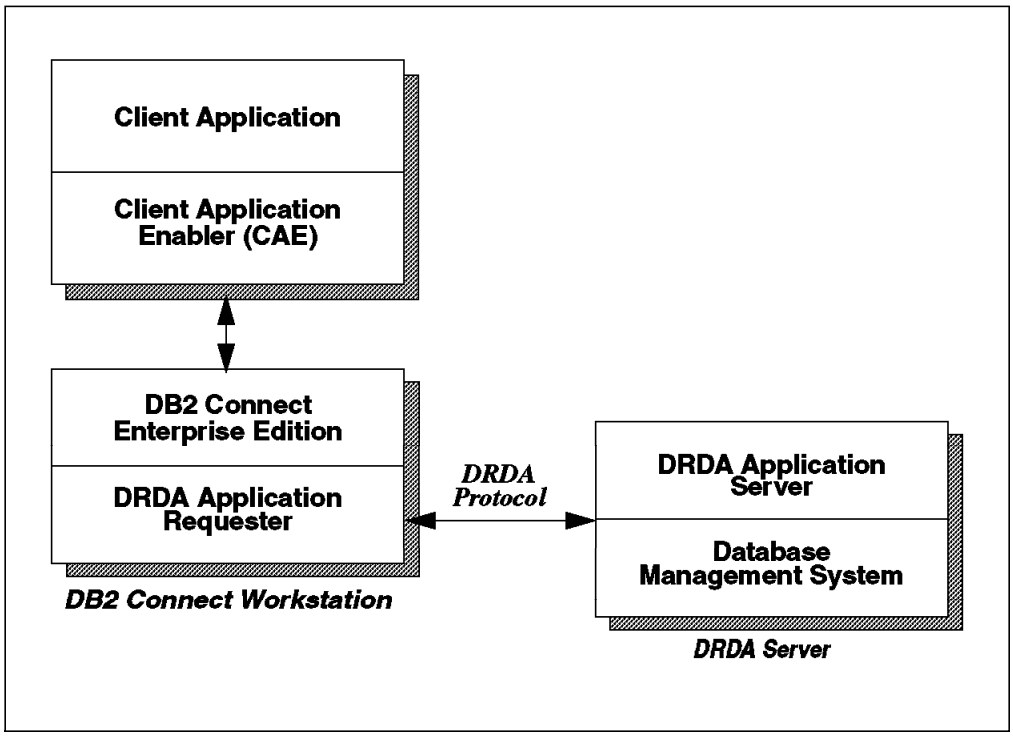


Figure 10. DRDA Flow in DB2 Connect Enterprise Edition

The licensing for DB2 Connect Enterprise is user-based. That is, it is licensed on the number of concurrent or registered users. The base license is for one user

with additional entitlements of 1, 5, 10 or 50 users. DB2 Connect Enterprise Edition is supported on OS/2, Windows NT, AIX, HP-UX and Solaris.

1.2.2.8 DB2 Developer's Edition

The DB2 Developer's Edition is a separate product that can be installed on either the DB2 server or on a DB2 client. It provides all of the necessary data access tools for developing SQL applications. It is available in DB2 Personal Developer's (PDE) Edition and DB2 Universal Developer's Edition (UDE). DB2 Personal Developer's Edition is available only for the Intel platforms. DB2 Universal Developer's Edition is available for all server platforms except Enterprise-Extended Edition.

The application development environment provided with both versions of the Developer's Edition allows application developers to write programs using the following methods:

- Embedded SQL
- Call Level Interface or CLI (compatible with the Microsoft ODBC standard)
- DB2 Application Programming Interfaces (APIs)
- DB2 data access through the World Wide Web

The programming environment also includes the necessary programming libraries, header files, code samples and precompilers for the supported programming languages. Several programming languages, including COBOL, FORTRAN, REXX, C and C++, Basic and Java are supported by DB2.

An application developed with the SDK can be executed on any workstation with the same operating system that has the CAE installed. To run the application on another operating system requires the application to be re-built using the SDK on the target operating system.

Figure 11 on page 12 shows the contents of both the DB2 Universal Developer's Edition and the DB2 Personal Developer's Edition. Both products contain the following:

- Software Developer's Kit (SDK) - Provides the environment and tools you need to develop applications that access DB2 databases using embedded SQL or DB2 Call Level Interface (CLI). This is found in both PDE and UDE. However, the SDK in the PDE is for OS/2, Windows 16-bit and 32-bit 3.x, Windows 95 and Windows NT. The SDK in the UDE is for all platforms.
- Extender Support - Provides the ability to define large object data types, and includes related functions that allow your applications to access and retrieve documents, photographs, music, movie clips or fingerprints.
- Visual Age for Basic - A suite of application development tools built around an advanced implementation of the Basic programming language, to create GUI clients, DB2 stored procedures, and DB2 user-defined functions. This is found in both the PDE and the UDE.
- Visual Age for Java - A suite of application development tools for building Java-compatible applications, applets, and JavaBean components that run on any Java Development Kit (JDK) enabled browser. It contains Enterprise Access Builder for building JDBC interfaces to data managed by DB2. It can also be used to create DB2 stored procedures and DB2 user-defined functions.

- Net.Data - A comprehensive World Wide Web (WWW) development environment to create dynamic web pages or complex web-based applications that can access DB2 databases. Net.Data is only available in the UDE product.
- Lotus Approach - Provides a graphical interface to perform queries, develop reports and analyze data. You can also develop applications using LotusScript, a full-featured, object-oriented programming language. This is found in both the PDE and the UDE products.
- Domino Go WebServer - A scalable, high-performance web server that runs on a broad range of platforms. It offers the latest in web security and supports key internet standards. This is found only in the UDE product.
- ODBC and JDBC support are found in PDE and UDE. versions of the Developer's Edition.

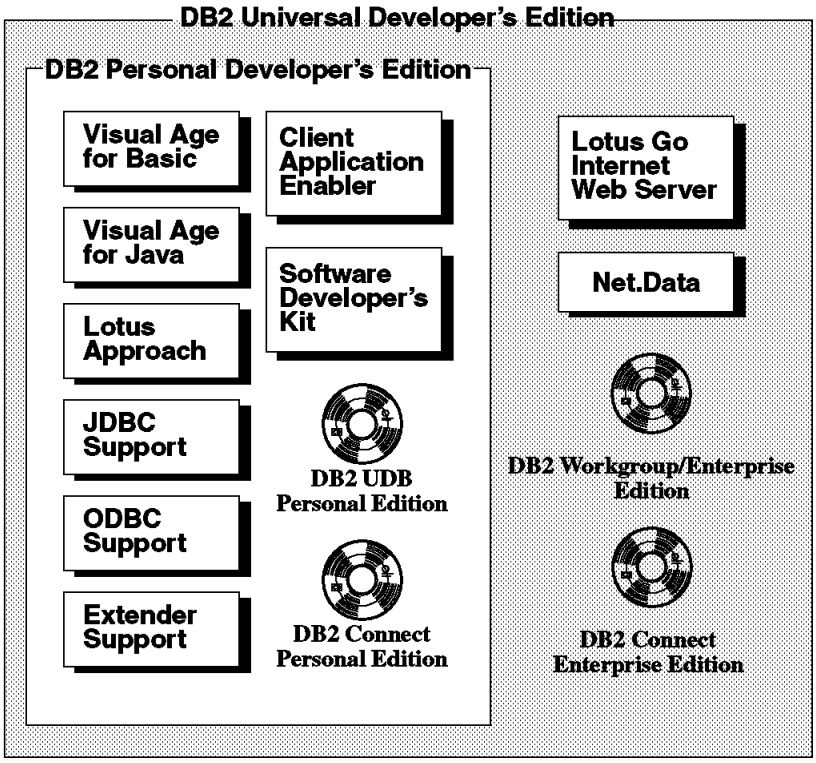


Figure 11. DB2 Personal Developer's and Universal Developer's Edition

Figure 11 shows that the UDB Server and Connect products are part of the Developer's Edition. The DB2 Personal Developer's Edition contains DB2 UDB Personal Edition and DB2 Connect Personal Edition. This allows a single application developer to develop and test a database application. DB2 Personal Developer's Edition is a single-user product available for OS/2, Windows 3.1, Windows NT and Windows 95.

DB2 Universal Developer's Edition is supported on all platforms that support the DB2 Universal Database server product, except for the Enterprise-Extended Edition product or partitioned database environment. DB2 Universal Developer's Edition is intended for application development and testing only. The database server can be on a platform that is different from the platform on which the application is developed. It contains the DB2 UDB Personal Edition, Workgroup and Enterprise Editions of the database server product. Also, DB2 Connect

Personal and Enterprise Edition are found in the UDE product. The UDE is licensed for one user. Additional entitlements are available for 1,5 or 10 concurrent or registered DB2 users.

1.2.3 Licensing Scenarios

The DB2 products use a licensing mechanism to control the usage of DB2 software. A program license key is provided with all DB2 products. This product license key is provided in a file on the install media (CD-ROM) and is automatically copied and set up during the installation.

As part of the installation and setup of a DB2 product, the product license information is installed on your system in a special license file called the nodelock file. Any changes or upgrades that you purchase must be entered in the nodelock file. Table 1 details the licensing for Version 5 DB2 products.

Product Name	Base License	Base Processor	Processor Upgrade	Add'l Installs	Add'l Users	Supported Platforms
UDB Personal Edition	1 user	Any	N/A	1,5,10,50	N/A	OS/2 Windows NT Windows 95
UDB Workgroup Edition	1 user	Uni to 4-way	N/A	1,5,10	1,5,10,50	OS/2 Windows NT
UDB Enterprise Edition	1 user [2,3]	Uni	1 to 4,8,16,24 4 to 8,16,24 8 to 16,24 16 to 24	1,5,10	1,5,10,50	Windows NT, OS/2, AIX, HP-UX, Solaris
UDB Enterprise-Extended Edition	1 user + 50 [1,2,3,5]	Uni to 4-way	4 to 8,16,24 [4] 8 to 16,24 16 to 24	1,5,10	1,5,10,50	AIX
Connect Personal Edition	1 user	Any	N/A	1,5,10,50	N/A	OS/2,Win 3.x, Windows NT Windows 95
Connect Enterprise Edition	1 user	Any	N/A	1,5,10	1,5,10,50	Windows NT, Windows 95, OS/2, AIX, HP-UX, Solaris
Personal Developer's Edition	1 user	Any	N/A	1,5,10	N/A	OS/2 Windows NT Windows 95
Universal Developer's Edition	1 user	Any	N/A	1,5,10	N/A	Windows NT, Windows 95, OS/2, AIX, HP-UX, Solaris

1 - Must license per database partition or node.
2 - Includes rights for 30 users for DB2 Connect Enterprise.
3 - First processor upgrade includes rights for 50 additional users per node.
4 - Processor upgrade is for single node. Must purchase Server Authorizations of 1, 5, or 10.
5 - Base license includes rights for 50 additional users. Users are cumulative. ((1 + 50) X number of nodes).

Table 1. Licensing Summary for DB2 UDB V5 Products

Each DB2 Version 5 product is a single-user product (with the exception of Enterprise-Extended Edition). There are three ways you can change your base license:

- Change the number of concurrent or registered DB2 users that can access the DB2 product installed on your system. This is referred to in Table 1 as Additional (Add'l) Users. User Entitlements or additional user licenses are

available in quantities of 1, 5, 10 or 50. Four of the DB2 products are single-user license products only: DB2 UDB Personal Edition, DB2 Connect Personal Edition, DB2 Personal Developer's Edition and DB2 Universal Developer's Edition.

DB2 Enterprise Edition and Enterprise-Extended Edition include the functions of DB2 Connect Enterprise. The base user license for DB2 Connect Enterprise function is for 30 users.

- Change the number of systems that have the DB2 product installed. This is referred to in Table 1 on page 13 as Additional (Add'l) Installs. This licensing feature is also referred to as Server Installs. Server Installs is the ability to install the DB2 product on another machine. If you are in a database-partitioned environment, for example the DB2 UDB Enterprise-Extended Edition, you can install the DB2 product on another node. Server Installs or additional installs are available in quanities of 1, 5, 10, or 50.
- Change the processor type (Processor Upgrade). If you upgrade your processor, you need to change your base processor license. This applies only to DB2 UDB Enterprise Edition, and DB2 UDB Enterprise-Extended Edition. The base processor license for Workgroup and Enterprise-Extended Editions is for either a uni-processor or 4-way SMP (Symmetric Multi-Processor) system.

  This upgrade is also referred to as a tier upgrade. The first tier upgrade entitles you to 50 additional concurrent or registered DB2 users on your system. This is only for DB2 Enterprise Edition and DB2 Enterprise-Extended Edition products. However, for DB2 Enterprise-Extended Edition, this applies to only one database partition. If upgrading multiple database partitions or nodes, you need additional Server Authorizations. Server Authorizations for DB2 Enterprise-Extended Edition are available for 1, 5 or 10 database partitions.
- You must license for the number of CPUs available on your system. For example, if you are using an 8-way SMP machine, you must purchase the tier upgrade from a uni-processor to an 8-way SMP.

Let's consider three scenarios in which we will change the number of users, the number of times we install the product, and the type of processor.

- **Changing the Number of Users**

In this first example, we will be increasing the number of users.

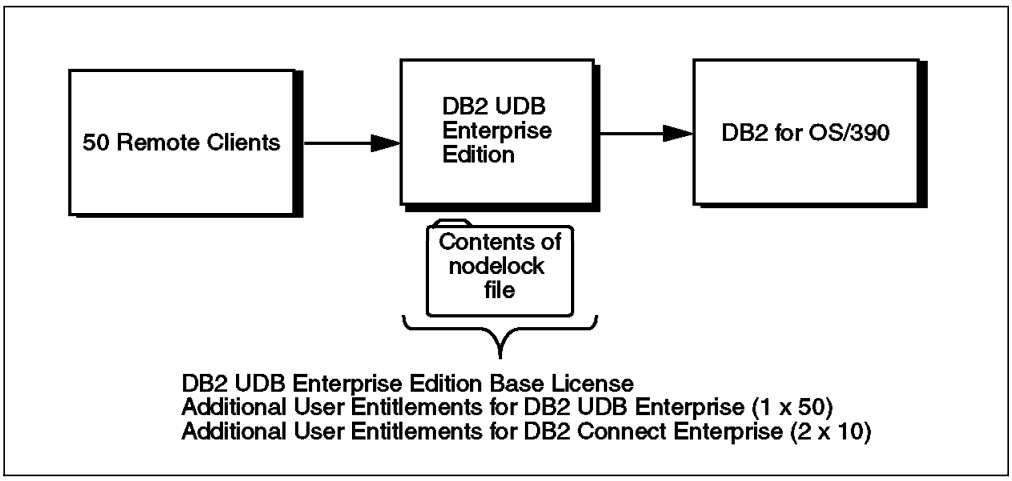


Figure 12. Increasing the Number of Users

Figure 12 shows a database server with the DB2 UDB Enterprise Edition product. There will be 50 remote clients that will need access to both the DB2 UDB server and a DB2 for OS/390 host server. The base license of DB2 UDB Enterprise allows for one concurrent user for the database and 30 users for the host database (DB2 Connect). The easiest solution for the database server is the Additional User Entitlements for 50. This would bring the number of users to 51. Additional User Entitlements are needed for DB2 Connect capability. An addition of 2 X 10 entitlements would bring the total number of DB2 Connect users to 50.

- **Changing the Number of Installs**

This example illustrates adding another application developer to your current environment.

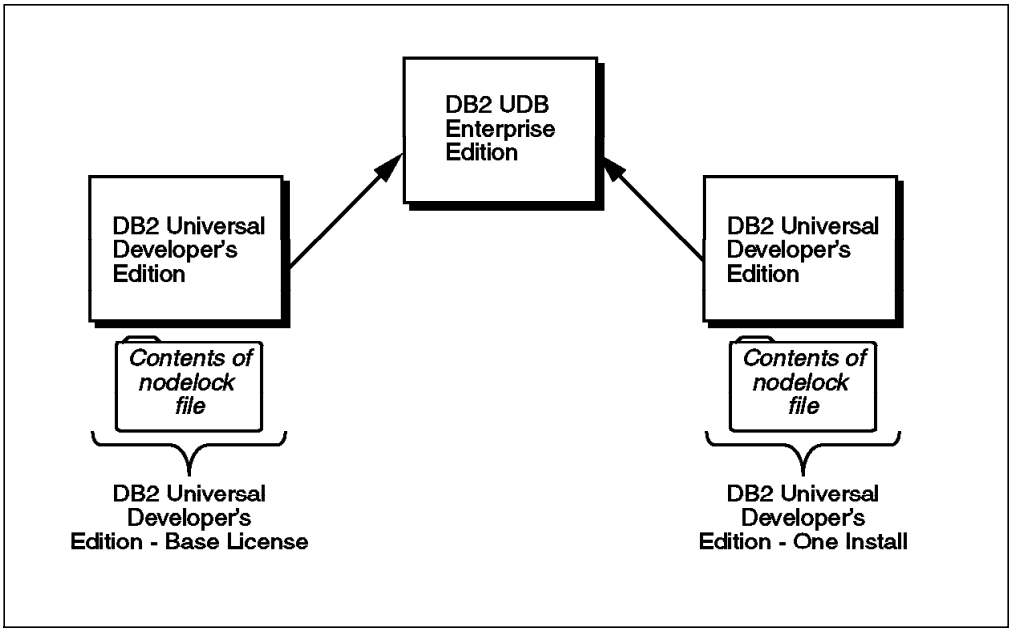


Figure 13. Increasing the Number of Installs

Figure 13 shows the addition of an application development environment. DB2 Universal Developers Edition is installed on a remote client. The database server has the DB2 UDB Enterprise Edition installed. This will allow one application developer to develop and test against the UDB server. Another

application developer needs an Additional Install of DB2 Universal Developer's Edition. This will allow you to develop and test a different client platform. The database server may or may not be on the same platform originally installed.

- **Changing the Number of Processors**

The last example in this chapter is shown as follows. DB2 Enterprise-Extended Edition (EEE) is the database product on four database partitions. The number of users is not discussed in this example.

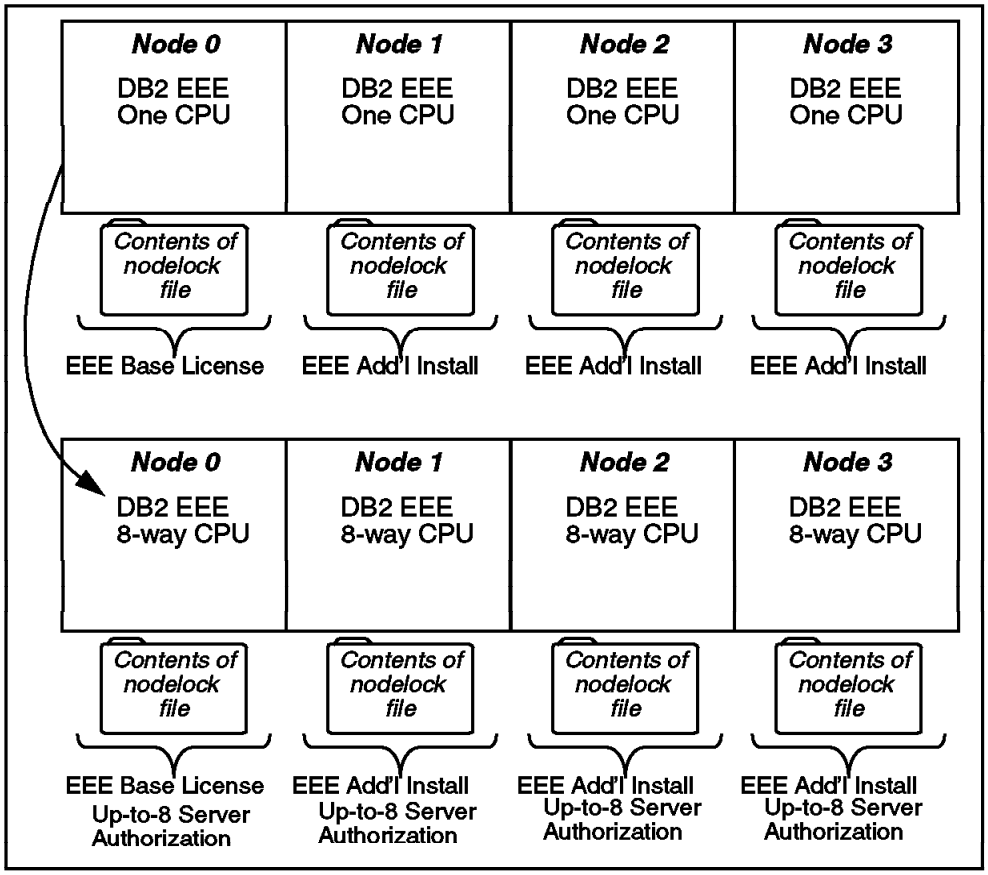


Figure 14. Increasing the Number of Processors

Figure 14 shows an installation of DB2 Enterprise-Extended Edition on four database partitions or nodes. Each database partition was installed on one CPU for 20 concurrent or registered DB2 users. Three Additional Install licenses were purchased for Node 1, Node 2 and Node 3. The processor on each system was upgraded from a uni-processor to an eight-way SMP. The additional licensing requirement was a processor Server Entitlement increase of Up-to-8 from Up-to-4. The base license was for a uni- to 4 way processor. You could either purchase 4 separate authorizations or a 5 server authorization. The number of users was not discussed. Each node or database partition in your configuration is a one-user license with an additional 50 users for that partition, multiplied by the number of database partitions. In our example, there are 4 nodes. Each node has a possible 51 users x 4 nodes, yielding a total of 204 users per node or database partition.

1.3 Migration to DB2 UDB Version 5

This section gives an introduction to DB2 UDB Version 5 migration. Step by step guidelines for the supported migration scenarios can be found in Chapters 5, 6 and 7.

1.3.1 Supported Migration Scenarios

Figure 15 illustrates the possible migration scenarios to DB2 Universal Database Version 5. From DB2/6000 or DB2/2 Version 1, you can migrate to DB2 UDB, on a single node or single node cluster. The single node cluster would be used if your hardware configuration had changed from a single machine to either a cluster or SP environment. Once the database and directories had been migrated to a single node cluster, you need to add nodegroups and redistribute the data in the database to the nodegroup.

From a DB2 Common Server V2.x, the same possible migration scenarios exist, migrating to DB2 UDB V5 either in a serial environment or single partition cluster.

DB2 Parallel Edition for AIX V1.x databases have only one migration path to a DB2 UDB V5 multi-partition system. The DB2 UDB V5 product in a multi-partition environment is called Enterprise-Extended Edition or EEE.

Note: DB2 UDB V5 does NOT support cross-platform migration, (for example OS/2 to NT).

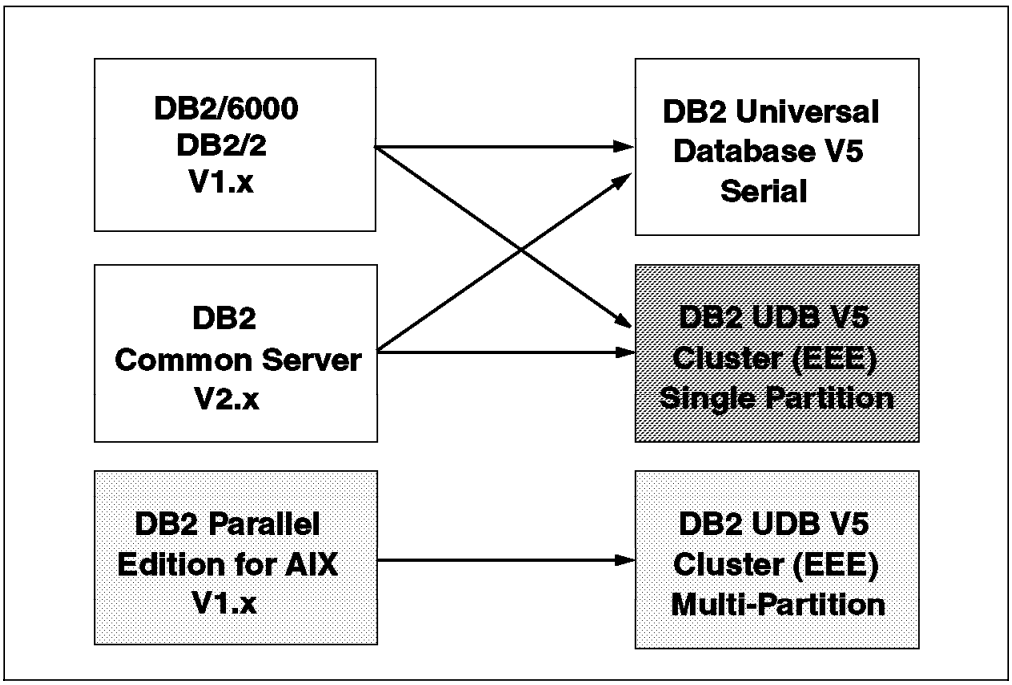


Figure 15. Supported Migration Scenarios

1.3.2 Migration Scenarios

- For a step by step guide to migrating to UDB for:
  - DB2 Common Server V2 for Windows NT systems
  - DB2 Common Server V2 for OS/2 systems
  - DB2/2 V1 systems

  refer to Chapter 5, “DB2 UDB V5 Migration on OS/2 and Windows NT” on page 199.
- For a step by step guide to migrating to UDB for:
  - DB2 Common Server V2 for AIX
  - DB2 Common Server V2 for HP-UX
  - DB2 Common Server V2 for Solaris
  - DB2/6000 V1

  refer to Chapter 6, “DB2 UDB V5 Migration on UNIX” on page 261.
- For a step by step guide to migrating to UDB for:
  - DB2 Parallel Edition V1

  refer to Chapter 7, “DB2 UDB V5 Migration for DB2/PE Systems” on page 307.

Chapter 2. New Features of DB2 Universal Database Version 5

DB2 Universal Database V5.0 offers many new features and enhancements. It also incorporates features from its predecessors DB2 Common Server V2 and DB2 Parallel Edition V1.2.

This chapter gives an overview of the new functionality and a detailed description of selected UDB V5 features. Since the focus of these book is on migration, we will concentrate on those new features and enhancements that are relevant to migration. For information on features that were introduced in DB2 Common Server V2 and which are new to users of DB2 Parallel Edition, see Chapter 3, “New Features in DB2 UDB V5 for Users of DB2/PE” on page 79.

We also include features and enhancements that are not directly related to the base code and data migration. They are included because we regard them as major improvements in terms of performance and availability, therefore they should be taken into consideration for any migration.

This chapter is comprised of the following sections:

- 2.1, “Large Database Support,” including:
  - Buffer Pools, and Extended Storage
- 2.2, “Utilities Enhancements” on page 25, including:
  - Import, Load, Backup and Recovery
- 2.3, “Parallelism and SMP Enablement” on page 40, including:
  - Inter and Intra-Query Parallelism, Utilities Parallelism
- 2.4, “Performance Features and Enhancements” on page 52, including
  - Dynamic Bitmap Index Anding, OLAP SQL, and Outer Join
- 2.5, “DB2 UDB V5 Environment” on page 70, including:
  - Adminstration Server, and Profile Registries

2.1 Large Database Support

Two new features are introduced in DB2 UDB V5 to support large databases, namely the support for multiple buffer pools and the enablement of areas of memory known as extended storage.

2.1.1 Multiple Buffer Pools and Extended Storage

DB2 UDB V5 supports multiple buffer pools. These buffer pools cache data in memory for the database manager. Each buffer pool can have one or many table spaces assigned to it.

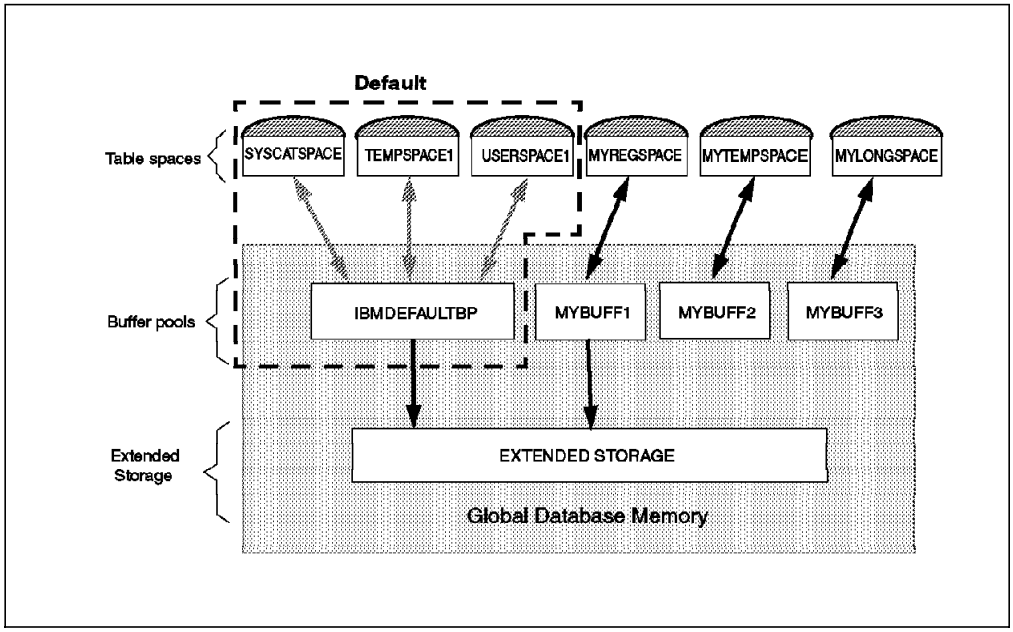


Figure 16. Buffer Pool Overview

Figure 16 gives an overview of the relationship between table spaces, buffer pools and extended storage in DB2 UDB. Each of the components will be discussed in detail in the following sections, and the benefits of this new functionality will be explained.

When a database is created, a default buffer pool is created. This is called IBMDEFAULTBP. In addition, three table spaces are created. The default names of these table spaces are SYSCATSPACE, USERSPACE1 and TEMPSPACE1. IBMDEFAULTBP is the default buffer pool for SYSCATSPACE, USERSPACE1 and TEMPSPACE1. The assignment of the default table spaces to IBMDEFAULTBP is represented by the light arrows in the diagram.

If the name of a buffer pool is not specified when creating table spaces within the database, they will by default use IBMDEFAULTBP as their buffer pool.

It is possible to alter the association of a table space to a buffer pool by using the ALTER TABLESPACE command or via the Control Center. For these changes to take effect, all existing connections to the database must be terminate and a first connect or activate db executed.

Three types of user-created table spaces are depicted in the diagram. These are intended to represent the table space types of Regular, Temporary and Long. The dark arrows between these table spaces and the buffer pools (MYBUFF1, MYBUFF2, and MYBUFF3) represent a user defined action to associate these table spaces exclusively to these buffer pools via the CREATE TABLESPACE or ALTER TABLESPACE commands or via the Control Center.

The buffer pools MYBUFF1, MYBUFF2 and MYBUFF3 have been defined by the user prior to making this association. This can be done via the CREATE BUFFERPOOL command or via the Control Center. All buffer pool memory allocations come from Global Database Memory which is available when the database is activated.

Extended storage is an area of memory defined by the database administrator. DB2 uses extended storage as a disk cache, storing images of some pages in memory so that it may more quickly get pages from memory and copy pages from extended storage into a buffer pool, rather than reading from disk.

Extended storage allows the database administrator to configure DB2 to make use of more than the maximum amount of process addressable memory on the machine where the database manager is operating. This is useful in the case where DB2 is running on a machine where the real memory is much larger than the maximum address space.

Note: This applies to high end Sun Solaris machines that have a maximum addressable memory of 2 GB and have more than 2GB of physical memory (RAM) installed.

This feature allows the user to specify an additional amount of memory that can be used as an extended cache to improve performance of the database manager. This technique will improve the performance of some current SMP machines that are limited in the amount of memory they can currently address

In Figure 16 on page 20, an extended storage area has been defined. This is also allocated as part of the total memory available to Global Database Memory. The dark arrows between IBMDEFAULTBP and MYBUFF1 and the extended storage represent that the user has defined that these buffer pools are enabled to use the extended storage area. This can be done in the CREATE BUFFERPOOL or ALTER BUFFERPOOL statements or via the Control Center.

The default situation when a database is created is that no extended storage is defined.

2.1.1.1 Memory Allocation

The space defined to the database buffer pools and extended storage is allocated at database activation. This occurs when either a DB2 ACTIVATE command is issued or when the first connection is made to the database. This allocation is made from the area of memory defined as the Global Database Memory.

Each one of the buffer pools and extended storage is allocated its own exclusive area of memory. The total area allocated to buffer pools can be estimated as:

Sum for all buffer pools of:

```
number of pages (NPAGES) * size of pages (PAGESIZE)
```

These values can be obtained from the catalog tables (see following section on examining buffer pools). There is a very small overhead associated with each buffer pool that is not included in this estimate.

Each operating environment for DB2 UDB will have its own default for buffer pool size. This is the value of BUFFPAGE in the database configuration parameters. The following are the default values of BUFFPAGE:

- UNIX - 1000 pages,
- OS/2, Windows NT, Windows 95 - 250 pages

Use the following command to change the value of BUFFPAGE:

```
db2 update database configuration for dbname using BUFFPAGE XXX
```

where dbname is the database name and XXX is the new default size for buffer pools of 4 K pages.

Changing the value of BUFFPAGE in the database configuration will not affect the size of any buffer pools which have already been created.

When creating buffer pools via the Control Center, use the value -1 if you want size to be the default value (that is BUFFPAGE), for this database.

The total space allocated to extended storage can be estimated as:

```
number of segments (NUM_ESTORE_SEGS)
* size of segments (ESTORE_SEG_SZ)
(in 4 K pages)
```

If there is not enough space available in the machine's virtual memory to allocate the memory required for the buffer pools and extended storage, then only the IBMDEFAULTBP buffer pool will be allocated. If there is not enough space to do this, then a reduced size IBMDEFAULTBP will be allocated.

If there is a problem with the allocation of memory space at database activation, warnings will be issued.

2.1.1.2 Performance Improvements

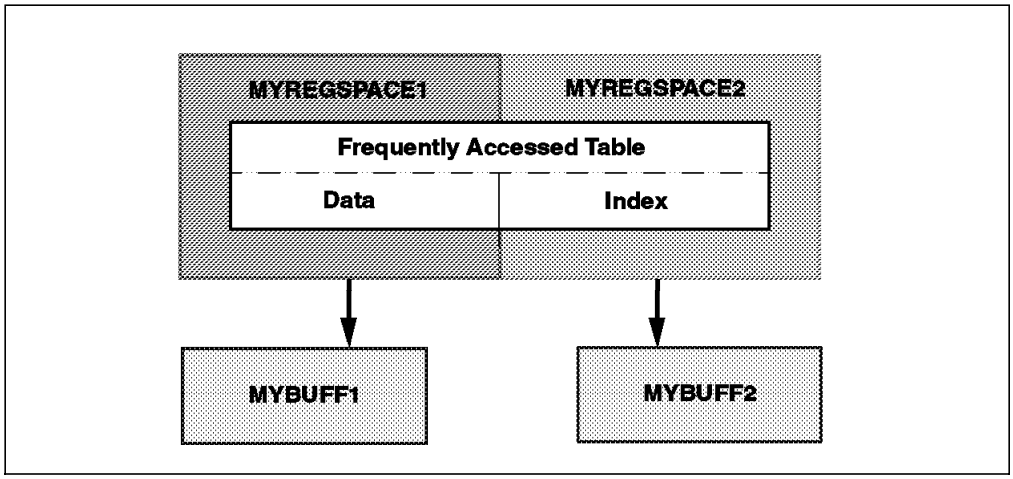


Figure 17. Performance Improvements

Overall granularity of buffer pool management has been improved. This new level of granularity provides the basis to make the following performance improvements:

- Allocate separate buffer pools for tables and indexes as shown in Figure 17.

  In an environment such as OLTP, where a number of key tables and indexes can be identified, the performance of the application may benefit from assigning the data portion of a table and the index portion of certain tables to different buffer pools. It may then be possible, for example, to provide

enough space in the index buffer pool to have a vital index permanently in memory once it has been accessed. As this is the only object in the database assigned to this area of memory, it will not be flushed out of memory by other application activity.

- Isolation of memory for important transaction tables

  If the application performance in your environment is heavily dependent on the use of one or more key tables then these tables can be defined in table spaces that are linked to their exclusive buffer pool. This will provide an area of memory exclusively dedicated to the processing of objects defined within the table space.

- Temporary table spaces limited to prevent ad-hoc queries affecting more critical applications

  A large ad-hoc query may typically require large temporary work tables for sorting during its processing. This can result in the flushing from the buffer pool of repeatedly used memory pages. By assigning temporary table spaces to their own buffer pools, you can prevent this effect. By isolating the temporary table space from memory used by tables critical to OLTP-type applications the impact of such ad-hoc activities on the more predictable application activities can be reduced.

- Limit bufferpools on large tables with random access patterns

  In an environment where random access patterns such as a data warehousing application occur, the benefits of defining a large table to its own buffer pool will be minimal.This is because the likelihood of the buffer pool containing information for reuse without disk I/O are low. In this type of scenario, a smaller buffer pool should be assigned to reduce impact on other more predictable system activity and conserve memory for other performance gains.

2.1.1.3 Table Space Relationship to Buffer Pools

All table spaces are assigned to a buffer pool. This assignment can be made with the CREATE TABLESPACE or ALTER TABLESPACE command or via the Control Center. A buffer pool must exist in order to be referenced in the CREATE TABLESPACE or ALTER TABLESPACE commands.

For example:

```
create tablespace MYTS bufferpool MYBP
```

A table space can have its association with its buffer pool changed to a different buffer pool by using the ALTER TABLESPACE command.This change does not become effective until the next “first connect” or “activate” against this database.

For example:

```
alter tablespace MYTS bufferpool MYBP
```

If during the creation of a table space a buffer pool is not specified, then IBMDEFAULTBP will be assigned as the buffer pool. During database creation, the three default table spaces that are created (SYSCATSPACE, USERSPACE1 and TEMPSPACE1) are assigned to the IBMDEFAULTBP buffer pool.

If a table space is dropped, the buffer pool to which it was assigned remains as an object within the database. This is the case whether or not any table spaces are now assigned to this buffer pool.

2.1.1.4 Catalog Views

Two new catalog views have been created to facilitate the use of buffer pools:

- SYSCAT.BUFFERPOOLS - This view contains a row for every buffer pool in every nodegroup. The columns defined are BPNAME, BUFFERPOOLID, NPAGES, PAGESIZE and ESTORE. NGNAME contains the nodegroup name. It is null if the buffer pool exists on all nodes in the database. NPAGES contains the number of pages in the buffer pool. PAGESIZE is page size for this buffer pool. ESTORE is a flag with settings Y or N as to whether this buffer pool can use extended storage.
- SYSCAT.BUFFERPOOLNODES - This view contains a row for each node in the node group where the size of the buffer pool is different from the size in SYSCAT.BUFFERPOOLS.

A new column has been added to the SYSCAT.TABLESPACES catalog view called BUFFERPOOLID. This is the ID (integer) of the buffer pool assigned to the table space. A value of '1' in the BUFFERPOOLID denotes the default buffer pool IBMDEFAULTBP.

2.1.2 Multi-Page File Allocation

Multi-page file allocation (MPFA) improves insert performance by allocating multiple pages ahead of a series of insert requests to remove the delay of allocating space one page at a time. However, using this type of allocation can provide a significant overhead for smaller tables defined within SMS table spaces. Multi-page file allocation is maintained in DB2 UDB V5 for compatibility with databases created under DB2 PE V1.2.

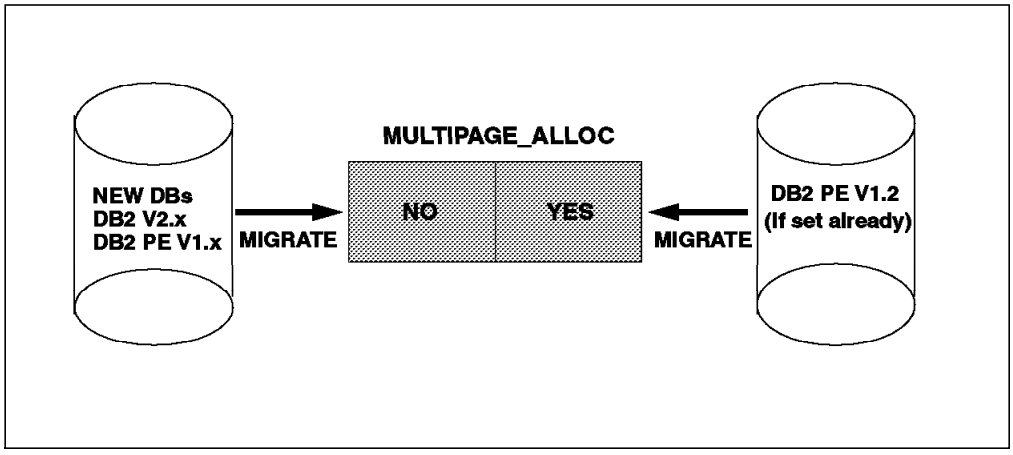


Figure 18. Multi-Page File Allocation

MPFA is controlled by the database configuration parameter, MULTIPAGE_ALLOC. Figure 18 shows how MULTIPAGE_ALLOC is set during the migration of database from previous versions of DB2.

- MPFA is only used for SMS table spaces

  When multi-page file allocation is enabled, it applies to all SMS table spaces in the instance. It is not possible to enable it for specific SMS table spaces.

DMS table spaces inherently pre-allocate space so this option does not apply.

It is recommended to use DMS table spaces to improve the insert performance and control overhead on a per table space basis.

- By default, migrated databases use MPFA

  Multi-page file allocation will only be set on in the case of databases being migrated from a DB2 PE V1.2 environment where the MULTIPAGE_ALLOC database configuration parameter is set to YES. All other migration cases will leave multi-page file allocation unset (NO).

- By default, new UDB V5 databases do not use MPFA

  The default for the multi-page database configuration parameter is not set. Therefore, any databases created in V5 will have multi-page allocation disabled. This default for a database can be changed by running the db2empfa utility. Once multi-page file allocation has been enabled, it cannot be reset without recovering to a pre-changed backup of the database.

2.2 Utilities Enhancements

A number of enhancements have been made to the database utilities in DB2 UDB V5. These include:

2.2.1 Index Free Space

Previously when a utility was executed where indexes were built (for example, the Load utility or REORG), each index was built with 10% of each index leaf page (pages which hold the RID’s for each row) reserved as free space for future inserts. If this space is filled up by later inserts, DB2 will split one or more pages into two or more pages to make space. Page splitting is expensive. It also affects the performance of index scans due to the index leaf pages not being in physical sequence on the disk.

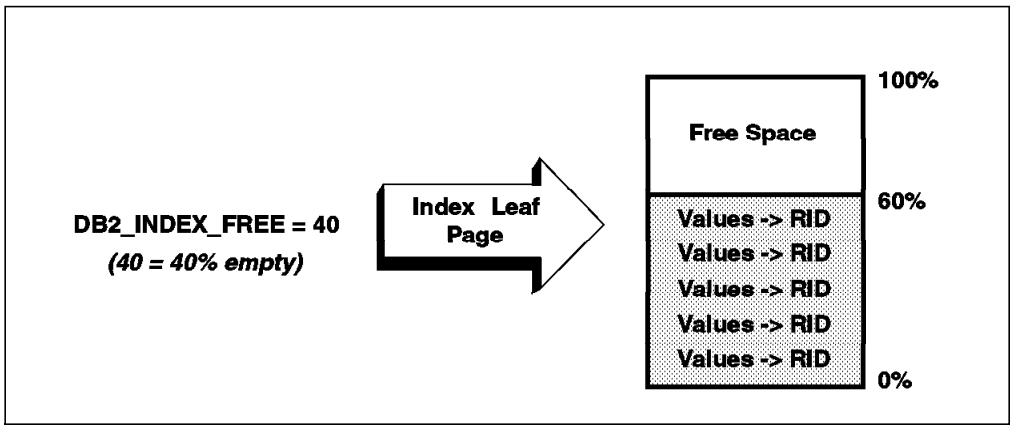


Figure 19. Index Free Space

A new environment variable DB2_INDEX_FREE controls the free space in index leaf pages as shown in Figure 19. This can be set to a value in the range of 0-90. For example, a value of 40 indicates that 40% of each index leaf page will be left empty by the Load utility. This variable must be set by the instance owner not the user running the Load. We can illustrate the effect of DB2_INDEX_FREE with an example. We have performed the same set of commands with DB2_INDEX_FREE=50, and then DB2_INDEX_FREE=0.

```
create unique index iid on tb4(id)
load from tb4.dat of del replace into tb4
runstats on table tb4 and indexes all

DB2_INDEX_FREE=50            DB2_INDEX_FREE=0

Rows  NLEAF  NLEVELS         Rows  NLEAF  NLEVELS
--------------------         --------------------
100     1       1            100     1      1
162     1       1            328     1      1
--------------------         --------------------
163     2       2            329     2      2
328     2       2            660     2      2
--------------------         --------------------
495     4       2            993     4      2
660     4       2
```

In the example, for DB2_INDEX_FREE set to 50, when NLEAF is 4, there are two pages. Whereas with DB2_INDEX_FREE set to 0, there is only one page at the same value of NLEAF. This is because in the first case, the index leaf pages are half empty.

2.2.2 Import Table Options

The following options have been introduced to the Import command.

2.2.2.1 New Numeric Data Options

The MODIFIED BY clause for DB2 Import has been expanded with two new options supporting:

1. Implied decimal points

Previously, when importing decimal data, if the decimal point was not included, DB2 would assume a decimal point at the end of the value. This meant that

12345 was imported into a DECIMAL(8,2) column as 12345.00. A new option (*filetype-mod*) MODIFIED BY IMPLIEDDECIMAL has been added to the MODIFIED BY clause of the Import (and Load) utility which instructs DB2 to use the column definition if no decimal point is included with the data. With this option, using our example of 12345 and column DECIMAL(8,2), the value stored by DB2 would now be 123.45. This option can be used with all current import file types (ASC,DEL,WSF,IXF). This option can avoid having to convert the input data from certain external data sources.

The following is an example of using implied decimal points with the import utility:

```
create table tb1 (name character(6), num decimal(8,2), id integer)

tb1.dat:
Tony,12345,01
Ken,12,02
Calene,987,03

import from tb1.dat of del modified by IMPLIEDDECIMAL insert into tb1

select * from tb1

NAME    NUM     ID
------  ------  --
Tony    123.45  1
Ken     0.12    2
Calene  9.87    3

  3 record(s) selected.
```

2.2.2.2 Importing Binary Numerics and Packed Decimal Data

Previously all numeric data had to be represented as character strings for input into the Import utility. A new option (filetype-mod) MODIFIED BY BINARYNUMERICS has been added to the MODIFIED BY clause of the Import (and Load) utility which instructs DB2 to treat the input data as binary or packed decimal. This option can avoid a potentially resource-intensive conversion from existing data sources to character-based representation of numeric data. DB2 is also able to process the binary/packed data faster than character numeric. This has the added benefit of improving the total run time for the Import utility. The DB2 Command Reference includes a number of restrictions for this option including the following:

- Support is only for positional ASCII (ASC) file type with fixed length records using RECLEN & NOEOF.
- All binary data must be in the binary format of the platform on which the IMPORT is performed.

2.2.2.3 Create Table into Table Space on Import

DB2 Common Server V2.1.2 provided the ability to create tables in tables spaces during import. However, many people may not be aware of this option.

Previously, the syntax of the Import command would allow you to indicate into which newly-created table you wanted to store the input data. This option was only supported under IXF. However, you could not specify the table space in which the table would reside. The Import CREATE INTO table-name clause has

now been expanded to include an optional table space parameter (TBLSPACE-SPECS). Note that the table space parameter cannot be used with the REPLACE_CREATE INTO table-name clause. The following is an example specifying the table space for the table you are creating:

```
import from tb1.dat of del
  create into tb2 in tbs1

import from tb1.dat of del create into tb2 in tbs1
SQL3303N  The file type must be IXF when using the CREATE or REPLACE_CREATE
keywords in the tcolstrg parameter.

export to tb1.ixf of ixf select * from tb1
import from tb1.ixf of ixf create into tb2 in tbs1
```

2.2.2.4 Real Data Type Support

The Import utility has been enhanced to include support for the new DB2 data type REAL. A “real” is a 4 byte floating point number with 7 digit precision.

```
create table tb3 (name character(6),RNUM real,DNUM double,id integer)

tb3.dat:
Tony,123.45,123.45,01
Ken,0.12,0.12,02
Calene,9.87,9.87,03

import from tb3.dat of del insert into tb3
select * from tb3

NAME    RNUM           DNUM                    ID
------  -------------  ----------------------  --
Tony    +1.23449E+002  +1.23450000000000E+002  1
Ken     +1.19999E-001  +1.20000000000000E-001  2
Calene  +9.86999E+000  +9.86999999999999E+000  3

  3 record(s) selected.
```

2.2.3 Load Utility Improvements

The Load utility is used primarily to move large amounts of data into a DB2 table. This can be a time consuming task and also introduces a number of restrictions which are not found in the Import utility. In UDB the overall performance of the Load utility has been increased with performance options like “Fast Parse”. Changes to the Load processing model allow greater I/O parallelism, SMP exploitation, and improved support in a multi-node environment.

2.2.3.1 Load Performance Options

A new file type mode (file-mod) has been added to the Load utility. It improves the performance of a load operation by reducing data validation checks. It is implemented using the MODIFIED BY FASTPARSE clause of the Load utility and is supported for all input data types. Performance gains should be greater for ASCII input files, either positional or delimited. The MODIFIED BY FASTPARSE option will apply to all data and all columns from the input data. Performance

gains will be greater when the input data is in the correct form for the DB2 table being loaded.

Tables loaded under this option are valid in terms of their structure. The Load utility will perform sufficient data checking to prevent a segmentation violation or cause a trap to occur in DB2. If the input data is in the correct form, it will be loaded correctly. This option improves the speed of loading data by reducing the amount of code validation to be performed by the Load utility.

Load the data without order restrictions: The MODIFIED BY ANYORDER option can be used in an SMP environment where multiple processors can work on the load operation at one time. Data is normally loaded in the sequence that appears in the input file(s). If a particular sequence is desired, the data should be sorted before the load is executed. The MODIFIED BY ANYORDER tells the Load utility to load the data where order does not matter. The order of the rows built on the data pages may be different to the order of the lines of data in the input file. The command used for the load is as follows:

```
Load from tb4b.dat, tb4c.dat of del MODIFIED BY ANYORDER insert into tb4
```

This will improve the performance of the Load utility. However, the order in which the rows are loaded into the table cannot be guaranteed.

Binary input data: The Load command also supports the MODIFIED BY BINARYNUMERIC option. This improves the performance of load by removing the need to convert character numeric input data into DB2's internal data formats.

Sort Indexes Using Third Party Tools: This option became available in DB2 Common Server V2.1.1. However, many people may not be aware of it.

DB2 V2.1.1 introduced an additional environment variable which instructs the Load utility to use an external sorting program rather than the one supplied with DB2. An example of an external sorting program is IBM SMARTsort. SMARTsort performs an index sort for the Load utility considerably faster than DB2's sort program. This is performed by installing the SMARTsort product and by setting the environment variable DB2SORT=SMARTSORT. Also, you may set DB2SORTLOG=/somedirectory. This option is only supported using Load in UNIX platforms. The SMARTsort product is available for OS/2 and NT as well and can be used to improve sort times when pre-sorting data before the Load is run.

SMARTsort can be defined as the native sort for UNIX platforms. Existing programs that call the UNIX native sort program need not be changed to take advantage of the superior performance of SMARTsort. More information on SMARTsort can be obtained from the SMARTsort web address:

http://www.storage.ibm.com/storage/software/sort/srtshome.htm

Optiload: Optiload is a third party utility to manage the complete loading process for a Data Warehouse. Optiload is produced by:

Leveraged Solutions Inc, Phone: (513) 984 5665, Fax: (513) 985 8143

Optiload allows all steps of an integrated load to be initiated and controlled from a single host-based process. The data maintenance specifications are controlled via a single script file. All data movement, conversion and updates are then

performed from a single execution of Optiload. For multi-node systems, data is split by node and fully parallelized. Optiload may also be controlled directly from most existing NCR Multiload TM (Multiload is a registered Trademark of NCR Corporation) scripts.

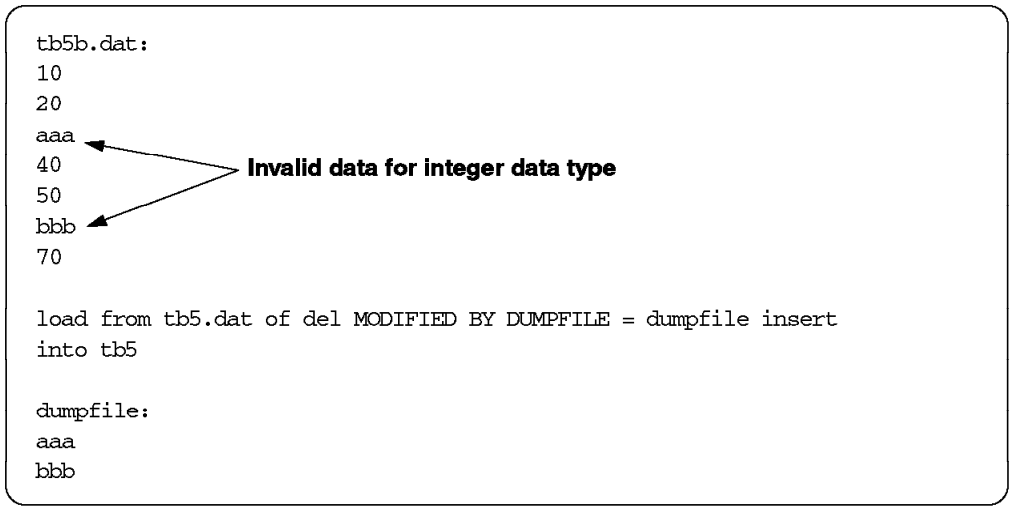
2.2.3.2 Load Functionality and Usability

Code Page Conversion Support: Previously, the Load utility was required to execute with the same code page as the database. A new option has been added to the load command to allow you to specify the code page of the input data. This is implemented using the MODIFIED BY CODEPAGE=x clause of the Load command. It is supported for all input data types.

This option will normally be needed when using the BINARYNUMERIC modifier. Modifiers can be combined by specifying one after another with a blank separator. For example: "LOAD... MODIFIED BY BINARYNUMERIC CODEPAGE=850..."

Binary Data, Implied Decimal and REAL: The Load command will also support the MODIFIED BY IMPLIEDDECIMAL option. The option is the same as was documented in the Import utility in this section.

Expanded support for exception file on ASCII loads: In DB2 PE V1.2, the option MODIFIED BY DUMPFILE for positional ASCII (ASC) would write rows that had been rejected due to invalid data during the load process to a file. This feature is available in DB2 UDB and has been extended to support delimited ASCII (DEL). For each input record, a maximum of 32 K of data will be written to the exception file. The following example illustrates the usage of the MODIFIED BY DUMPFILE option with the Load utility:



Bypass Multi-node Headers: In a multi-node environment, db2split writes a header into each output file. The header includes the node number, the partitioning map, and the partitioning specification. The Load utility uses this information to verify that the data is being loaded at the right node. When running the Load utility in a single-node nodegroup, data files do not have to be processed by the splitter utility. To load such files (those that do not have a header) into a table that exists on a single-node nodegroup, a MODIFIED BY NOHEADER option was added to the MODIFIED BY clause in DB2 PE V1.1 (only when using ASC or DEL formats). In UDB EEE, this option can now be used to

allow the processing of PC/IXF files (still only when you are loading a table in a single-node nodegroup). The Load utility with multi-node nodegroups does not support the IXF file type.

User Interface Support: The Load utility can be run from the DB2 Control Center by selecting the table to be loaded from the list of tables. Click on the right mouse button then select load from the pop-up menu. The Load notebook is displayed and the source input stream and the load options can be specified. Once completed, choose the **OK** button to submit the load request.

Improved API Support: The DB2 REXX API's have been expanded and now include the load function.

2.2.3.3 Non-Recoverable Load

One of the new options to the Load utility is the NONRECOVERABLE option. This option allows you to perform a non-recoverable load of a table without affecting the recoverability of all other tables in the database. This option can be used when forward recovery (archival logging) has been enabled and the default COPY NO load option is used. The NONRECOVERABLE option specifies that the table being loaded is to be marked as non-recoverable. To recover the table, you must either re-issue the load command or use import.

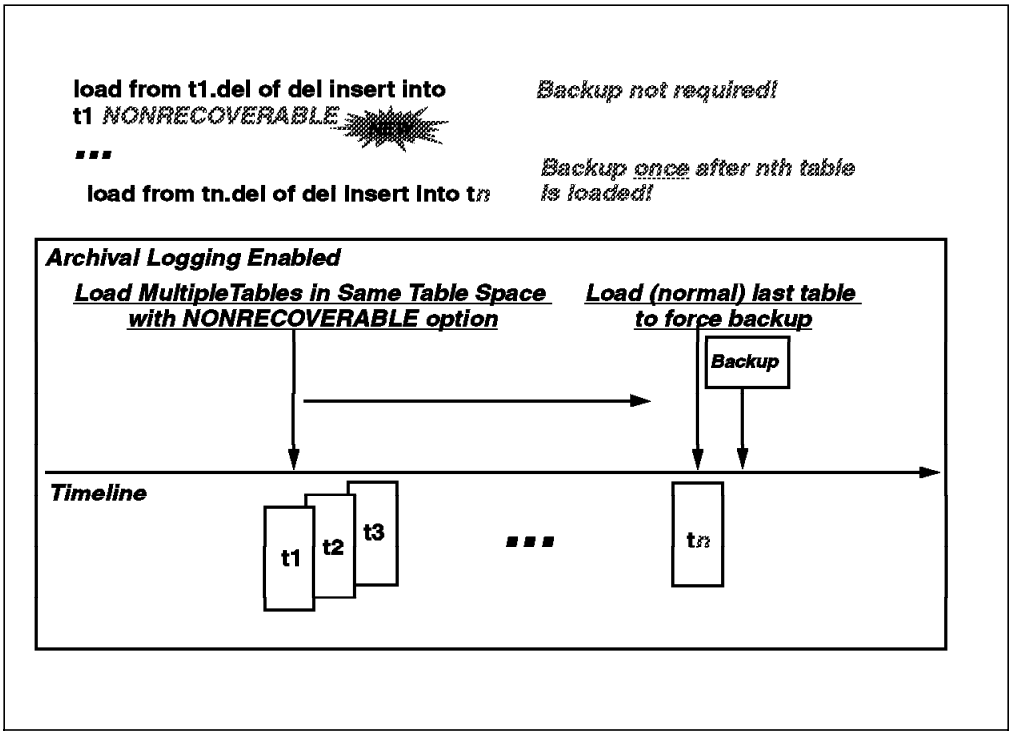


Figure 20. Non-Recoverable Load

One of the scenarios where this option could be used is shown in Figure 20. If you need to perform multiple load operations in the same table space, you could use the NONRECOVERABLE option on all but the last table. The last table would be loaded without using the NONRECOVERABLE option causing the table space to be placed in a backup pending state. The benefit of this method is that only one backup is necessary after all the loads are completed.

Another use of this option could be for loading a read-only table, especially if the table is large. In this case, it is not important that the table is non-recoverable since in the event of a failure, a load could be used to restore the table.

2.2.4 Multi-Node Load Improvements

For users of DB2 Parallel Edition V1.x, a number of improvements have been made in the area of loading data.

2.2.4.1 Concurrent Split

When executing DB2 in a multi-node environment, the db2split utility allows you to divide a source file (positional or delimited ASCII) into a number of partitioned files that could then be loaded on separate nodes. In the previous version of DB2, the Load utility could then be executed on each node in parallel. However, db2split had to first be executed on a single node. The db2split utility can now be executed at a number of nodes in parallel taking full advantage of a multi-node environment and its performance capabilities, as shown in Figure 21. This allows the splitting stage to complete faster, thus reducing the overall load time.

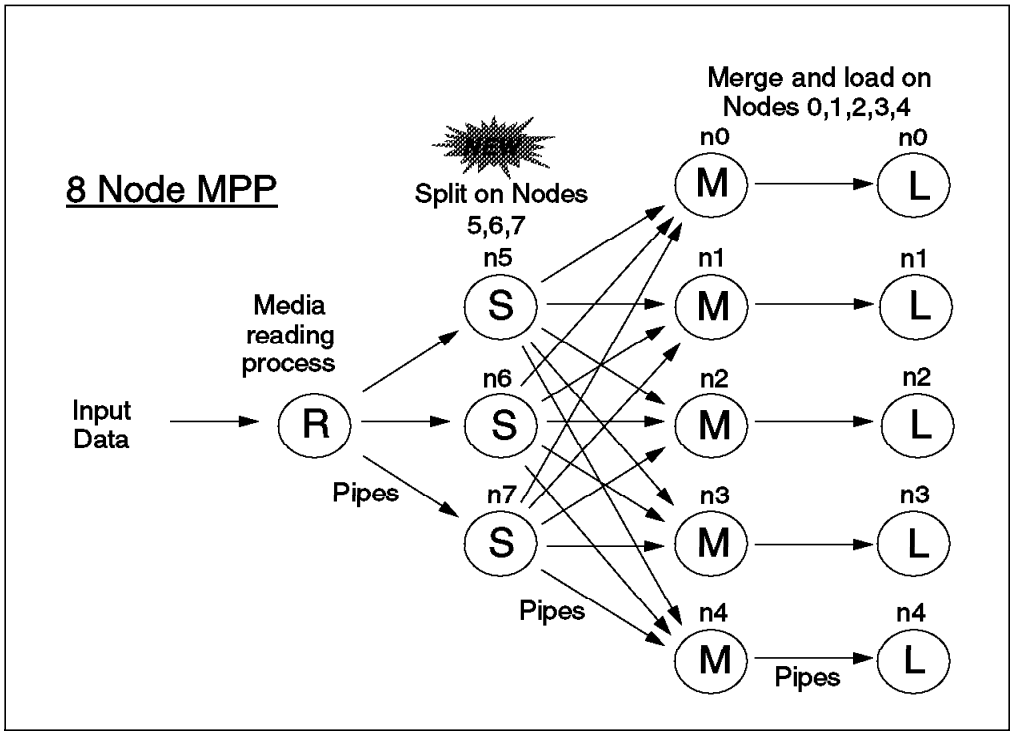


Figure 21. Multi-Node Load - db2split

Db2split will no longer be delivered as source code and therefore will not require compilations to be performed.

It has also been enhanced with a new BINARY option which allows binary numerics to be processed similarly to the BINARYNUMERIC option in the Load and Import utilities. The BINARY option will support the following data types:

- integer
- small integer
- float

- double
- packed decimal
- character

For more details, see the Import/Export utility in this section.

For delimited ASCII (ASC), the limit of 32 K on record lengths has been removed.

The following considerations also apply:

- REAL and FLOAT data types are only supported as partitioning keys if the BINARYNUMERIC option is used.
- If splitting delimited ASCII data (DEL), only non-control character data is allowed. Some invalid characters in the data stream (such as binary zero) may cause unpredictable behavior.
- When splitting positional ASCII (ASC) data, special characters (such as newline, binary zero, and so forth) will be split correctly if RECLEN is specified and NEWLINE is FALSE. Otherwise, behavior is unpredictable.

2.2.4.2 Autoloader

Autoloader has been enhanced to take advantage of parallelism during the splitting stage of the autoload process. In DB2 UDB EEE, users have the option to select a set of nodes, which may be the same or different than the nodes being loaded, to participate in the parallel split process. The output from these multiple splitters will be directed to multiple nodes for a multi-node load. In addition, users only need a single configuration file called autoloader.cfg. All temporary files such as the splitter.cfg, load.script and ftp.script will be generated automatically. In short, autoloader has been made more user-friendly.

The following restrictions also apply:

- A single splitting node should be used if the data needs to be loaded in the order of the original data.
- Load TERMINATE and RESTART are not supported with autoloader.
- Only a single autoloader run should be performed at once.
- When running the Load utility on multiple nodes, statistics (runstats) can only be collected on the catalog node. A warning will be issued if the Load command option STATISTICS=YES is attempted on any other node.

The command syntax for the autoloader is the following:

```
db2autold [-h] [-d] [-i] [-c] config_file
```

The configuration file can be used to specify the operating mode for a particular run, which nodes the db2split processes will run on, which node will be involved in the load, and the load option to be used. There are four run modes for Autoloader:

- **analyze** - generates balanced partition maps,
- **split** - only splits data with the output going to files,
- **load** - only loads previously split files, and
- **split and load** - splits and load data.

2.2.5 Backup and Recovery

A number of improvements have been made concerning backup and recovery, notably:

- Restart

  The number of log file checkpoints taken by DB2 has been increased. Accordingly, the SOFTMAX parameter has been altered to allow users more control over this frequency of these checkpoints. By increasing the frequency of these check points, the amount of processing on restart can be reduced.

- Backup

  The performance of the backup utility has been enhanced with increased parallelism. See 2.3, “Parallelism and SMP Enablement” on page 40 for further details of this support.

- Restore

  The performance of the restore utility has been enhanced with increased parallelism. See 2.3, “Parallelism and SMP Enablement” on page 40 for further details of this support.

  Additional functionality has been added to the restore utility to allow a subset of table spaces to be restored from a single backup image. This subset table space restore will allow users greater flexibility when building recovery strategies. Additionally, the table space definition may be redefined to use a new container name during the restore operation.

- Rollforward Recovery

  Prior to UDB V5 when table spaces were recovered, all log records had to be applied for the database. The point-in-time based recovery for complete databases has now been extended to table space-level recoveries. This means a partial numbers of logs may be applied giving the user greater flexibility when using a table space recovery.

- Improved History and Status Information

  With additional functionality being added to the DB2 restore utility, there has been a number of corresponding changes to the Recovery History Files and `LIST TABLESPACES` commands to report the status or history of these events.

2.2.6 DB2 Governor

The DB2 Governor is used to monitor the users’s activity and, if required, take appropriate actions.

It collects statistics on a regular basis defined by a configuration file.

The governor then evaluates the statistics against a set of rules that are also defined in the configuration file. Based on these rules, the governor may change the application’s priority, or force the application off the database. The governor will also record the actions it has taken in a log file.

In the DSS environment, long running queries are quite common. However, you may want to limit their use to certain portions of the day, or reduce their priority so they do not negatively impact any other work on the system.

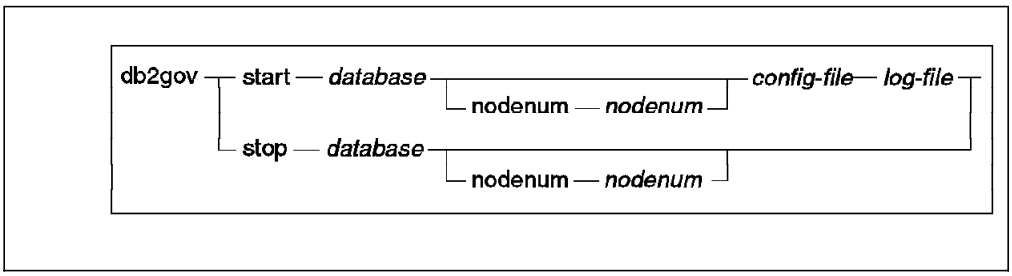


Figure 22. DB2 Governor Command Syntax

The *db2gov* front-end utility is used to start or stop the governor. SYSADM authority is required to use the utility. If one of the actions requires changing the priority of a process, root must be defined as a trusted user on all machines in the instance. The root user ID also needs to be in the user list associated with the SYSADM group.

The syntax of the db2gov command shown in Figure 22 indicates:

- *database* - the name of the database for which the governor is being started or stopped.
- *nodenum node-num* - if the governor daemon only needs to be started or stopped on a single node, this optional syntax will allow that to be specified.
- *config-file* - specifies where the rules for monitoring the database can be found.
- *log-file* - specifies the base name of the log file to which the governor will write.

Figure 23 shows an an example of a governor configuration file:

```
{Wake up every minute, the database is dss, account every 10 minutes }
interval 60; dbname dss;  account 10;

desc "CPU restrictions apply 24 hours a day to everyone"
setlimit cpu 6000 rowssel 1000000;

desc 'Slow down a subset of applications'
applname jointA, jointB, jointC, queryA
setlimit cpu 3 locks 1000 rowssel 500;

{ Slow down the use of db2 CLP by the novice user }
setlimit uowtime 3600;authid novice:
applname db2bp;
setlimit cpu 5 locks 100 rowssel 250;

{ During day hours do not let anyone run for more than 10 seconds }
time 8:30 17:00 setlimit cpu 10 action force;

{ Some people should not be limited -- database administrator
   and a few others. As this is the last specification in the file, it
   will override what came before. }
authid melanie, tony, calene, patti setlimit cpu -1 locks -1 rowssel -1;

{ increase the priority of an important application so it always
  completes quickly }
applname pgm1 cpu 1 locks 1 rowssel 1 action priority -18;
```

Figure 23. Governor Configuration File

The configuration file contains the rules to govern applications executing against the database. It can be changed without stopping the governor; each governor daemon will detect if it has changed and re-read it. In a multi-node environment, the configuration file must be in a directory that is mounted across all of the nodes.

Comments are specified within {}. Rules that must be specified include the name of the database, how often the governor should check the applications, and the rules on how to govern the applications. Each rule must end with a semicolon.

The rules that govern the application may include a description of the rule, the time period during which the rule is to be evaluated, a specific authid to check for, a specific applname to check for, the limits to apply to this particular time period or authid or application, and the action that should be taken should one or more of the limits be exceeded.

The governor has two intervals:

1. Snapshot interval - specified in seconds in the *interval* rule.
2. Accounting interval - specified in minutes in the *account* rule.

A snapshot is taken every snapshot interval and accounting records are written when an application termination is detected or at the accounting interval statistics for each connection. The accounting interval can not be shorter than the snapshot interval. For a short connect session that occurs entirely within the snapshot interval, no log record is written.

The limits can indicate a maximum for CPU seconds, locks, rows selected, or time exhausted by the UOW. The actions can include forcing the application, or changing its priority. Changing its priority is the default action if none is specified.

2.2.7 Diagnostic and Repair Advances

In DB2 UDB V5, additional functionality has been added to the DB2DART repair tool. This tool checks the integrity of data and index pages and can either run in check mode or repair mode.

Two new options have been added to enable users to repair damaged tables:

- /DDEL to allow users to dump any readable rows from a corrupted data page to a file.
- /IP to mark a corrupted page as empty.

A typical use of DB2DART to repair a damaged table is:

1. User notices from messages in DB2DIAG.LOG that there are problems accessing table t1.
2. He or she runs DB2DART in inspect (check) mode on table t1. This writes a message into DB2DIAG.LOG saying that page 53 is damaged.
3. Next, he or she runs DB2DART /DDEL on page 53 and using t1.out as output file. This dumps all the readable rows from page 53 into t1.out in DEL format. Some of the rows are unreadable and are not output.
4. Then, he or she runs DB2DART /IP using page 53. This marks page 53 as empty.

5. To make the table available again, he or she now exports the table in IXF format, drops the table, and uses import create to recreate the table and reload the data. This forces the data and referential integrity of all the rows in the table to be checked.
6. He or she inserts the rows that were dumped in step 3. These are the rows that were salvaged from the damaged page 53.

2.2.8 Other Utilities Enhanced

Online help for some tools can be found by using the -h option when invoking them. For example, db2batch -h. The tools are found in the misc. subdirectory.

- db2look is used to examine the catalog statistics. It has many options to limit the output to particular table spaces, tables, indexes, and columns. The new -e option allows you to extract DDL from the system catalog to recreate a single table.
- db2batch is used primarily to get performance information easily. It reads SQL statements from either a flat file or standard input and can dynamically describe and prepare statements. It returns an answer set that is controlled in size.

Runstats Utility has the following improvements:

- improved frequency algorithm
- runs faster and uses less memory
- correlation statistics for multi-column indexes
- new statistics FIRST2KEYCARD, FIRST3KEYCARD, and so forth.

2.2.9 Data Replication

The administration of IBM Data Replication is performed using the DB2 UDB Control Center GUI tool.

Figure 24 on page 38 shows you how to use the Control Center to define sources and targets for replication, as well as the schedule for updating the targets, enhancements that are made to the target data, and any triggers that kick off replication.

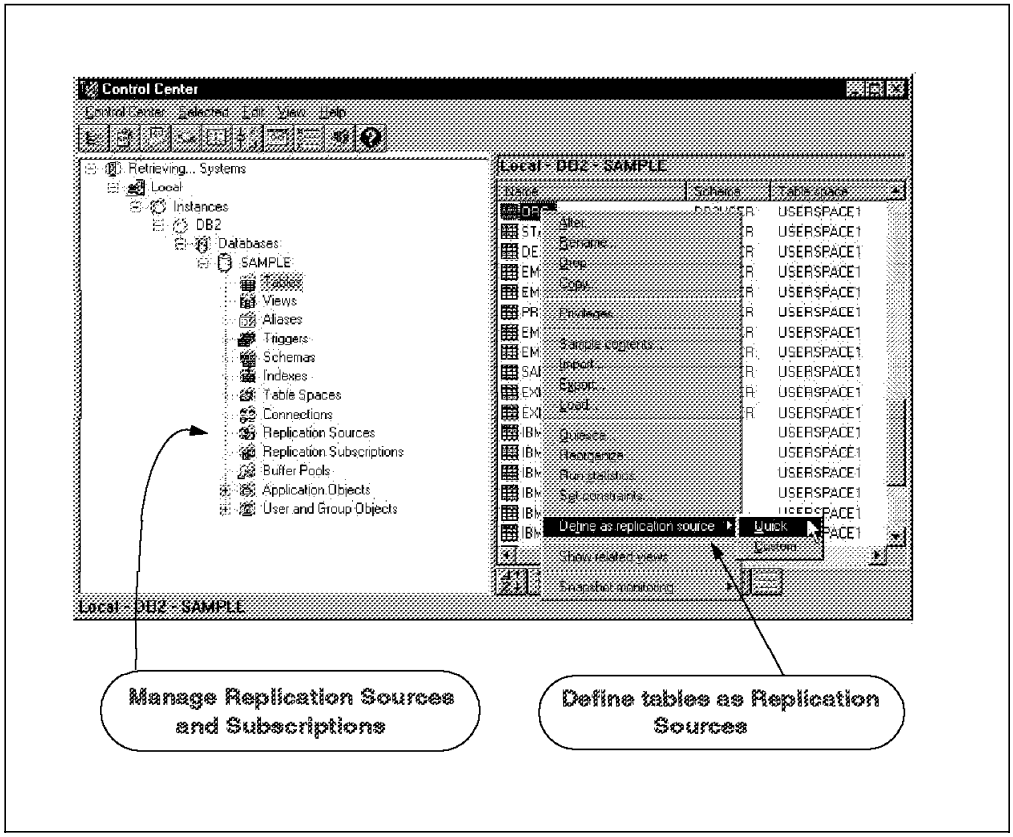


Figure 24. Replication Administration

Replication administration has two major tasks, defining replication sources and defining replication targets, or subscriptions. Replication sources are DB2 or external tables, or views, used as source for copying data to a target table. Replication subscriptions are the specifications for one or more target tables and their location, structure, and timing schedule, as well as any SQL enhancements that are necessary. Other replication tasks include maintaining the sources and targets, and cloning them to other servers.

There are three containers for organizing the objects in the Control Center that you use to set up and maintain your replication environment:

- Tables folder
  - The folder containing user-defined DB2 relational tables.
- Replication sources folder
  - The folder containing tables that have been defined as replication sources: DB2 tables, internal tables, views, or target tables redefined as sources for replication.
- Replication subscription folder
  - The folder containing subscription definitions for copying updated source data to target tables.

Each folder contains the objects that you use for replication activities: source tables and subscription definitions. The replication objects have pop-up menus with the actions that you can perform against them, like defining a source table or cloning a replication subscription. The folder has several actions that are

performed against the entire folder. Each object also has a folder with the actions that can be performed with the object.

2.2.9.1 New Functions in Data Replication

New graphical user interface:

- The new function in the GUI provides features for power users to define multiple registration, now known as *defining replication sources*, or multiple subscriptions, in a single user action.

Administration support for both 'push' and 'pull' configurations:

- Determines whether the Apply program pushes data or pulls data to the target table.
- Pushes configurations for low latency requirements.
- Pulls configurations for lowest cost.
- Updates replication using a PUSH+PULL model to support occasionally connected users.

Update-anywhere capability:

- IBM Replication now provides update anywhere copying.
- Rigorous conflict detection mechanisms for declarative constraint conflicts and trigger-detected conflicts.
- Automatic compensation of conflicting transactions.
- Asynchronous transaction coordination:

  Conflicting transactions are compensated much like the familiar DB2 rollback processing. Any detected dependent transactions are also rejected and compensated.

Support for occasionally connected systems:

- IBM Replication provides ′dial′ and ′disconnect′ exits for telephone connectivity.
- Apply can defer reporting the subscription progress to the pruning control table to reduce remote SQL CONNECT processing for a set subscription to a single CONNECT call and optimize remote thread allocation and password verification.
- Mobile users - By standardizing on a push/pull Apply configuration to maintain updateable target tables, Apply users need only client connectivity to support bidirectional update-anywhere propagation. The single Apply process on the notebook computer handles posting of updates propagating to or from the notebook. While disconnected, the notebook imposes no burden on the network. Central servers are not kept busy looking for the mobile notebook computer. Rather, the notebook computer ′finds′ the central server when it dials in.

2.3 Parallelism and SMP Enablement

In DB2 Universal Database V5, parallelism support has been improved to exploit the symmetric multi-processor (SMP) systems. This new functionality will better exploit SMP shared memory architecture to speed up individual SQL queries and utilities. The utilities also have improved support for parallel I/O servers for driving multiple disk drives, similar to that already supported with SQL requests. The new SMP support as shown in Figure 24 on page 38 builds upon the parallelism support that existed in prior versions of DB2 including:

- Parallel transactions - Each users request could be run on a separate processes or threads
- Parallel computers - use DB2 Parallel Edition to support MPP machines with multiple parallel database nodes
- Parallel I/O - the ability to define multiple processes or threads to manage the I/O activity for different physical drives
- Parallel disks - DB2 would allow databases to be created over many disk drives allowing concurrent physical access to data

The new SMP parallelism support in DB2 can be used in these environments:

- A single system where users processing needs can be efficiently and cost-effectively satisfied by a single SMP machine which over time is upgraded by adding CPUs.
- A shared nothing cluster or MPP environment which provides large scale, shared nothing parallelism.

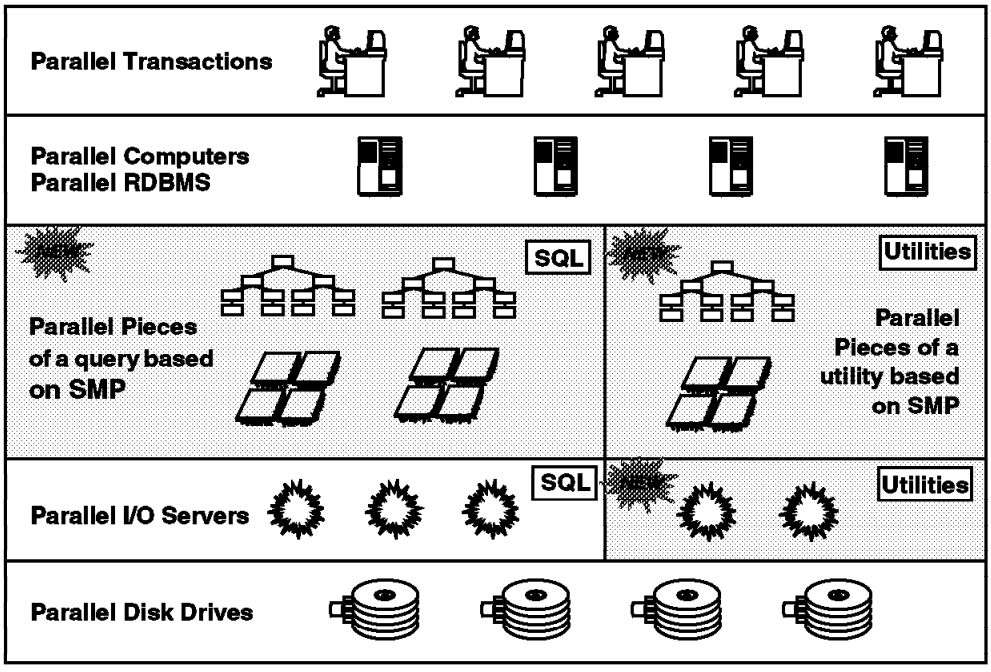


Figure 25. What's New in DB2 UDB Parallelism

The major changes in DB2 to exploit SMP parallelism start in the optimizer. The optimizer must build SQL access plans which fully exploit an SMP computer. This has been achieved in DB2 by the optimizer being extended to consider

access plans which instead of running a complete query, break the query down into different parts. This is often called query decomposition. Where as before a query would be a sequence of operators that were sent to a single process or thread to be executed, DB2 can now consider different groupings of the operators that can be sent to different processes or threads to be executed. The parts are called subsection pieces (SSPs), with one SQL statement consisting of one or many subsection pieces. The use of subsection pieces or SSPs will only be considered by the optimizer if parallelism support has been enabled in DB2.

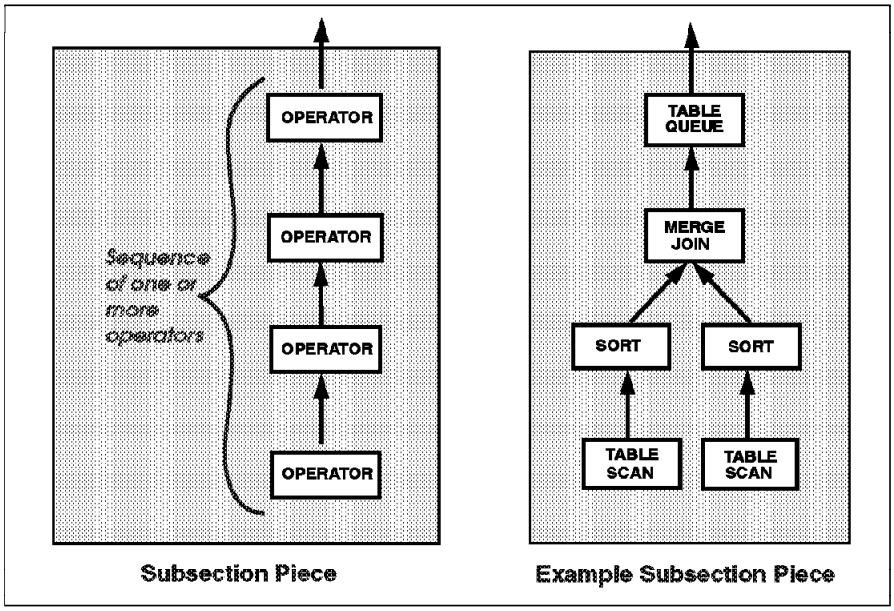


Figure 26. Subsection Pieces

Subsection Piece (SSP): A subsection piece, as shown in Figure 26, is a sequence of one or more database operators belonging to the same SQL query, which, when executed, complete the processing part of that query. A subsection piece must be executed in one DB2 process or thread. A subsection piece may receive data from other DB2 subsection pieces or feed data to another subsection piece or return data back to the user. Subsection pieces can be cloned or duplicated by the optimizer (this is discussed later) to speed up the processing.

Subsection: A subsection is the smallest unit of a SQL request that may be shipped between database nodes. In contrast, a subsection piece is a subset of (or a complete) subsection and is the smallest unit that can be shipped between processes or threads on the same computer or database node. When running DB2 Universal Database Extended Enterprise Edition in an MPP environment, the optimizer is able to decompose a single SQL request to be run over a number of database nodes. The optimizer will perform the decomposition of the SQL request into subsections which, like subsection pieces, are a sequence of database operators. This decomposition of an SQL request will happen before it is decomposed further into subsection pieces. Subsections do not exist in a single node environment like DB2 Universal Database Workgroup or Enterprise Editions since an SQL request is broken down directly into subsection pieces.

Operator: An operator is the smallest step in the execution of a SQL request externalized to users by DB2. Operators perform a distinct action on the data being processed. One example is a sort operator. By combining operators into a sequence of steps, DB2 is able to fully resolve an SQL request, manipulating its source tables into the answer set required by the user.

Degree of Parallelism: DB2 UDB has introduced a new configurable parameter known as DEGREE OF PARALLELISM. The degree of parallelism limits the number of subsection pieces (SSPs) a query can be broken down into. In Figure 27, the query is broken in 4 SSPs and, therefore, the degree of parallelism is 4. When a query is broken into multiple SSPs, a table queue is built to coordinate the return of data from the SSPs.

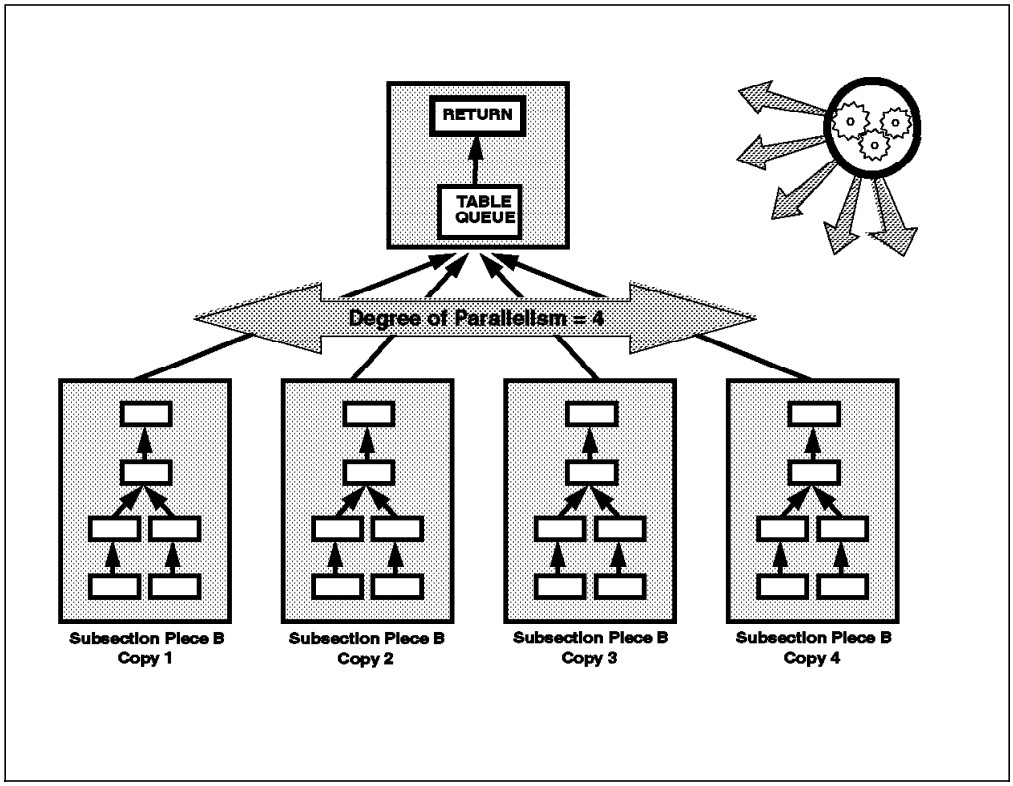


Figure 27. Degree of Parallelism

The optimizer handles all of this behind the scenes and therefore the user does not need to change anything in their application to utilize SMP parallelism, other than setting the degree of parallelism > 1. Users can explicitly set the degree of parallelism either for the DB2 instance, the application, or the statement. In addition, if a section is built by the optimizer with a certain degree of parallelism, it can be reduced if the instance default degree of parallelism is lower.

When breaking a query into multiple SSPs, the optimizer will also determine if the SSPs will work on the same data or if the data can be partitioned among the SSPs. In determining the optimal way to split the section into SSPs and to partition the data the optimizer will be influenced mainly by the degree of parallelism and the cardinality of the data.

In an SMP environment, the SSPs will be duplicates or clones of each other. DB2 will then execute the SSPs in parallel and return the results faster than if the query was run on one processor. Each SSP copy will work on a subset of

the data. As determined by the optimizer, this subset may be based on the data values or on equal divisions of the data based on the number of rows.

Note: If the degree of parallelism is set to a value > 1, the number of SSPs created is equal to the value of *degree of parallelism*. When setting the degree of parallelism for your system, you should match this to the number of CPUs in the SMP system. (This would be 1 in a uni-processor environment.)

If the degree of parallelism is set to ANY, then the optimizer will decide the degree of parallelism for each SQL query.

For instance, on a 4-way SMP machine, a simple insert statement will gain no advantage from being run in parallel across the CPUs, so using the ANY setting will result in this type of statement running on only 1 CPU (a degree of parallelism of 1).

Data and Functional Parallelism: Figure 28 on page 44 shows an example of both functional or pipeline parallelism and data parallelism. There is one SMP machine in the diagram. The data from the lineitem table is dynamically partitioned between the four CPUs. Each CPU then performs a series of operations, for example a sort, against its portion of the lineitem table. This is an example of *data parallelism*.

Each CPU (CPU 1 - 4) copies its sorted data set into one of four table queues, marked t1 - t4 in the diagram. CPU5 then reads the top page from each of the table queues and sorts the data in these pages. If there is a page filled in each of the table queues, CPU5 is able to perform an overall sort even though the table queues may not contain all of the data from the four previous sorts. This is an example of functional or pipeline parallelism. In this manner, the final sort done by CPU5 is working simultaneously with the sorts on CPUs 1-4. The output from the sorts from CPUs 1-4 is pipelined to CPU5.

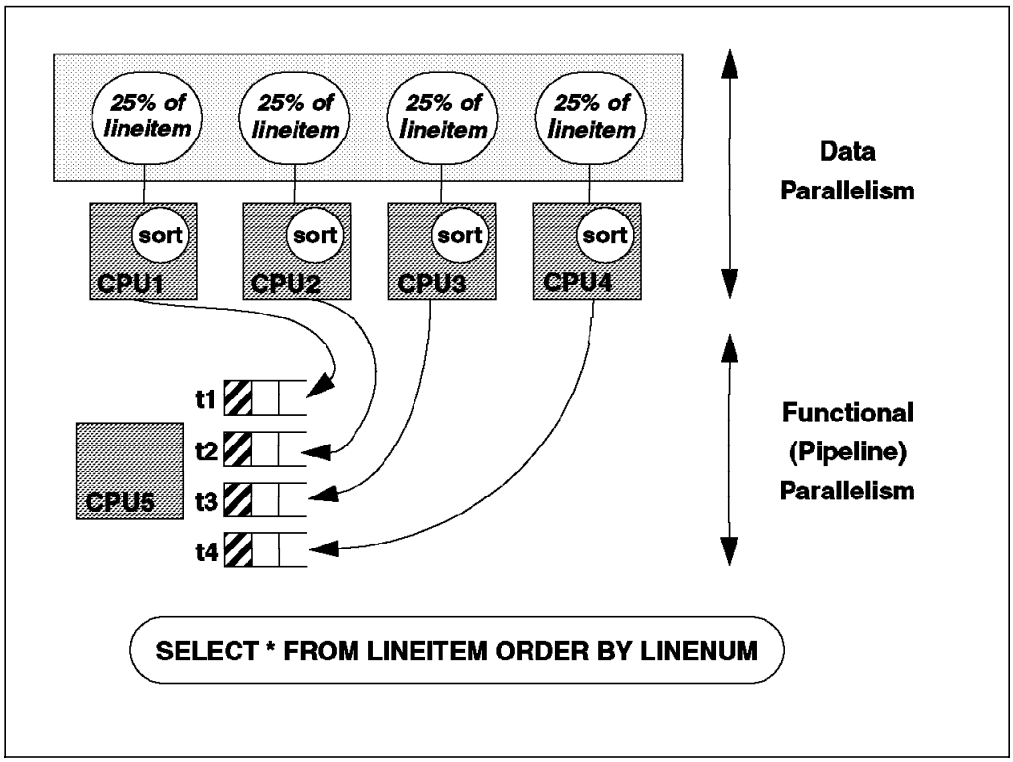


Figure 28. Data and Functional Parallelism

2.3.1 Configuration Parameters Affecting Degree of Parallelism

There are many configuration parameters that affect the degree of parallelism used in an application. First, we will discuss the parameters, then we will explain how DB2 decides which degree of parallelism to use.

Instance Level: There are two parameters in the database manager configuration that affect parallelism:

- **INTRA_PARALLEL** - Can be set to either ON or OFF. In a Uni-processor environment the INTRA_PARALLEL defaults to OFF at install time. If INTRA_PARALLEL flag is ON, parallelism is allowed for that DB2 instance. If this parameter is changed, all bound packages will be marked as invalid and will be implicitly rebound on their next execution.
- **MAX_QUERYDEGREE** - Range is from 1 - 32767. The default is ANY, which allows the optimizer to choose the optimal degree of parallelism based on the SQL statements, the number of CPUs and the data cardinality. This sets the maximum degree of parallelism for the instance. No SQL executed on databases within the instance can have a degree of parallelism greater than this setting.

Database Level: There is a parameter in the database configuration file that affects parallelism:

- **DFT_DEGREE** - Allowed values: 1- 32767, ANY. The default is 1. That is, no parallelism. This provides the default value for the DEGREE option on bind as well as setting the default for the CURRENT DEGREE special register. If an application connects to two or more databases, the DFT_DEGREE for each database controls the degree of parallelism. For example, the queries

accessing database1 may have a degree of parallelism of 2, while the queries accessing database2 may have a degree of parallelism of 4.

Static and Dynamic SQL: The following are options to control the degree of parallelism in static and dynamic SQL:

- **CURRENT DEGREE** - Allowed values: 1 - 32767, ANY.

  The default is the DFT_DEGREE setting. This sets the degree of parallelism that is used when executing dynamic SQL. The value can be changed for different SQL statements within a dynamic SQL program. To change its value use:

  ```
  SET CURRENT DEGREE='X'
  ```

  where X is between 1 and 32767.

- **DEGREE BIND OPTION** - Allowed values: 1 - 32767, ANY.

  The default is the DFT_DEGREE setting. This sets the degree of parallelism for all static SQL statements within the package being bound to the database. To change its value use:

  – For prep and bind in one step:

    ```
    prep program.sqc degree X
    ```

  – For prep and bind in two steps:

    ```
    prep program.sqc bindfile
    bind program.bnd degree X
    ```

    where X is between 1 and 32767, or ANY.

One important point that needs to be considered in the degree of parallelism is that INTRA_PARALLEL must be ON in order for the optimizer to break sections into SSPs and distribute them among the processors. Also, the degree of parallelism for a package, dynamic SQL statement or executing package can be limited by the MAX_QUERYDEGREE parameter.

Active Applications: This parameter affects the degree of parallelism for active applications:

- SET RUNTIME DEGREE - Allowed values: 1 - 32767, ANY.

  Sets the maximum degree of parallelism for an active application. If an active application is executing with a degree of parallelism less than this setting, nothing happens. If the application has a degree of parallelism higher than this setting, the application will have its degree of parallelism reduced for all queries within the application. This does not affect currently executing queries and will only affect queries in the application started after the SET RUNTIME DEGREE command has been executed. If a SYSADM or DBADM user notices performance degradation due to an application using a degree of parallelism inappropriate for the environment, he/she can set this value to reduce the number of SSPs (and thus DB2 agents) created by the optimizer for each query within the application. To use this option, you must specify the application ID of the application on which to reduce the degree of parallelism. For example:

  ```
  LIST APPLICATIONS
  SET RUNTIME DEGREE FOR 350 to 4
  ```

CLI Applications: This parameter affects the degree of parallelism for CLI applications:

- DB2DEGREE keyword in db2cli.ini - Allowed values: 1 - 32767, ANY

  If the value specified is anything other than 1 (the default), then DB2 CLI will issue the following SQL statement after a successful connection: SET CURRENT DEGREE value. This specifies the degree of parallelism for the execution of the SQL statements. The database manager will determine the degree of parallelism if you specify ANY.

Which Degree of Parallelism is Used?: As we have shown, there are a number of parameters that can influence the degree of parallelism that is used when executing SQL queries. The actual degree of parallelism used is determined as:

If INTRA_PARALLEL is set to NO:

- degree of parallelism = 1

If INTRA_PARALLEL is set to YES:

- degree of parallelism is the lower of:
  - MAX_QUERYDEGREE
  - DFT_DEGREE
  - CURRENT_DEGREE (Dynamic SQL)
  - DEGREE BIND OPTION (Static SQL)
  - RUNTIME DEGREE (if set)

For an MPP environment, the DB2 optimizer makes some basic assumptions and adopts some basic rules. The optimizer assumes that the database nodes are roughly equal. Since it cannot determine the exact configuration (number of CPUs, memory, I/O speed, buffer pool settings and so on) for each system, it assumes they are all the same. DB2 uses the coordinator node as the model for the system and uses a degree of parallelism based on the configuration of the coordinator node.

Early tests have demonstrated that using a degree of parallelism of one less than the number of CPUs in an SMP machine can lead to optimal response times for complex queries.

Table 2 on page 47 summarizes all the different parameters which affect the degree of parallelism:

PARAMETER	VALUES
INTRA_PARALLEL (dbm level)	• ON/OFF • Defaults to OFF on uni-processor • Defaults to ON on SMP machine • If changed, packages already bound will automatically be rebound at next execution.
MAX_QUERYDEGREE (dbm level)	• 1-32767,ANY • Defaults to ANY - allows optimizer to choose degree of parallelism based on cost. • No SQL executed on a database in this instance can use a degree of parallelism higher than this value.
DFT_DEGREE (db level)	• 1-32767,ANY • Defaults to ANY - allows optimizer to choose degree of parallelism based on cost. • No SQL executed on a database in this instance can use a degree of parallelism higher than this value.
CURRENT DEGREE (Spec. Register)	• 1-32767, ANY • Sets degree of parallelism for dynamic SQL • Defaults to DFT_DEGREE
RUNTIME DEGREE (Spec. Register)	• 1-32767,ANY • Sets degree of parallelism for running applications • To change: SET RUNTIME DEGREE FOR (100) to 4 • Only affects queries issued after SET RUNTIME is executed.
DB2DEGREE (CLI Keyword)	• 1-32767,ANY • *Default is 1* • Sets degree of parallelism for CLI applications • CLI application issues a SET CURRENT DEGREE statement after database connection

Table 2. Parameters Affecting Degree of Parallelism

2.3.2 Process Model on SMP

The process model as shown in Figure 29 on page 48 represents the communication that occurs between database servers and client and local applications. It ensures that database applications are isolated from critical database resources such as database control blocks and critical database files. In Intel operating systems such as OS/2 and Windows NT, communication occurs via threads; in UNIX-based systems, via processes. For each database being accessed there are various processes or threads started to deal with the various database tasks (for example, logging). There is also a one-to-one mapping of the processes or threads of client applications to coordinating agents that operate on a database. A coordinating agent works on behalf of an application, and communicates to other agents/subagents using Inter-Process Communication (IPC) techniques or remote communications protocols. The

firewall is used to protect the database and the database manager from applications, unfenced stored procedures and user-defined functions (UDFs). A firewall maintains the integrity of the data in the databases, because an application programming error cannot overwrite an internal buffer or file of the database manager. It also improves the reliability of the database manager, because an application programming error cannot crash the database manager.

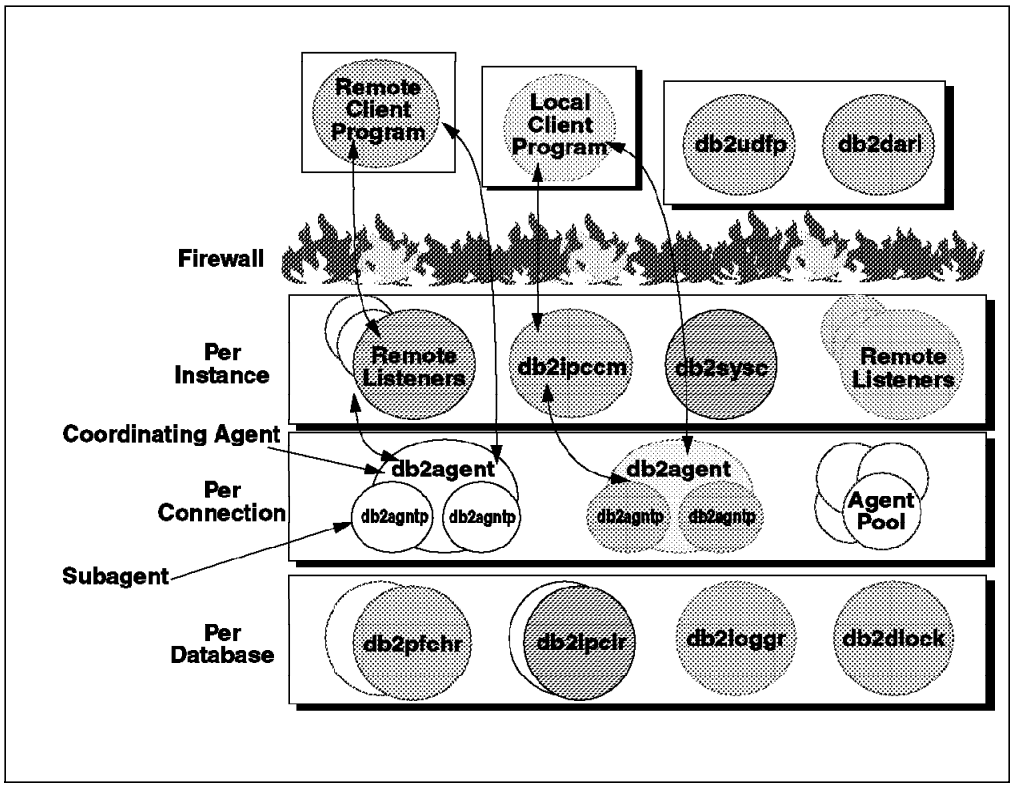


Figure 29. Process Model on SMP

In DB2 UDB V5, the process model used in DB2 Common Server has been integrated with the parallel model used in DB2 Parallel Edition. When client programs connect, a remote listener (or IPC listener) allocates a coordinating agent (db2agent) to represent the user's application within the database manager. The coordinating agent will perform any work or arrange another process or thread to perform the work on its behalf. All data and return code information is passed back to the user's application via the coordinating agent. There is a pool of spare or idle agents kept by DB2 to allow for the quick allocation of new agents for an application. When SQL requests are passed to the coordinating agent which involve parallel pieces the coordinating agent will pass the work out to a number of subagents (db2agntp) to perform this work on its behalf. Since each subagent is a separate process or thread, they can be executing simultaneously if there are adequate CPUs in the machine to support the number of subagents.

2.3.3 Load Parallelism

The Load utility can now take advantage of both CPU and DISK parallelism to improve the speed of loading the data, as shown in Figure 30 on page 49. The CPU_PARALLELISM parameter controls the number of sub-agents used for data parsing and sorting. The DISK_PARALLELISM parameter controls the number of I/O writers used to build the table within the table space containers.

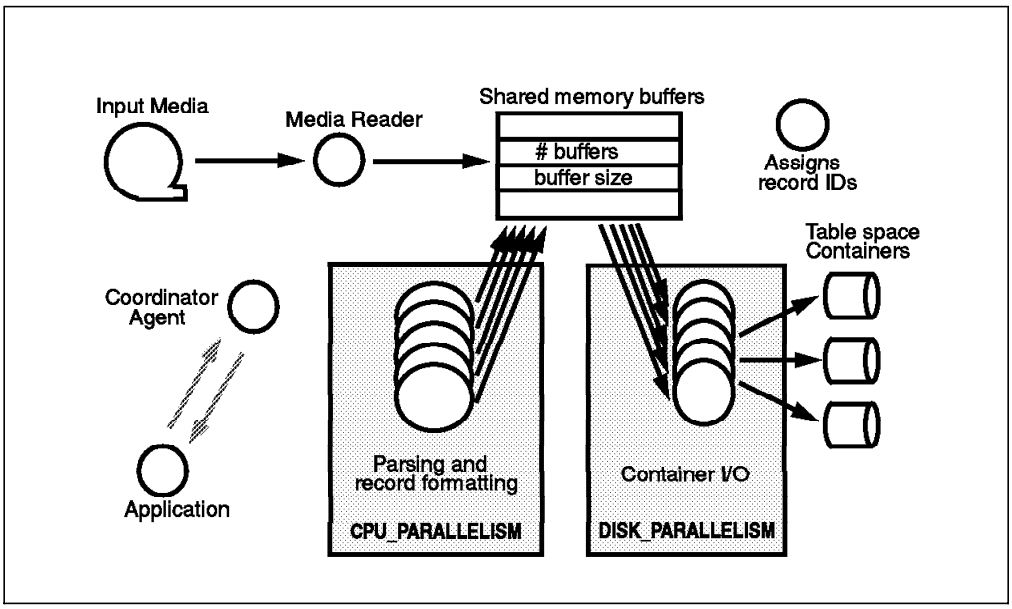


Figure 30. Load Parallelism

Load can also take advantage of disk parallelism on its input since it can process input data from multiple locations.

If the ANYORDER modifier on the Load command is used, then the preservation of source data order is not maintained when loading rows, yielding significant additional performance benefit on SMP systems. For more information on ANYORDER. see 2.2.3.1, “Load Performance Options” on page 28.

2.3.4 Index Create Parallelism

When creating an index on SMP machines, DB2 will consider using multiple scan and sort operators to gather the information to build an index, as shown in Figure 31 on page 50. When used, these multiple operators will speed up the initial processing stages for the create index though the later actual index create is performed using a single process. Speeding up index create will also improve performance for the reorg utility, creation of primary and unique key constraints and database restart commands where corrupt indexes are involved.

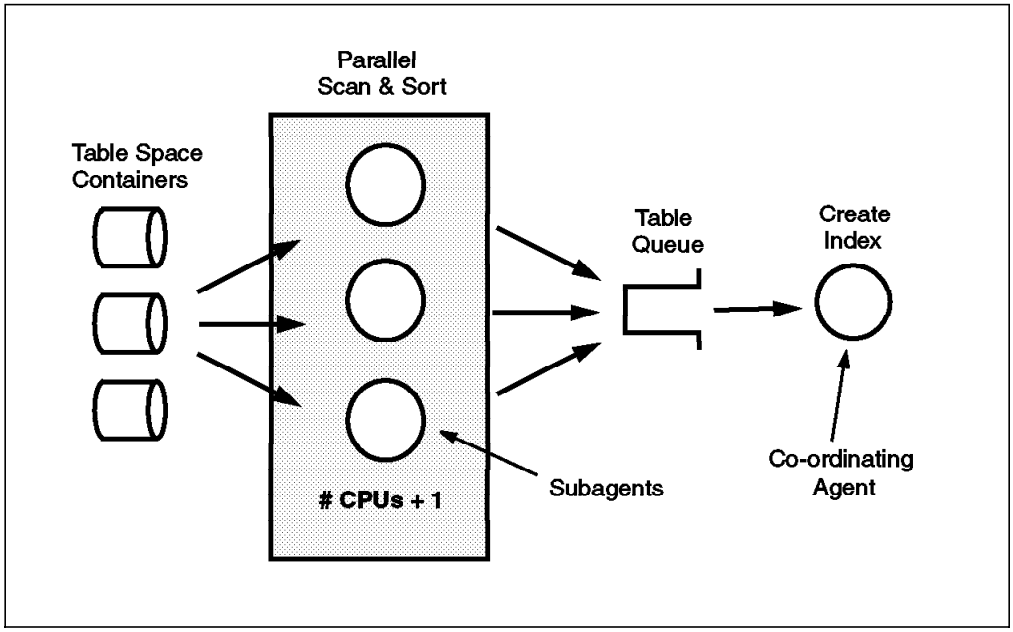


Figure 31. Index Create Parallelism

When creating an index, if INTRA_PARALLEL is ON, and the table is large enough to benefit from the use of multiple processes or threads, then the degree of parallelism used will be the number of CPUs plus one. The actual processing will be performed using a coordinating agent and subagents similar to an SQL request. The coordinator dispatches a number of subagents that scan the data in parallel. Each subagent performs an independent sort of the data it reads. The subagents then insert the data, in sorted order, into a sorted table queue. The sorted table queue, consolidates the data from the subagents and gives it to the coordinator agent in sorted order. The coordinator agent then builds the index using the sorted data.

The degree of parallelism used by create index is not limited by the MAX_QUERYDEGREE configuration parameter. Also, the degree of parallelism is not affected by the DEGREE option of the `Bind` command or the CURRENT DEGREE special register.

2.3.5 Backup Parallelism with SMP

During backup processing, DB2 can take advantage of running the buffer manipulators in parallel to read the data from disk into the memory buffers, as shown in Figure 32 on page 51. The media writers can subsequently run in parallel, reading data from the memory buffers and writing to the backup devices if the backup devices are not on the same device.

INTRA_PARALLEL must be set ON.

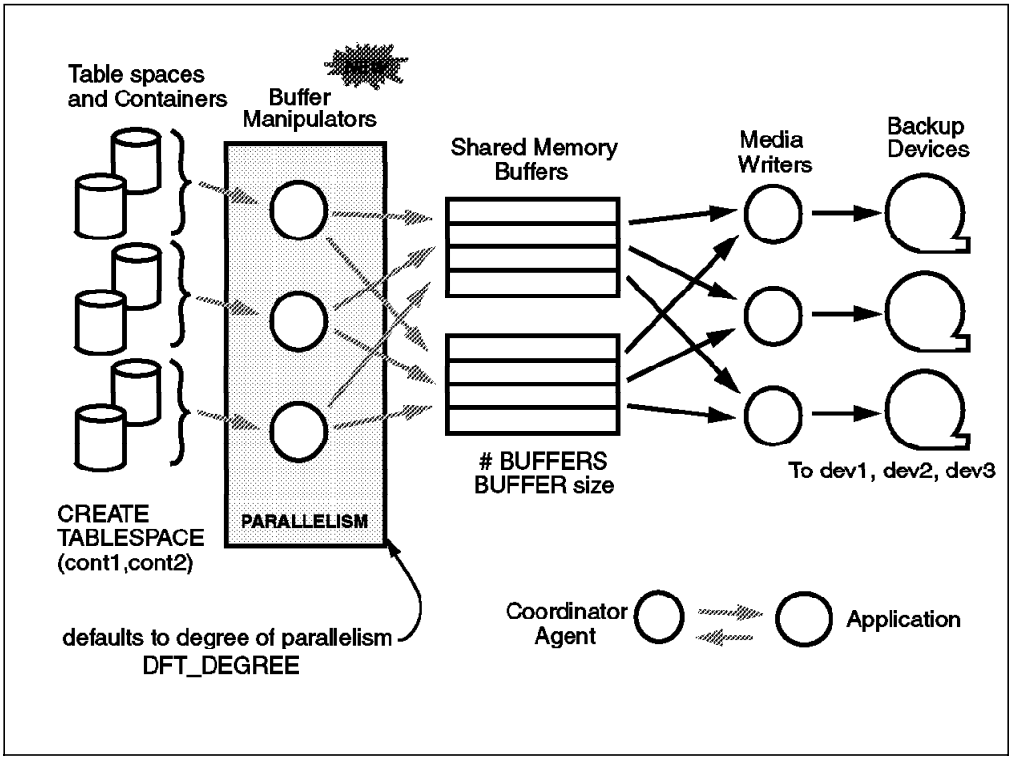


Figure 32. Backup Parallelism

2.3.6 Restore Parallelism with SMP

During restore processing DB2 can take advantage of running the media readers in parallel if the backup is on multiple disks/devices, as shown in 51. The buffer manipulators can subsequently be run in parallel reading from the memory buffers and writing to the database.

INTRA_PARALLEL must be set ON.

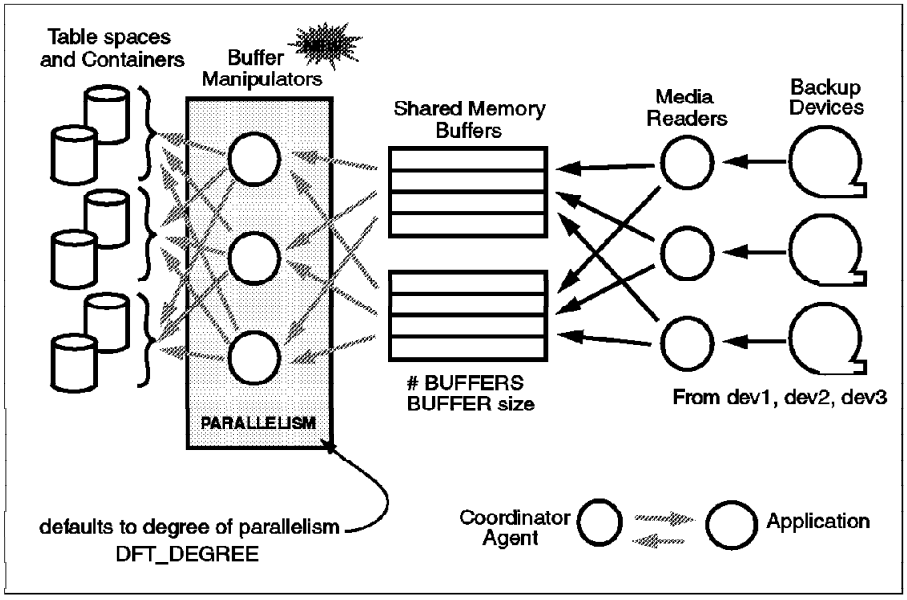


Figure 33. Restore Parallelism

2.4 Performance Features and Enhancements

DB2 UDB V5 introduces a number of features to improve the performance of utilities and applications. These features are discussed in the following sections:

- 2.4.1, “Dynamic Bitmap Index ANDing”
- 2.4.2, “STAR Join” on page 55
- 2.4.3, “OLAP SQL” on page 57
- 2.4.4, “Meta-Optimizer” on page 62
- 2.4.5, “Communications Costing” on page 63
- 2.4.6, “Outer Joins” on page 63
- 2.4.7, “Partitioned Database Join Strategies” on page 64
- 2.4.8, “Package Caching” on page 66
- 2.4.9, “Deferred Prepare” on page 68
- 2.4.10, “Smartguide for Performance” on page 70

2.4.1 Dynamic Bitmap Index ANDing

Dynamic Bitmap Index ANDing will improve the performance of queries with multiple ‘AND’ predicates that are applied against columns from a single table for which multiple indexes exist on the predicates.

The selection of Dynamic Bitmap Index ANDing in an access plan is a cost based decision taken by the optimizer.

Dynamic Bitmap Index ANDing will perform multiple index scans, each scan identifying a set of qualifying RIDs that meet the criteria for that particular predicate. The qualifying RIDs for each scan are hashed to a vector in a bitmap.

The hashing of RIDs to the bitmap is done on a ‘reducing’ basis to reduce memory requirements. This means that there are usually less vectors in the bitmap than qualifying rows. The algorithm used for this hashing is based on the expected cardinality of the index column. Because the RIDs are hashed to the bitmap on a reducing basis, it is possible for two or more RIDs to be hashed to the same position in the bitmap.

For the second and subsequent index scans, the qualifying entries in the bitmap will be checked against the value in the corresponding position in the bitmap created by the previous scan. If this value indicates a qualifying RID, then it can be carried forward to the bitmap being built for the current scan. This is known as ‘probing’ the previous bitmap. This process has the effect of reducing the amount of qualifying positions in the bitmap for each scan performed.

Dynamic Bitmap Index ANDing makes use of the Sort Heap for the Dynamic Bitmap creation. The amount of Sort Heap used varies based on the cardinality of the index scan (ISCAN) in the Index ANDing plan. This will range from 6 KB for ISCANs returning few rows to 24 MB for ISCANs returning 6 million or more rows.

Locking is only performed during this process when the actual qualifying rows are being fetched from the table. Therefore, it is possible that rows inserted during this query that match all the criteria of the predicates will not be detected. The exception to this situation is when an isolation level of repeatable read (RR) is used. Locks will then be held that prevent this occurrence. However, the penalty for using RR is a reduction of concurrency for this table.

Rows deleted during the execution of this query will be detected as the predicates are re-evaluated against the qualifying rows being fetched from the table.

Dynamic Bitmap Index ANDing is not used for index-only access.

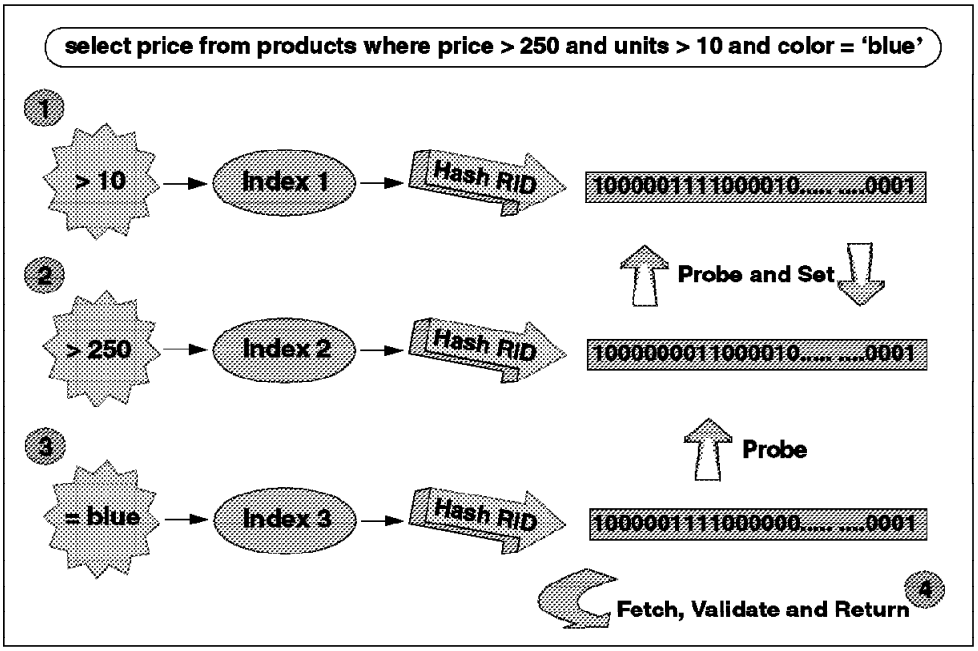


Figure 34. Dynamic Bitmap Index ANDing

Figure 34 illustrates how Dynamic Bitmap Index ANDing works in a query where three predicates are used in a query. These are the steps that are performed:

1. First Index
   - Scan first index.
   - Hash each qualifying RID (where units > 10) to generate a vector into the bitmap.
   - Set (make equal to 1) the corresponding bit in the bitmap.
   - Repeat this for the rest of the first index scan.
2. Second Index
   - Scan second index.
   - Hash each qualifying RID (where price > 250) to generate a vector into the bitmap.
   - Test corresponding bit in the previous bitmap and if set, set the corresponding bit in the current bitmap.
   - Repeat this for the rest of the second index scan.
   - For all but the last index scan, repeat this process (hash, probe and set).
3. Last Index
   - Scan last index.
   - Hash each qualifying RID (where color = 'blue') to generate a vector into the bitmap.
4. Fetch, Validate and Return
   - Test corresponding bit in previous bitmap and if set, return the RID so that the qualifying row of data can be fetched.

- Validate that the row qualifies for all the predicates.
- Repeat this for the rest of the last index scan.

2.4.2 STAR Join

The join pattern of this kind of OLAP or data warehouse query resembles a star structure with a large, central (fact) table joining to a number of other smaller (dimension) tables as shown in Figure 35.

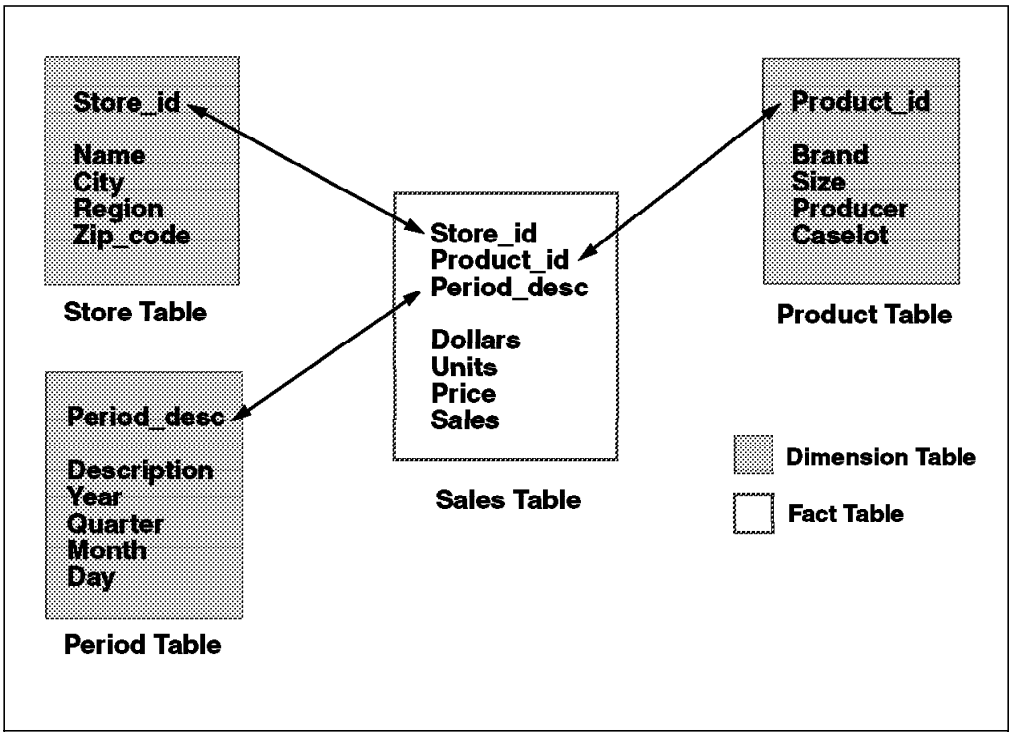


Figure 35. Star Join Visualization

The join predicates are usually between primary keys in the dimension tables and foreign keys in the fact table.

2.4.2.1 Typical OLAP or Data Warehouse Query

1. Queries involve joining Dimension Tables with Fact Table

   A typical OLAP or data warehouse query is a multi-way join between several small dimension tables and a large fact table.

2. Predicates applied to Dimension Tables

   Predicates are applied to the Dimension tables. The Dimension tables are joined to the Fact table using a *semi-join* to act as filtering.

3. Aggregation(s) applied to Fact Table

   When all joins have been 'tried', qualifying RIDs from the Fact table are returned and fetched from the table itself. Aggregation is applied to the fact table at this point. The joins are then completed by joining the fetched rows with the dimension tables.

This type of join is difficult to execute efficiently because, although the combination of join predicates results in a small answer set, no single join predicate reduces the cardinality of the fact table rows significantly. The greater

the number of dimension tables involved in the join, the more potential exists to plan an inefficient access path.

Star Join Example: Find 1995 and 1996 sales by store of products manufactured by Tetra for stores in regions NE or NW:

```
select
    st.name, pe.year, sum(sa.units) as sales
from
    store st, sales sa, period pe, product pr
where
    st.store_id = sa.store_id
and pr.product_id = sa.product_id
and pe.period_desc = sa.period_desc
and pe.year between 1995 and 1996
and pr.producer in ('TETRA')
and st.region in ('NE','NW')
group by
    st.store_id, pe.year
```

Figure 36 illustrates the technique of Dynamic Bitmap Index ANDing to process this query.

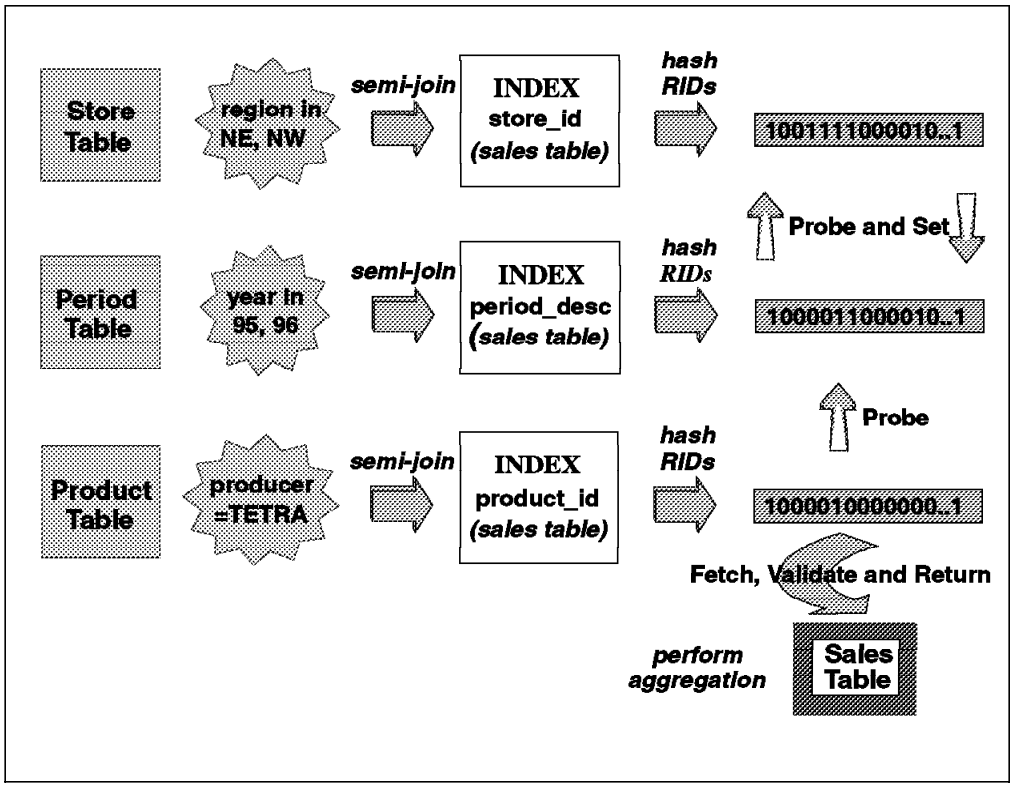


Figure 36. Star Join via DBIA

Dynamic Bitmap Index ANDing is a join technique used by the optimizer to efficiently process a star join query. The general process of dynamic bitmap index ANDing is explained in the previous pages.

The difference when this technique is applied to a star join is that the qualifying RIDS are obtained by joining a dimension table to an index of the fact table. The fact tables's RIDs are then used in the hashing process to build the bitmaps.

Each further join of a dimension table to the fact table index then produces a bitmap that contains qualifying RIDs from that join; these RIDs have also been ‘probed and set’ from the previously created bitmap.

The last join of a dimension table to the index of the fact table then returns RIDs that can be used to access qualifying rows on the fact table. The final result set is created by joining these fetched rows to those of the dimension tables.

2.4.3 OLAP SQL

Support has been added for the OLAP functions: ROLLUP, CUBE, GROUPING SETS, and the GROUPING column function.

2.4.3.1 ROLLUP

A ROLLUP grouping is an extension to the GROUP BY clause that produces a result set that contains ordered super-aggregate rows in addition to the ‘regular’ grouped rows. Figure 37 shows the syntax used by ROLLUP.

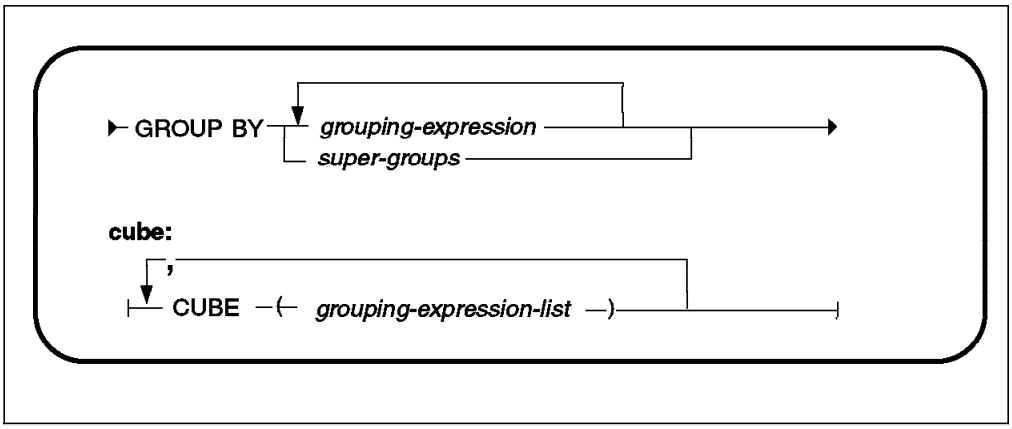


Figure 37. Rollup Syntax

Ordered super-aggregate rows are rows that contain further aggregates whose values are derived by applying the same column functions that were used to obtain the grouped rows. This produces a series of groupings in the result set that contain different levels of aggregation.

In summary, the result set contains regular grouped rows and the results of these groupings re-applied up to a grand total level.

The result set produced by a ROLLUP function is constructed by one pass of the data.

2.4.3.2 CUBE

A CUBE grouping is a further extension to the GROUP BY clause that produces a result set that contains all the rows of a ROLLUP aggregation and, in addition, contains symmetric super-aggregate rows. Figure 38 on page 58 shows the syntax used by CUBE.

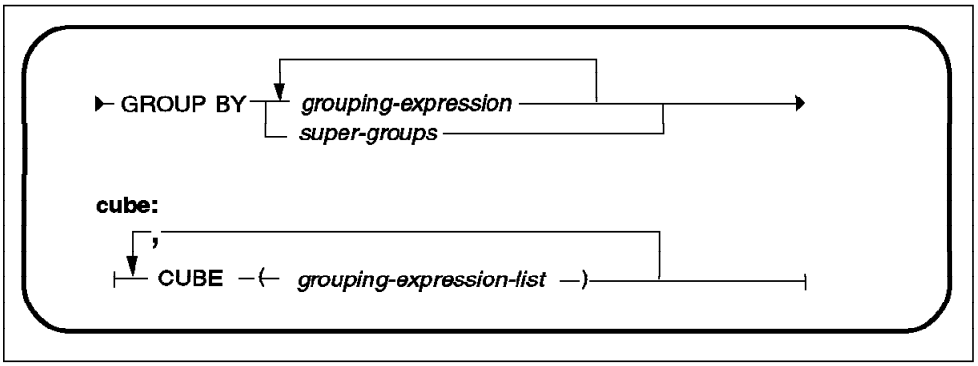


Figure 38. Cube Syntax

The aggregations in the symmetric super aggregate rows are often known as *cross tabulation* aggregates.

This cross tabulation in the answer set allows the result set to be *sliced* and *diced* in multi dimensions in order to extract the required information at various levels.

Example of Rollup and Cube: 58 shows an example of the output produced by using the ROLLUP and CUBE groupings.

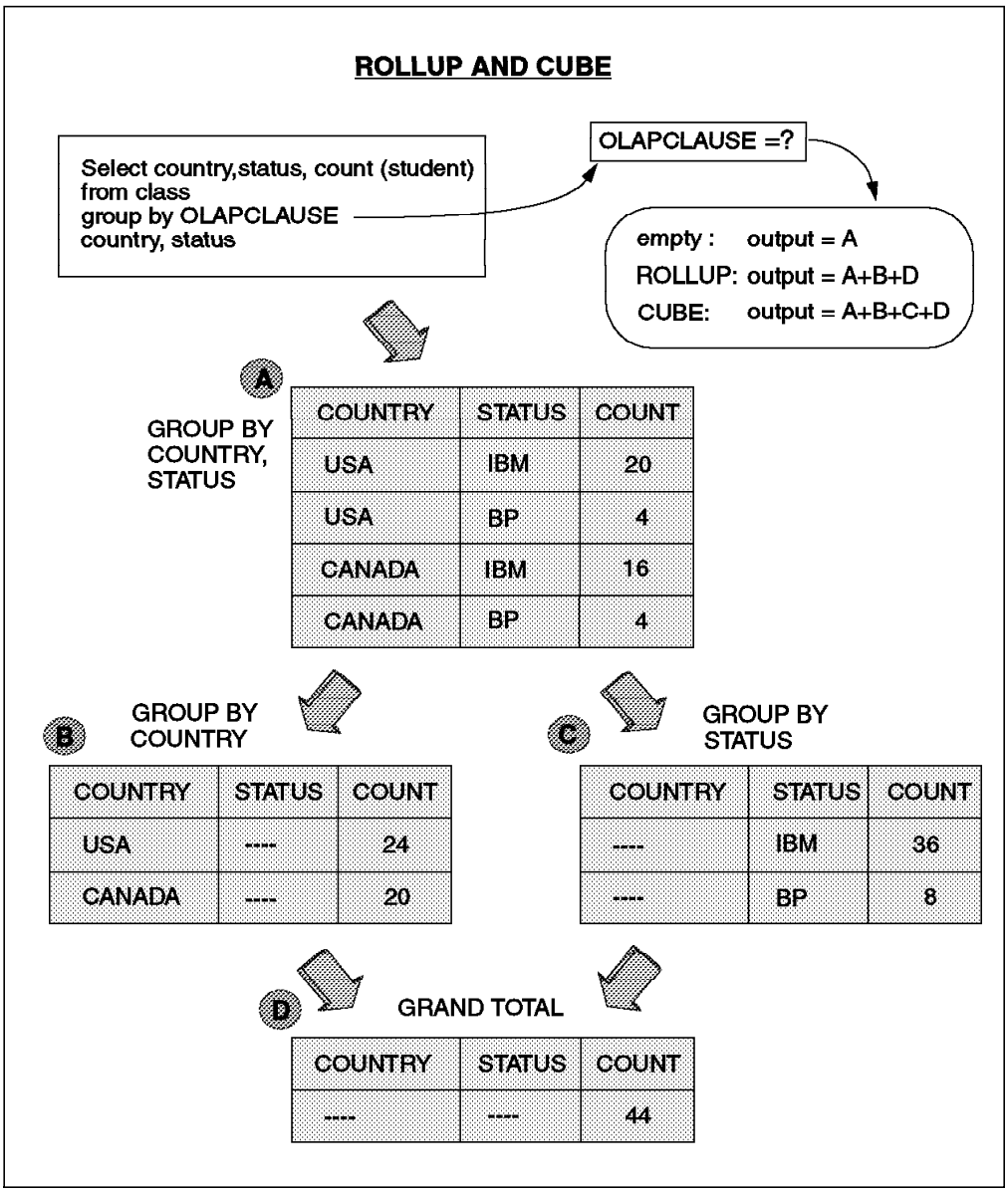


Figure 39. Example of Rollup and Cube

In this example, the result rows represented by B and D are known as *super-aggregate* rows, and those by C as *symmetric super-aggregate* rows.

Another way to express the output from ROLLUP and CUBE is as follows:

If a query has GROUP BY 1,2,3 in the group by clause, then:

- Rollup will perform aggregation for:
  - GROUP BY 1,2,3
  - GROUP BY 1,2
  - GROUP BY 1
  - GRAND TOTAL (or no GROUP BY clause)

  So, progressive groupings are performed by eliminating the last column in the GROUP BY clause until the grand total.

- CUBE will perform aggregation for:
  - GROUP BY 1,2,3
  - GROUP BY 1,2
  - GROUP BY 1,3
  - GROUP BY 2,3
  - GROUP BY 1
  - GROUP BY 2
  - GROUP BY 3
  - GRAND TOTAL (or no GROUP BY clause)

  So groupings are performed for all possible combinations of the columns in the GROUP BY clause for 3,2 and 1 columns followed by the grand total.

2.4.3.3 Grouping Sets

Grouping Sets are used to construct a query that pastes together two or more groups of rows into a single result set. Figure 40 shows the syntax used by GROUPING SETS.

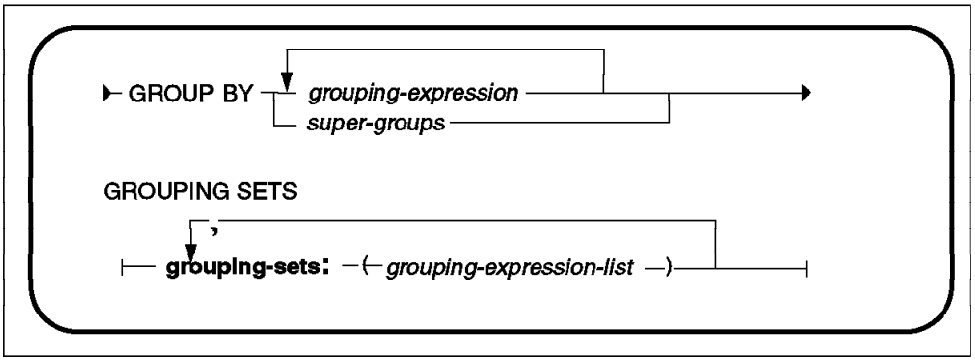


Figure 40. Grouping Sets Syntax

The individual groups that are pasted together can be any combination of regular grouping expression lists, ROLLUP aggregations, CUBE aggregations, grand-total aggregation or concatenations.

Figure 41 on page 61 shows an example of the output produced by using GROUPING SETS.

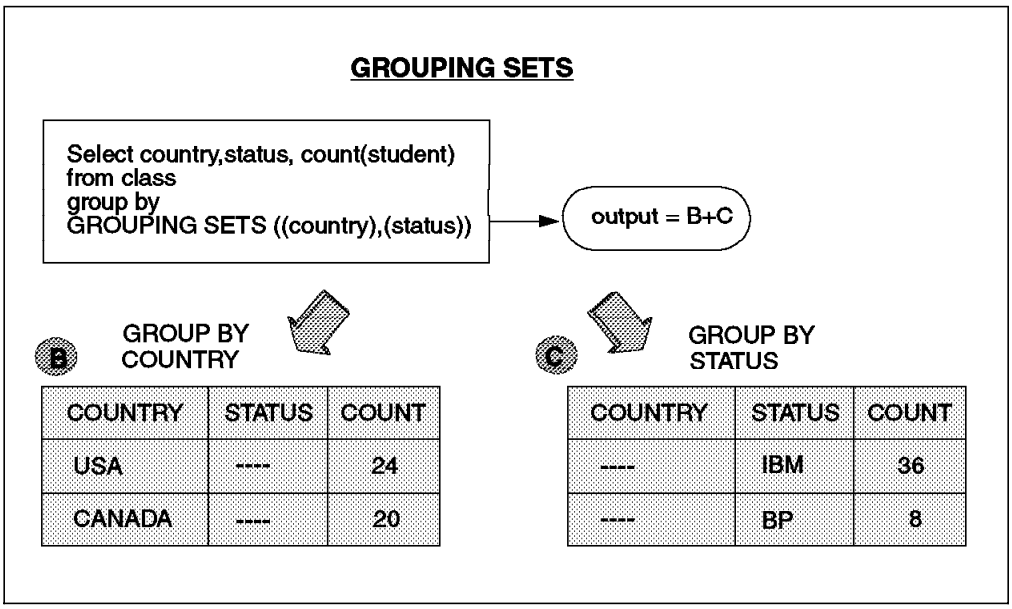


Figure 41. Example of Grouping Sets

2.4.3.4 Grouping Column Function

The purpose of the grouping column function is to return a value that can be used to determine whether or not a value is a null as a result of a GROUP BY (column function or extension thereof such as ROLLUP or CUBE).

The returned value is an integer of value 1 or 0.

- 1 - The value is a special null value that represents either the set of all values used in an aggregation, or the value resulting from the combining together of grouping sets.
- 0 - Otherwise

The main benefit of this column function is for the interpretation of the answer sets. It allows the user to identify which rows have been generated by a GROUP BY function.

This is of particular value to graphical tools in order to allow the interpretation of answer sets to be displayed in a manner which visualizes the relationships between result set rows.

Figure 42 on page 62 shows an example of the output produced by using the GROUPING column function.

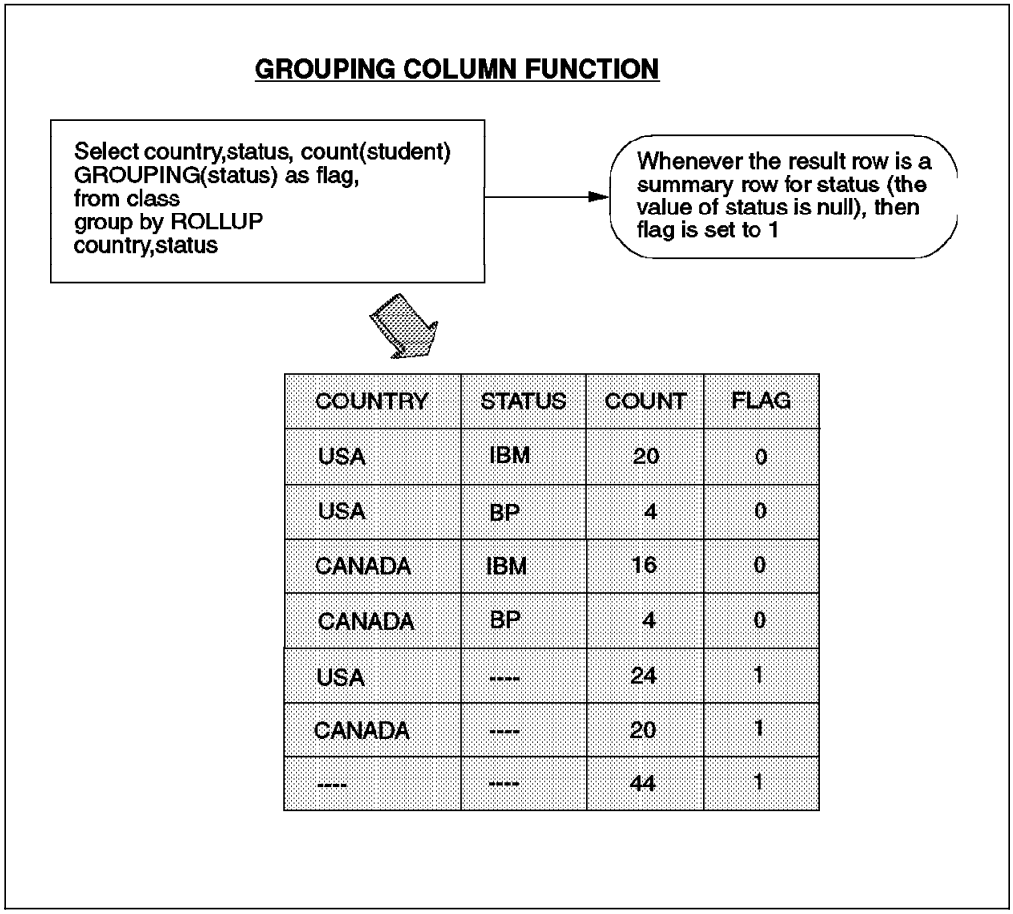


Figure 42. Example of the Grouping Column Function

2.4.4 Meta-Optimizer

The meta-optimizer can be thought of as a *pre-optimizer* in that it takes an initial look at the query, data, and statistical information and then does the following:

- Estimates the cost of the query. If a low level, quick optimization is performed.
- Estimates the cost of the query. If a high level, thorough optimization is performed.

If the computed costs are close, the quick optimization is performed. If there is a significant difference between the estimated costs, the thorough optimization is done. This allows the optimizer to handle queries in a more flexible manner and to be able to handle a wide variety of queries from easy to complex.

For the meta-optimizer to work, the query optimization level must be set to 5. The default query optimization level for a database is specified by the DFT_QUERYOPT configuration parameter. The default setting for this parameter is 5.

2.4.5 Communications Costing

CommCost is a new optimizer component of cost added to DB2 UDB for MPP environments only. It will always have a value of zero in any other environment.

CommCost will represent an estimate of the number of IP frames transferred between all nodes.

Communications costs will be converted from IP frames to milliseconds by a weight factor in order to compare them to TotalCost. As with DB2 Common Server V2, TotalCost will represent CPU and I/O resources consumed in milliseconds (on one node).

The weight factor represents the number of milliseconds to transfer an IP frame between nodes. This weight factor will be stored as a new database configuration parameter called COMMSPEED.

When a MPP parallel database instance is installed or migrated, the configuration parameter COMMSPEED will be set to an appropriate default. Typically, COMMSPEED will be set to one of two defaults depending on the availability of a high speed switch between nodes. In a non-MPP environment, COMMSPEED will be set to 0.

By changing the value of COMMSPEED, it is possible to model the communications costs of protocols other than TCP/IP.

The overall cost of a communications bound plan will be CommCost * COMMSPEED. The overall cost of a CPU/IO bound plan will be TotalCost. The optimizer will select the plan with the lowest overall cost, where:

```
overall cost = max((CommCost * COMMSPEED), TotalCost)
```

The overall cost is not recorded in the plan itself. However, it will be derived and displayed in the SQLCA and by the explain tools.

2.4.6 Outer Joins

An INNER join is a join between two tables which only returns result rows where there are joining rows from the tables involved in the join. The query:

```
SELECT empname,deptname FROM emp, dept INNER JOIN WHERE emp.deptno=dept.deptno
```

yields the same results as the INNER join in Figure 43 on page 64.

An OUTER join is a join between two tables that returns results rows for both matching and non matching rows in the tables involved in the join. Rows that do not match those of the other table are padded with NULLS.

Some applications of the OUTER join require that the result includes the unmatched rows of only one of the tables. Other applications require that the result includes the unmatched rows of both tables.

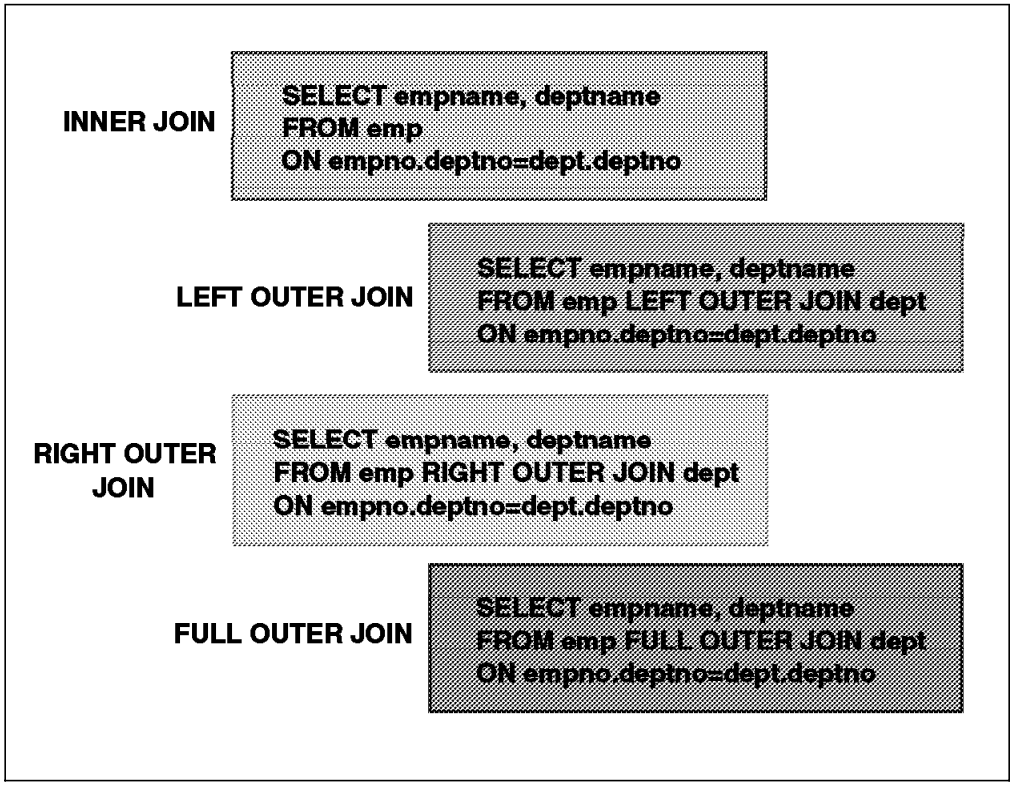


Figure 43. Inner and Outer Joins

There are several types of OUTER join:

- *Left Outer Join*

  Includes the matched rows and preserves unmatched rows of the left operand table to specify whether to display the Alert Center window on new warnings and alarms, on new alarms only, or never.

- *Right Outer Join*

  Includes the matched rows and preserves unmatched rows on the right operand table.

- *Full Outer Join*

  Includes the matched rows and preserves the unmatched rows of both.

The role of the outer join condition is similar to that of the inner join, but there is an essential semantic difference. Unlike a join condition for an inner join, the join condition of an outer join specifies a pairing that is not also a restriction. More precisely, the join condition of an outer join specifies the criteria by which rows of the two tables are paired, but unpaired rows are not eliminated from the result unless they are rows of the left operand table of a right outer join, or rows on the right operand table of a left outer join.

2.4.7 Partitioned Database Join Strategies

There are four join strategies available in DB2 Parallel Edition V1, notably:

1. Collocated Join
2. Directed Outer Join
3. Directed Inner and Outer Join

4. Broadcast Outer Join

Two new join strategies for partitioned databases are available in DB2 UDB V5, Directed Inner Join and Broadcast Inner Join.

2.4.7.1 Directed Inner Join

In this join strategy, the inner table for the join is hashed and directed to the nodes where the outer table resides.

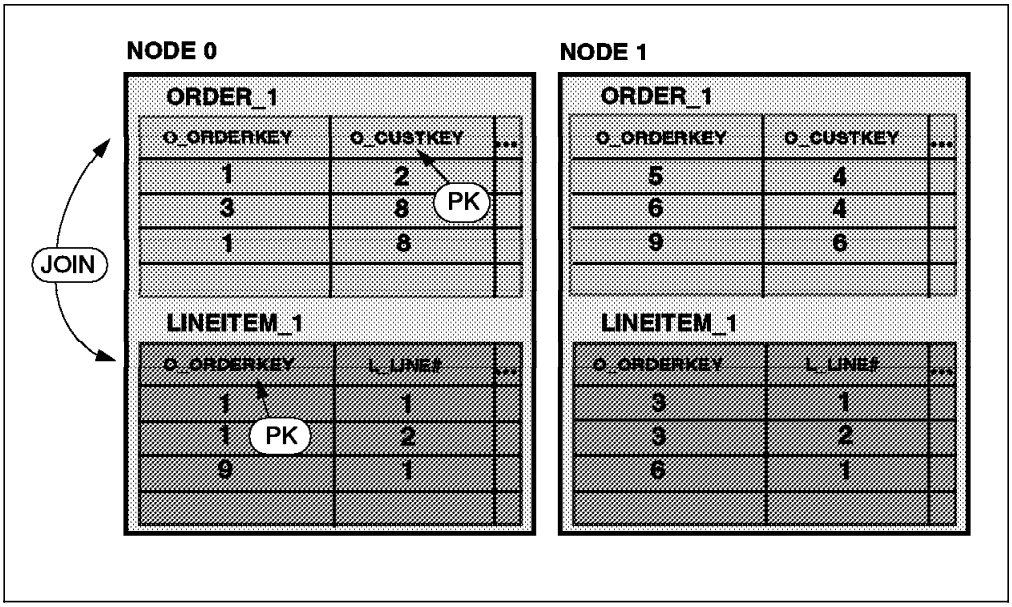


Figure 44. Directed Inner Join

Figure 44 shows a possible scenario for a Directed Inner Join.

In this example, the ORDER_1 table is partitioned by O_CUSTKEY, and the LINEITEM_1 table is partitioned by O_ORDERKEY. The tables are joined by O_ORDERKEY.

In this join strategy, the ORDER_1 table is hashed on the L_ORDERKEY value and directed to the LINEITEM_1 table after predicates have been applied and columns reduced.

A directed inner join is considered if one of the tables is partitioned on the join columns. This join strategy is equivalent to a Directed Outer Join with the inner and outer tables reversed.

2.4.7.2 Broadcast Inner Join

In this join strategy, all the rows from the inner table are broadcast to the nodes where the outer table has rows.

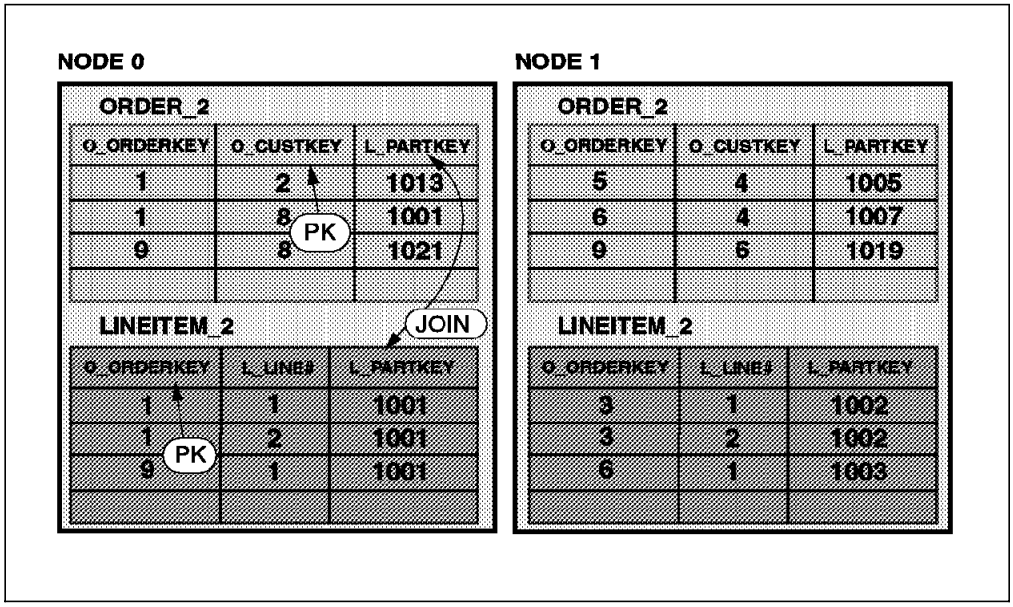


Figure 45. Broadcast Inner Join

Figure 45 shows a possible scenario for a Broadcast Inner Join.

In this example, the ORDER_2 table is partitioned by L_PARTKEY, and the LINEITEM_2 table is partitioned by O_ORDERKEY. The tables are joined on L_PARTKEY.

For a broadcast inner strategy, neither of the joining tables is partitioned by the join columns. After predicate application and column reduction, all the rows from the inner table are broadcast to the nodes where the outer table has rows.

This join strategy is equivalent to a Broadcast Outer with the inner and outer tables reversed.

2.4.8 Package Caching

There are two types of SQL, static and dynamic:

- SQL that is prepared before being run is called static SQL. Access plans for this type of SQL are stored in the database. All the access plans for one program are stored in a package. Each package is divided into sections that in most cases correspond to an SQL statement.
- Dynamic SQL is compiled immediately before run time. At run time the access plan is stored in the cache.

Both types of SQL produce an access plan in order to execute the request.

DB2 UDB uses a global SQL cache for access plans. The aim of the global SQL cache is to minimize the amount of system catalog access required for sections of static SQL statements and to maximize the sharing of sections for dynamic DML statements.

The global SQL cache is composed of two parts:

- Static SQL cache:

  This part of the global cache will contain the package frameworks and sections for any static SQL statements in a package.

- Dynamic SQL Cache

  This portion of the cache is dedicated to containing sections for dynamic SQL statements.

The total amount of memory available for the global SQL cache is determined by the value of the existing PCKCACHESZ configuration parameter.

Sections of packages in the global SQL cache reflect the status of the database. If objects (tables, indexes, aliases, and so forth) in the database are changed, access plans may be invalidated. If this happens, then all of sections of the invalidated access plans resident in the cache are flushed from memory.

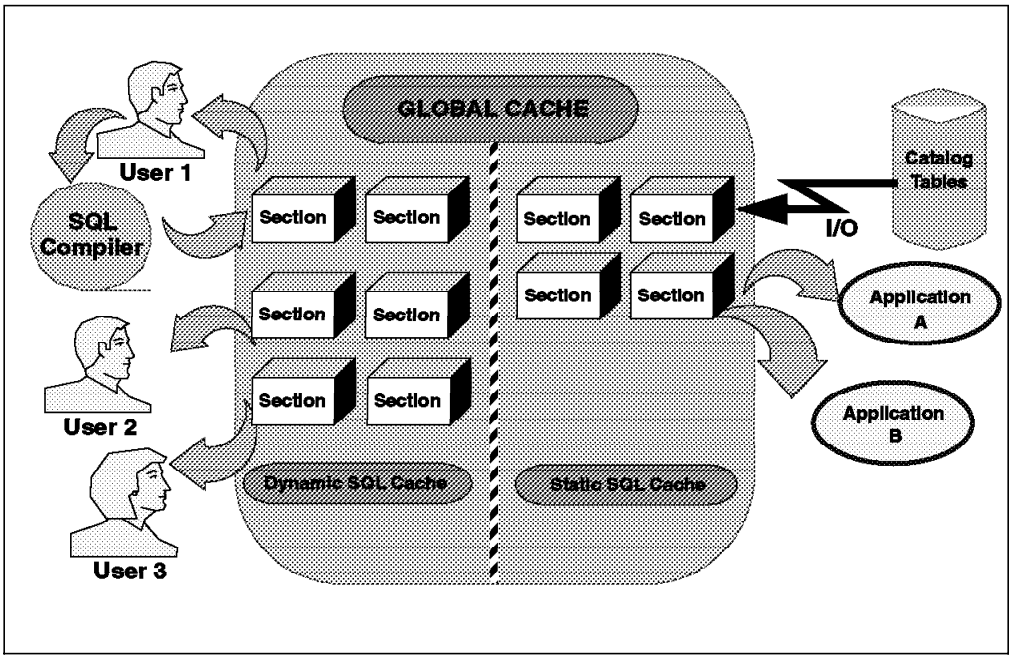


Figure 46. Package Caching

Figure 46 shows the benefit derived by users 2 and 3 who are able to reuse a section resident in the global SQL cache that had been previously compiled. User 1, on the other hand, has created an SQL request that does not already exist in the global SQL cache. Therefore, the request has to go through the process of being compiled and written to the global SQL cache before it can be used.

Also shown in the illustration is the benefit that Application A and Application B derive from using a section already in the global SQL cache. This prevents them from incurring the overhead of the I/O associated with reading a section into the global SQL cache from the system catalog tables prior to use.

Benefits

- The sections of packages remain resident in the cache until they are removed because space is required for more current processing of packages. This allocation of memory is done via an LRU (least recently used) algorithm.
- When the compiler runs into an identical statement in the same section, it will recognize that the plan has already been produced and fetch the information from the cache for reuse.

- Package and section information will be loaded into the cache from the system catalogs as required in response to requests for either dynamic or static SQL sections.
- By maximizing the reuse of information in the cache, the I/O overhead required to access sections of packages can be reduced.
- By making the caching area global instead of per user, the probability of reuse is increased leading to improved performance when executing dynamic or static SQL.

Monitoring: To aid in the tuning of the global SQL cache, the following database monitor statistics are available:

- Global SQL Cache (pckcachesz)
- Current space used (4 K pages)
- Current utilization (%)
- Number of Global SQL Cache inserts
- Number of Global SQL Cache lookups
- Static SQL Cache Hit Ratio (%)
- Dynamic SQL Cache Hit Ratio (%)

2.4.9 Deferred Prepare

When remote access is involved, the cost of sending an SQL request/reply between a client and a remote server is quite significant and sometimes more expensive than the time spent at the database engine.

Deferred Prepare improves the performance of dynamic SQL and ODBC applications that perform queries - especially in the case where the answer set is very small and the overhead of separate request flows is not spread among multiple blocks of query data.

The response time for dynamic queries improve by saving one send/receive flow in the stand-alone case, and two send/receive flows in the DDCS gateway case. The response time is around 20% faster for local access. For remote access, response times are close to 100% faster for short client/server queries on slow links, and less noticeable on fast links or for large queries that return many blocks of data.

Deferred prepare can benefit both DB2 UDB and DB2 Connect client/server communications.

How does it work?

- Deferred prepare enables performance improvements by combining the SQL PREPARE statement flow with the associated OPEN, DESCRIBE or EXECUTE statement flow. This minimizes the inter-process or network flow.
- When deferred prepare is enabled, DB2 will defer the sending out the SQL PREPARE statements until the associated OPEN, DESCRIBE or EXECUTE statement is issued by the application. For example, a PREPARE that does not have the INTO *<sqlda>* clause will be deferred until an OPEN is issued.
- In addition, a PREPARE that is not eligible for deferring could be sent right away and have its cursor opened at the same time. For example, a

PREPARE that has the INTO <sqlda> clause will also have the cursor opened if the associated OPEN statement does not have input parameters.

Precompiler Option: A new precompiler option DEFERRED_PREPARE is added to provide deferred prepare support to embedded SQL applications. The default is NO and it disables the Deferred Prepare option.

Command Line Processor: The DB2 CLP is precompiled with the DEFERRED_PREPARE_YES option to take advantage of deferred prepare.

SET CLIENT API: There is an existing API that specifies connection settings for the application. It uses the SQL_CONN_SETTING structure to specify the connection types and values. The structure and types can be found in the sqlenv.h header file and the values can be found in the sql.h header file. Refer to the API Reference for usage of this API.

The Deferred Prepare support has added a new type SQL_DEFERRED_PREPARE with the following values:

- SQL_DEFERRED_PREPARE_NO (default)
- SQL_DEFERRED_PREPARE_YES
- SQL_DEFERRED_PREPARE_ALL

Once the SET CLIENT API has executed successfully, the connection settings are fixed and can only be changed by again executing the SET CLIENT API.

ODBC/CLI Enhancements: A new configuration keyword DEFERREDPREPARE has been added to the db2cli.ini file. The value of DEFERREDPREPARE (0 or 1) controls whether or not deferred prepare will be used for ODBC/CLI applications.

The default value for DEFERREDPREPARE is 1 (turned on).

The CLI/ODBC Set Connection Option API has added a new SQL_Deferred_Prepare option:

- SQL_Deferred_Prepare_NO
- SQL_Deferred_Prepare_ALL (default)

This API allows a ODBC application to override the Deferred Prepare setting defined in the db2cli.ini file or via the ODBC/CLI Administrator Tool.

2.4.10 Smartguide for Performance

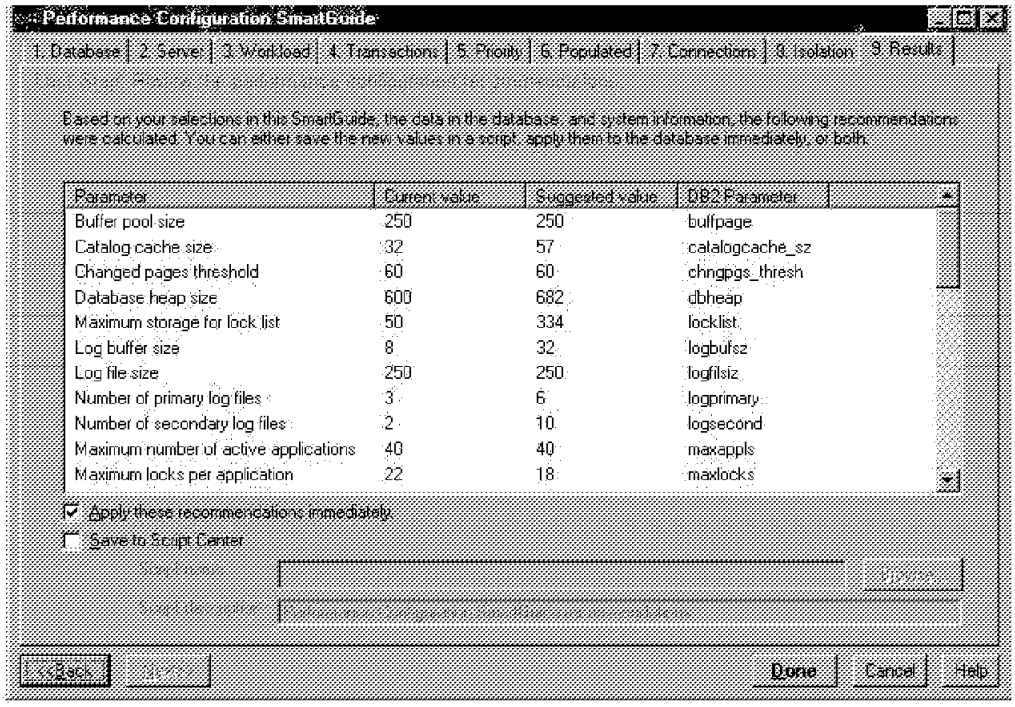


Figure 47. SmartGuide for Performance

Tuning Aid: The Performance Smartguide is a graphical tool to assist the database user in tuning the database configuration parameters.

It has a number of sections that ask questions about the environment and the applications which will be using the database. Based on the user's responses, a number of calculations will be performed and a best set of configuration parameter values assigned for the database. The user can then choose whether to apply these changes to the database or to save the changes in a script for later execution.

The changes in the parameter values made by the Performance Smartguide are not guaranteed in all cases to lead to the best possible performance. It should be used as a starting place for tuning a database's performance.

Refer to 4.4.7, "Tuning Your Database" on page 183 for a more comprehensive guide to the contents of this smartguide.

2.5 DB2 UDB V5 Environment

Significant changes have been made in the DB2 UDB V5 environment with the introduction of the Database Administration Server and DB2 Profile Registries.

2.5.1 DB2 Administration Server (DAS)

The DB2 Administration Server (DAS) is a special DB2 instance that manages DB2 servers both locally and remotely. The DB2 Administration Server or DAS provides the following functions:

- Enables remote administration of DB2 Servers.
- Provides a scheduler that is used to execute jobs locally or remotely. The scheduled jobs are user-defined and may also include operating system commands.
- Provides a mechanism for DB2 Discovery to return information to remote clients.

The DB2 Administration Server is configured during the installation of the DB2 UDB product. It starts automatically when the system is booted. All remote DB2 administration tasks are sent to the DB2 Administration Server for local execution.

DAS can be used to perform administrative tasks on the remote client for the database server. Some of these tasks may require specific authority for execution. The user on the remote client must part of the SYSADM_GROUP for the instance. For security reasons, you may not want this remote client user to have access to all SYSADM functions. In general, the tasks performed by the DAS are:

- Query the operating system configuration information.
- Query the operating system for user and group information.
- Be able to start/stop DB2 instances.
- Execute scheduled jobs.
- Collect information for DB2 Discovery.

2.5.1.1 Using the Administration Server

When you install and configure the DB2 UDB database product, the Administration Server instance is automatically created. You can also manually create an Administration Server instance and start, stop, list or remove this instance.

On UNIX platforms, to create an Administration Server, you must have root authority to execute the `dasicrt` command. The syntax is as follows:

```
dasicrt ASName
```

where `ASName` is the name of the Administration Server instance. This is a string of up to eight alphanumeric characters. You use the name of the Administration Server to set up the directory structure and access permissions. You can only start the newly-created Administration Server:

- Following a system reboot after installation
- Using a manual start

To start or stop the Administration Server instance, you must login as the Administration Server and issue the following:

```
db2admin start
db2admin stop
```

The Administration Server is automatically started after each system reboot.

To obtain the name of the Administration Server on your system, you must have root authority (UNIX only) to execute the db2iset command at a command prompt. The syntax is as follows:

```
db2iset -g DB2ADMINSERVER
```

To remove the Administration Server, you must login as the Administration Server owner and issue the following command:

```
db2admin stop
```

Next, you must back up the files in the ASHOME/sqllib or ASHOME/sqllib directory, if needed, where ASHOME is the home directory of the Administration Server. Logout as the Administration Server owner. Then login as root (UNIX only) and remove the Administration Server instance using the dasidrop command as follows:

```
dasidrop ASNAME
```

where ASNAME is the name of the instance being removed. The dasidrop command removes the sqllib directory under the home directory of the Administration Server.

2.5.2 DB2 Profile Registries

Registry values, environment variables and configuration parameters control your database environment. Prior to the introduction of the DB2 profile registry in V5, changing your DB2 environment on Windows or OS/2 workstations, required you to change your session's environment variables and reboot. In Version 5, almost all of the environment variables have been moved to the DB2 profile registry. Use the db2set command to update DB2 registry values without rebooting your system. The DB2 registry applies the updated information to the DB2 applications started after the changes are made. (The DB2 profile registry is not supported in Windows 3.1).

DB2 configures its operating parameters by checking for registry values according to the following search order:

1. The session's environment variables.
2. Profile registry values set with the db2set command in the instance level profile. This profile contains instance level settings and overrides values set at the global level.
3. Profile registry values set with the db2set command in the global level profile. This profile contains machine-wide variable settings. Any variable not defined in the node or instance levels will be evaluated at this level.

The objective of the DB2 Profile Registry is to consolidate the DB2 environment variables into an environment registry. The concept of registry refers to the usage of an operating system database or flat-file to contain variable names and values.

Table 3 on page 73 shows the variables that continue to be implemented as environment variables set outside of the Profile Registry:

PLATFORM	LOCATION
OS/2	DB2INSTANCE (optional), DB2PATH, DB2INSTPROF
Win32 (NT/95)	DB2INSTANCE (optional)
UNIX	DB2INSTANCE (optional)

Table 3. Profile Registry Variables

The use of this profile registry will:

- Support multiple environment profiles (one per instance)
- Provide global (machine-wide) default profile
- Support global (machine-wide) default DB2 instance name
- Eliminate the need to reboot the machine when a variable is modified
- Support applications that have no environment settings
- Support for user/application overrides
- Centralize control of environment variables

The profile registry is made up of three distinct parts, detailed as follows:

2.5.3 System Profile Registry

The System Profile Registry contains the listing of the local instance names. The location of this registry varies by platform, as shown in Table 4:

PLATFORM	LOCATION
OS/2	%DB2INSTPROF%\profiles.reg
Win32 (NT/95)	\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\ PROFILES
AIX	/var/db2/v5/profiles.reg
HP, SUN	/var/opt/db2/v5/profiles.reg

Table 4. System Profile Registry Location

2.5.4 Global Profile Registry

The Global Profile Registry contains global (machine-wide) default variables and DB2 system variable settings. The location of this registry varies by platform, as shown in Table 5:

PLATFORM	LOCATION
OS/2	%DB2INSTPROF%\default.env
Win32 (NT/95)	\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\ GLOBAL_PROFILE
AIX	/var/db2/v5/default.env
HP, SUN	/var/opt/db2/v5/default.env

Table 5. Global Profile Registry Location

2.5.5 Instance Profile Registry

The Instance Profile Registry contains instance variable settings per instance. The location of this registry varies by platform, as shown in Table 6:

PLATFORM	LOCATION
OS/2	%DB2INSTPROF%\<instance>\profile.env
Win32 (NT/95)	\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\ PROFILES\<instance>
UNIX	$HOME/sqllib/profile.env

Table 6. Instance Profile Registry Location

The DB2 Instance Profile Registry is not available for Win16 or MacOS client platforms. The existing flat-file implementation is used.

2.5.6 The db2set Command

The db2set command line tool is used to administer the DB2 Profile Registry. This command displays, sets, resets, or removes profile variables.

The format of the command is as follows:

```
db2set
variable = value
-g
-i instance [node number]
-n DAS node [[-u user] [-p password]
-r
-l
-lr
-v
-?
-h
-all
-null
```

The command options are:

-g Modifies the global profile variables.

-i Specifies the instance profile to use instead of the default.

-n Specifies the remote DB2 Admin Server node name.

-u Specifies the user id to use for the admin server attachment.

-p Specifies the password to use for the admin server attachment.

-r Resets the profile registry for the given instance.

-l Lists all instance profiles.

-ir Lists all supported registry variables.

-v Verbose mode.

-? Displays the command help message.

-h The same as -?.

-all Displays all occurrences of the local environment variables as defined in: The environment, denoted by [e], the node level registry, denoted by [i], and the global level registry, denoted by [g].Modifies the global profile variables.

-null Sets the variables value to null at the specified registry level to prevent looking up the value in the next registry level as defined in the variable value search order.

Some parameters will always default to the global level profile. They cannot be set at the instance of node level profiles. Examples of this are DB2SYSTEM and DB2INSTDEF.

The following are examples:

- To set a parameter for the session, type:

  ```
  db2set parameter=value
  ```

- To set a parameter's value for a specific instance, type:

  ```
  db2set parameter=value -i instance-name
  ```

- To set a parameter at the global profile level, type:

  ```
  db2set parameter=value -g
  ```

- To delete a parameter's value at a specified level, you can use the same command syntax to set the parameter, but specify nothing for the parameter value. For example, to delete the parameter's setting at the node level, type:

  ```
  db2set parameter= -i instance-name node-number
  ```

- To explicitly unset a parameter's value at a specified level and prevent evaluating the parameter at the next level, use the -null option. For example, to use the parameter's setting at the instance level, type:

  ```
  db2set parameter= -null -i instance-name
  ```

- To evaluate the current session's parameter's value, type:

  ```
  db2set parameter
  ```

- To evaluate the parameter's value at all levels, type:

  ```
  db2set parameter -all
  ```

- To view a list of all values defined in the profile registry, type:

  ```
  db2set -all
  ```

2.5.7 Setting Environment Variables Hierarchy

Registry values, environment variables and configuration parameters control your database environment, as shown in Figure 48. Prior to the introduction of the DB2 profile registry in Version 5, changing your DB2 environment on Windows or OS/2 workstations required you to change your session's environment variables and reboot. With Version 5, almost all of the environment variables have been moved to the DB2 profile registry. Use the db2set command to update DB2 registry values without rebooting your machine. The DB2 registry applies the updated information to the DB2 applications started after the changes are made.

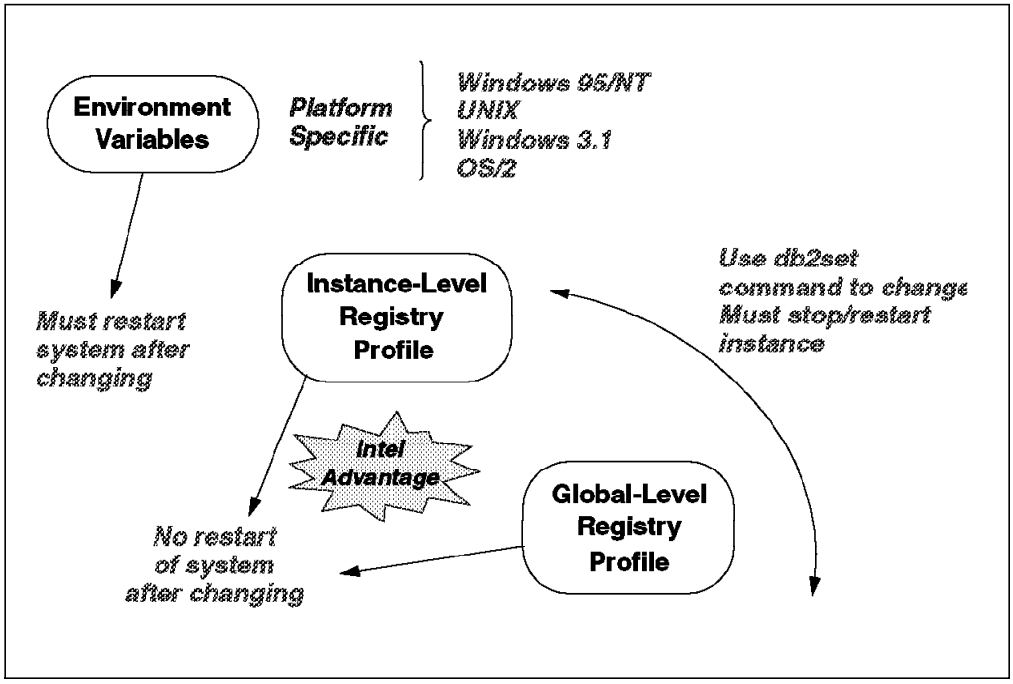


Figure 48. Environment Variables and Profile Registries

We strongly recommend that all DB2 specific registry values be defined in the DB2 profile registry. If DB2 variables are set outside of the registry, remote administration of those variables will not be possible, and the workstation will have to be rebooted in order for the variable values to take effect.

You may want to set the environment variable DB2INSTANCE if you have more than one DB2 instance defined on your system. The DB2 profile registry variable DB2INSTDEF may be set in the global level profile to specify the instance name to use if DB2INSTANCE is not defined.

The method to set DB2 environment variables depends on the platform:

- On Windows 95, set the DB2 environment variable DB2INSTANCE as follows:
  1. Edit the autoexec.bat file.
  2. Reboot to make the change take effect.
- On Windows NT, set the DB2 environment variable DB2INSTANCE as follows:

1. Click on **Start** and select Settings->Control Panel.
2. Double-click on the **System** icon.
3. In the System Properties panel, click on the **Environment** tab and do the following:

– If the variable does not exist:

- Select any system environment variable.
- Change the name in the Variable field to the name of the environment variable you want to set, for example, DB2INSTANCE.
- Change the Value field to the instance name.

– If the variable does exist:

- Select the environment variable you want to change.
- Change the Value field to the instance name.

– Click on the **Set** push button. Select **OK** and reboot the system for the changes to take effect.

- On OS/2, there are three system environment variables that are not stored in the DB2 profile registry. These environment variables, DB2INSTANCE, DB2PATH, and DB2INSTPROF are set when the database is installed. However, because these environment variables are not set in the profile registry, you will need to reboot if you change their settings.

  To change the setting of an environment variable, enter the following at an OS/2 command prompt:

  ```
  set parameter = value
  ```

  To determine the setting of an environment variable, enter:

  ```
  echo %variable-name%
  ```

  To modify system environment variables you must edit CONFIG.SYS, then reboot to make the changes take effect.

- On UNIX:

  The scripts db2profile and db2cshrc are provided as example to help in setting up the database environment. They are located in INSTHOME/sqllib, where INSTHOME is the home directory of the instance owner. The scripts include statements to:

  – Update a user's path with the following directories:

    - INSTHOME/sqllib/bin, INSTHOME/sqllib/adm, INSTHOME/sqllib/misc

  – Set DB2INSTANCE to the default local instance_name for execution.

  An instance owner or SYSADM user may customize the scripts for all users of an instance. Alternatively, users can copy and customize a script, then invoke a script directly or add it to their .profile or .login file.

- On Windows 3.1:

  The DB2 environment on Windows 3.1 is not controlled by profile registries. Instead, Windows 3.1 clients define environment keywords in the file db2.ini (typically found in c:windows directory).

  The db2.ini initialization file is an ASCII file that stores values for the Windows 3.1 client environment keywords. Within the file, there is just one section header titled [DB2 Client Application Enabler]. The parameters are set by specifying a keyword with its associated keyword in the form:

```
KeywordName = keywordValue
```

Note the following:

- All the keywords and their associated values must be located below the section header.
- The keywords are not case sensitive. However, their values can be if the values are character based.
- Comment lines use a semi-colon in the first position of a new line.
- Blank lines are permitted. If duplicate entries for a keyword exist, the first entry is used (no warning is given).

The file is located in the Windows product directory. For example, it would typically be found in *c:windows* directory in a native Windows environment. In Windows, CAE for DB2 V2.1 and V5 must set this information only in the db2.ini file.

Chapter 3. New Features in DB2 UDB V5 for Users of DB2/PE

This chapter covers the features included in DB2 UDB V5 that are new for users of DB2 Parallel Edition V1. These features were introduced to single-partition databases in DB2 Common Server V2.

We will discuss changes in:

- Section 3.1, “Storage Objects” including:
  - Containers and table spaces
- Section 3.2, “DB2 Objects” on page 88 including:
  - Large objects, triggers and user defined functions
- Section 3.3, “Data movement” on page 97 including:
  - Load utility
- Section 3.4, “Backup and Recovery” on page 104 including:
  - Table space backup and recovery
- Section 3.5, “Concurrency” on page 111 including:
  - Isolation levels and locks
- Section 3.6, “Distributed Management” on page 116 including:
  - Distrbuted unit of work and two-phase commit
- Section 3.7, “Security” on page 125 including:
  - Authorites and privileges

3.1 Storage Objects

This chapter discusses those database objects that specifically deal with storage. The objects that we will discuss in detail are nodegroups, table spaces and tables. We will also discuss another storage object called the container. Let’s start by defining some of these objects and their relationship to the database.

3.1.1 Container

A container is a generic term used to describe the allocation of physical space.

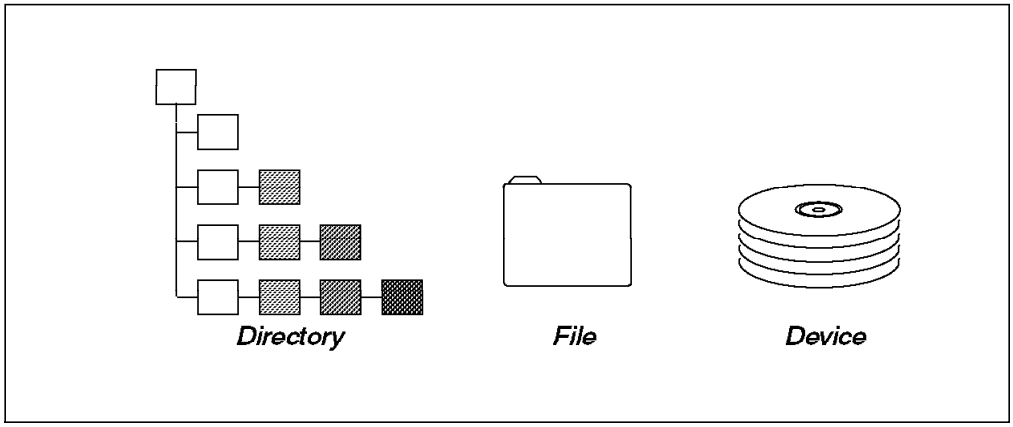


Figure 49. Containers in DB2

Figure 49 shows that a container can be any of the following:

- File
- Directory
- Device

The type of container depends on the type of table space and the platform. For example, in AIX, a device container can be a logical volume.

3.1.2 Table Spaces

Table spaces exist in DB2 to provide you with a logical layer between your data and the storage devices. All DB2 tables reside in table spaces. This means you will be able to control where data is stored. You can use different kinds of table spaces to store different kinds of data. This gives you the ability to create a more detailed physical database designed to fit your particular environment. For example, you can choose slower disks to store less-used data and faster disks to store indexes or frequently accessed data. You can also specify a particular table space for the system catalog tables, user tables or temporary tables.

Recovery can be done at the table space level. Backup and recovery operations can be made for a table space. This will give you more granularity and control since you can back up or restore table spaces individually.

3.1.2.1 Table Spaces and Containers

There is a one-to-many relationship between table spaces and containers. Multiple containers may be defined for a table space. However, a container can only be assigned to one table space. Figure 50 on page 81 shows this one-to-many relationship.

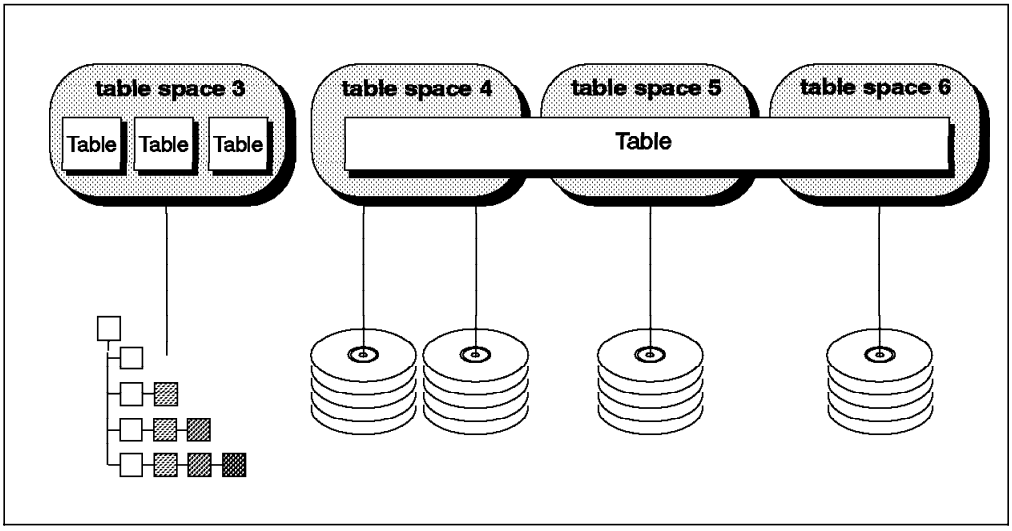


Figure 50. Table Spaces and Containers

Table space 3 has only one container assigned to it, a directory. Table space 4 has two containers assigned to it. The containers for table space 4, table space 5 and table space 6 are devices. A mixture of containers is possible within a database. You may also mix container types within a table space, though it is not recommended for performance reasons. Notice that a table can span multiple table spaces. In Figure 50 one table spans table space 4, table space 5 and table space 6.

3.1.3 Nodegroups

In a multi-partition database, table spaces are defined in nodegroups. A nodegroup is a collection of nodes (or partitions) which exist in the database manager environment. There is a one-to-many relationship between nodegroups and table spaces.

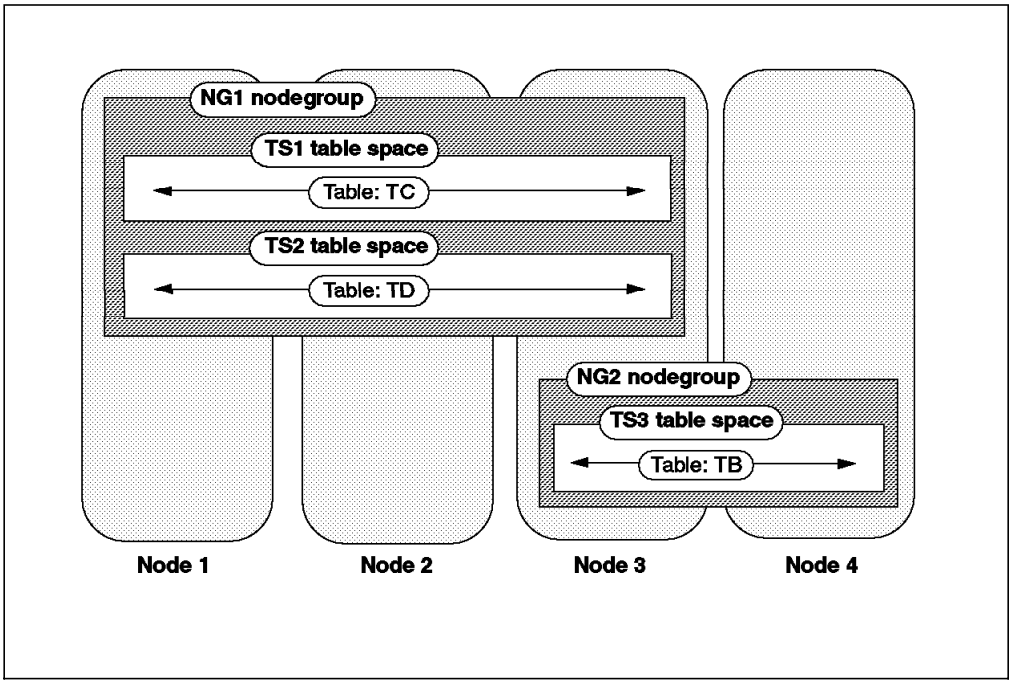


Figure 51. Nodegroups, Table Spaces and Tables

As shown in Figure 51, nodegroup NG1 is defined to include nodes 1, 2 and 3. This means that the table spaces defined in nodegroup NG1, namely TS1 and TS2, exist on these 3 nodes. Any tables defined in the table spaces TS1 or TS2, such as TC and TD, will have partitions on these 3 nodes.

3.1.4 Extents

An extent is an allocation of space within a container of a table space. Database objects are stored in pages within DB2 (except for long varchar and LOBs). These pages are grouped into allocation units called extents. The extent size is defined at the table space level. Once the extent size is established for the table space, it cannot be changed.

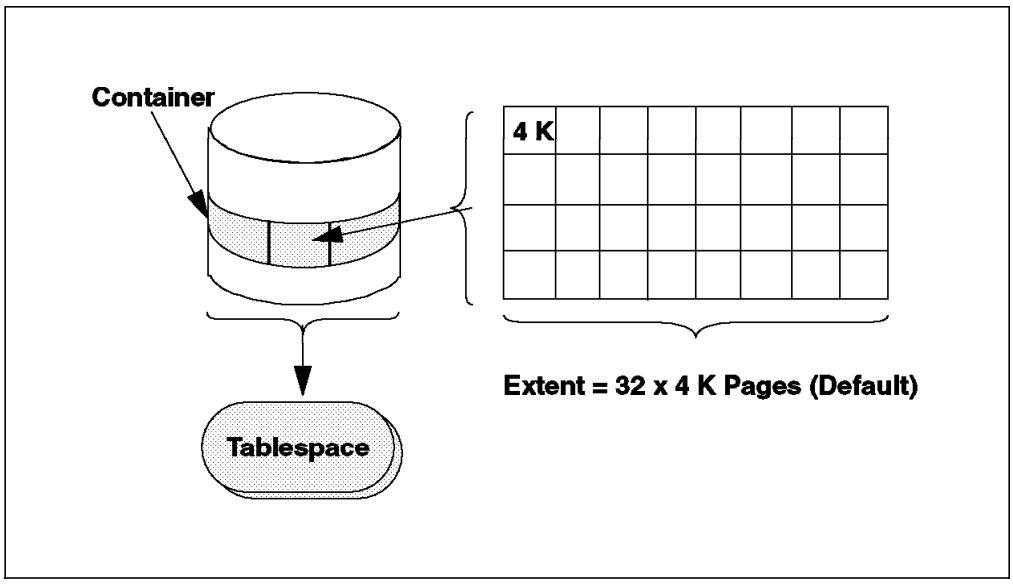


Figure 52. Extents, Containers and Table Spaces

Figure 52shows the relationship between an extent, a container and a table space. The table space is initialized when it is created. As part of this initialization, an allocation size is set for the table space. This parameter is called the extent size. The table space page size is 4096 bytes (4 K). The default extent size for a table space is 32 x 4 K pages.

3.1.5 Types of Table Spaces

DB2 supports two kinds of table spaces:

- System Managed Storage (SMS) table space
- Database Managed Storage (DMS) table space

Both types of table spaces may exist in a database. SMS table spaces are called System Managed because the Database Manager uses the file system mechanism provided by the operating system.

3.1.5.1 SMS Table Spaces

System Managed Storage is based on the storage model where storage is acquired as needed. Containers in SMS tablespaces have the following characteristics:

- The container in an SMS table space does not pre-allocate its storage. A small amount of space is allocated during table space creation.

- Containers cannot be dynamically added to an SMS table space after the table space is created.
- The total number of containers in an SMS table space must be specified when creating the table space.
- SMS containers are represented by directories at the operating system level.

3.1.5.2 DMS Table Spaces

Database Managed Storage (DMS) table spaces are characterized by tablespaces that are built on pre-allocated portions of storage. This storage container can either be a device or a file.

The database manager controls the storage space and allocates space when the container is created. When working with containers and DMS table spaces, the following statements apply:

- If the container is a file, it is created when the table space is created and dropped when the table space is dropped.
- If the container is a device, such as a logical volume in AIX, the logical volume must exist before creating the table space. After dropping the table space, the device still exists and must be removed.
- Storage is pre-allocated to a container when a container is created.
- Containers can be added to a table space after the table space is created.

3.1.5.3 Distributing Tables in DMS Table Spaces

One of the biggest differences and advantages to using DMS table space over SMS table space is the ability to span a table over multiple table spaces. When creating a table in a DMS tables pace, you can decide to place certain objects of the table in different table spaces. DMS table spaces allow you the flexibility to store Large Object Data (Long Field [LF] and Binary Large Objects [BLOBs]) and indexes in different table spaces from the regular table data. The table spaces used to store the table are selected when the table is created.

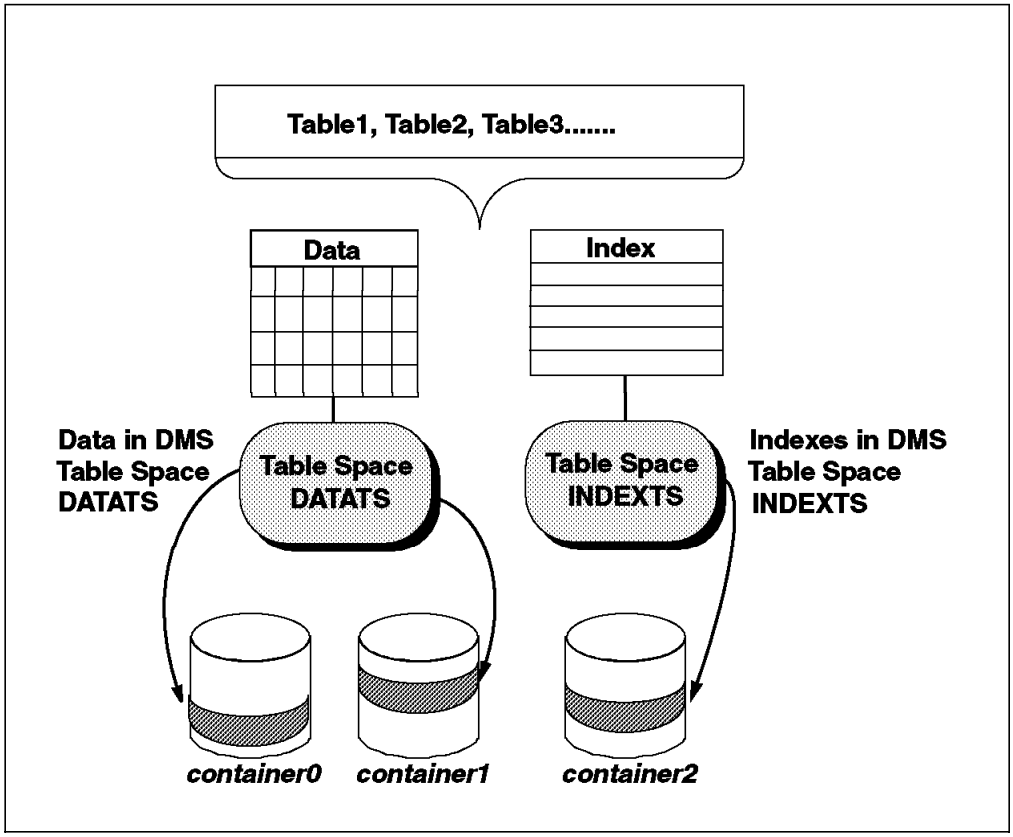


Figure 53. DMS Tablespaces

Figure 53 shows an example of tables being split among two DMS table spaces. The regular table data is placed in table space DATATS. Indexes for these tables are placed in table space INDEXTS. We can see from Figure 53 that Tablespace DATATS is created by using two containers. These containers do not have to be on separate disks. However, for performance reasons, it is better to distribute the containers to different physical devices if you can.

3.1.5.4 AIX Considerations - Device Usage

In AIX, you may want to create a volume group consisting of specialized disk(s) for your database. The containers will be logical volumes. Additional tasks to be performed when using logical volumes as containers in AIX can be summarized as follows:

- If you plan to assign a set of hard drives for the exclusive use of DB2, create a volume group within these disks. By doing so, you will not share these drives with other AIX logical volumes.
- Create as many logical volumes as containers. It is best to place containers on separate physical drives to realize true I/O parallelism. Be careful, since space assigned to the logical volume that is not assigned to the container will be wasted.
- Assign the devices you have created to the instance owner and group. This will ensure that the database manager can read and write to the devices.
- Make sure that all pages in the device are assigned to the container, because they will be unavailable for use by any other application.

3.1.5.5 When to Use DMS Table Spaces

The DMS storage model has important benefits when compared to the SMS storage model. The following is a list of possible advantages:

- You have more control over the placement of database objects according to their type. Tables may be split across multiple DMS table spaces, allowing the separation of table parts.
- You can control the placement of less-frequently accessed items, like BLOBs, that may store images on separate table spaces. These BLOBs, once created, may be data that is neither accessed nor frequently updated. This gives you more flexibility over administrative tasks such as backup and restore operations.
- There may be performance benefits to using DMS table spaces because DB2 has more knowledge of the placement of the data. If using devices for DMS table spaces, you can avoid the overhead of using the operating system's file system.
- DMS table spaces provide you expandability. You can dynamically add containers to the table space online. Rebalancing of the data is done asynchronously when a container is added. Rebalancing is the process of evenly redistributing data among the containers.
- If you know the maximum size of your table space, consider using DMS tablespaces. DMS pre-allocates space as database objects are inserted. The database does not have to compete with other resources that may be used by other applications.

3.1.5.6 Default Table Spaces

When a database is initially created, a set of three table spaces will be created. If you do not specify any table space parameters with the `db2 create database` command, the Database Manager will create these table spaces as SMS tablespaces. The extent size for these table spaces will be set to the default.

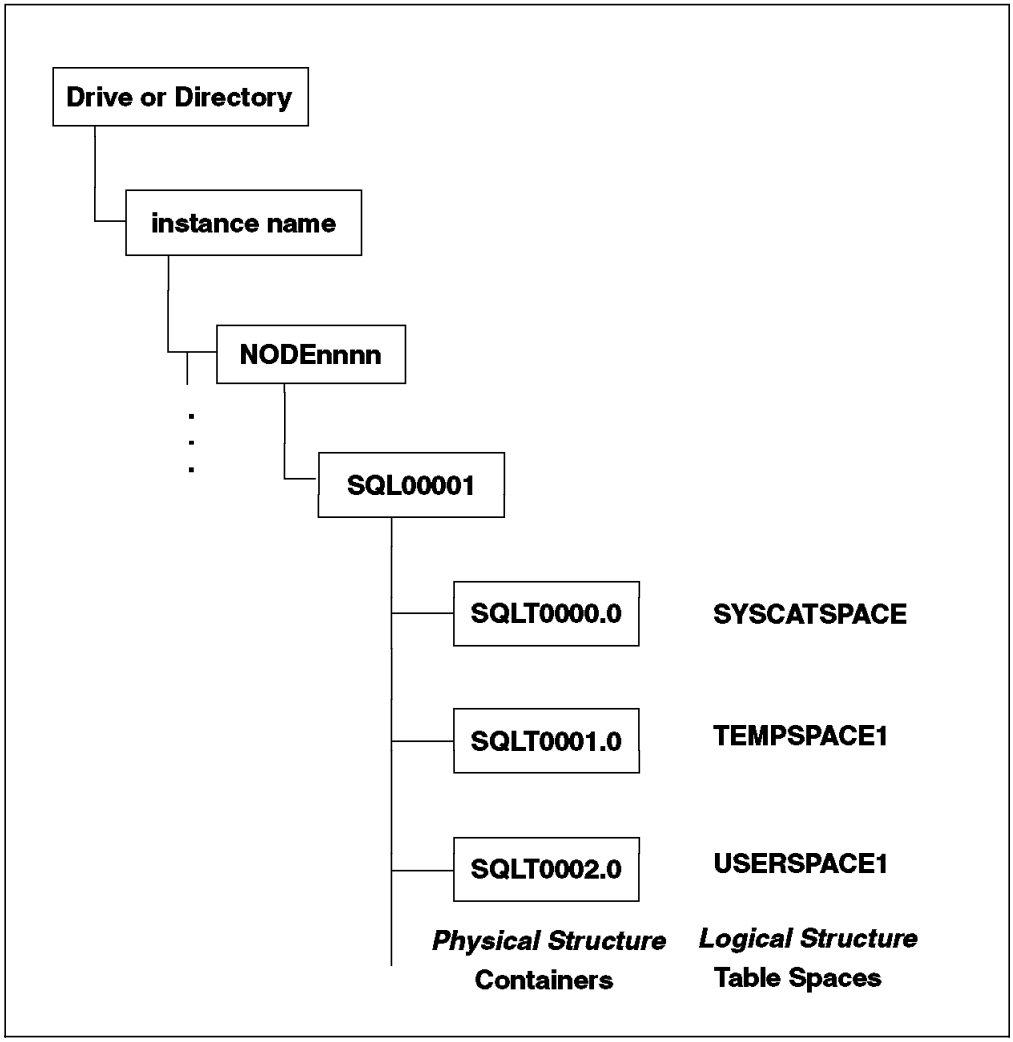


Figure 54. Table Spaces Created with Default Database Configuration

When DB2 creates a database, all data structures, log files, and internal structures for cataloging a database and clients are placed in a directory structure with the naming convention found in Figure 54. The directory which identifies the partition, NODEnnnn is followed by the database (token) directory. The first database directory will be SQL00001. If another database were created, the database directory would be SQL0002, and so on. Within the database directory, a number of objects are created, including three table spaces. Figure 54 shows the three table spaces that are created when default values are used during the db2 create database command. These table spaces are SMS table spaces and as such are directories that will contain the following objects:

- System catalogs (SQLT0000.0)
  This tablespace is called SYSCATSPACE and is defined in the IBMCATGROUP nodegroup. This nodegroup includes only the node (partition) where the database was created, the catalog node. By default, it is an SMS table space. The table space that holds the system catalogs, SYSCATSPACE, cannot be dropped or changed after the database is created. There is only one table space for the system catalogs.
- Temporary space (SQLT0001.0)
  Every DB2 database must have at least one temporary table space assigned to it. This table space, by default, is called TEMPSPACE1 and is defined in

the IBMTEMPGROUP nodegroup. This nodegroup includes all the nodes (partitions) in the instance. There may be multiple table spaces for temporary data. However, only one temporary table space will be used by one SQL statement. This table space may be dropped as long as at least one temporary table space exists.

- User Space (SQLT0002.0)
  By default, DB2 creates a table space for user data called USERSPACE1 defined in the nodegroup IBMDEFAULTGROUP. This nodegroup includes all the nodes (partitions) in the instance. This is the default location for user data. Other table spaces may be created for user data after the database is created. USERSPACE1 may be dropped.

3.1.6 Table Spaces and Containers

This section looks at table spaces and containers. We will focus on DMS table spaces in this section. It is very important to first understand the relationship between containers and table spaces. Let's first look at how the database manager writes data to containers in both SMS and DMS table spaces.

3.1.6.1 Writing to Containers

The database manager will try to evenly distribute a table among containers in the table space. In doing so, the database manager writes a defined number of pages to each container before writing to the next container. If only one container is assigned to the table space, the database manager will write to the same container. The number of pages that are written to a container before writing continues in the next container is called the extent size.

Figure 55 shows three containers, Container 0, Container 1 and Container 2 assigned to table space 4.

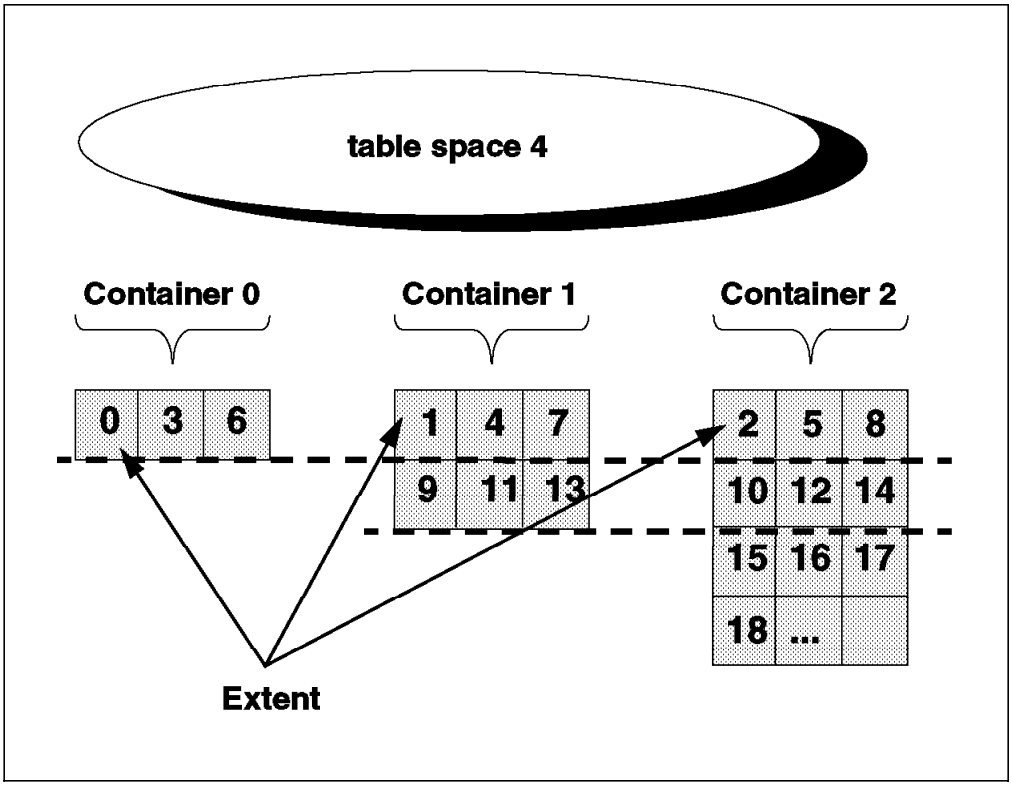


Figure 55. Writing to Containers

Table space 4 has three containers assigned to it, each container having a different size. However, the extent size for table space 4 is the same for each container.

In this example, once the database manager has written an extent to all the containers allocated to table space 4, it will write back to the fijrst one it wrote to. This round-robin process of writing to the containers is designed to balance the data load. In Figure 55 on page 87, the database manager will write to the first extent, Extent 0 on Container 0 in table space 4. When that extent is filled, writing will continue in the second extent, Extent 1 in Container 1 in table space 4. The third extent will be written to Extent 2 in Container 2 in table space 4, and so on.

When Container 0 is full, the database manager will continue to write to the remaining containers, Container 1 and Container 2. Writing will still be in this round-robin manner, alternating between the two containers.

3.2 DB2 Objects

This section will cover the following:

- Large Objects (LOBs)
- DB2 Relational Extenders
- Referential Integrity
- Check Constraints
- Triggers
- User Defined Data Types (UDTs)
- User Defined Functions (UDFs)

3.2.1 Large Objects (LOBs)

Three base large object types are available with DB2 UDB EEE (these were originally introduced in DB2 V2). These are:

- BLOB (binary large object)
- CLOB (character large object)
- DBCLOB (double-byte CLOB)

These types allow multimedia objects such as documents, video, image, and voice to be stored in the database and manipulated like other database objects. Multiple LOB columns, with a maximum size of 2 gigabytes each, can be defined per table. The exact number depends on the size defined for each column. LOB data is directly maintained in the database, and SQL is used for storage and retrieval.

Examples of LOB usage include:

- Banks storing customer’s signatures
- Resumes kept on file with data and a photo
- Voice prints for authentication

3.2.1.1 Locator and File Reference Variables

There are mechanisms provided to allow application programs to manage large object data in efficient ways. One of these mechanisms is locator variables. Locator variables are used to reduce the storage (memory) requirements for the applications and improve the performance by reducing the flow of data between the client (where the application resides) and the server (where the database resides). For example, locator variables can be used to retrieve a single chapter from a 35-chapter book stored as a large object. Once you have retrieved the chapter, you can edit it as required and then replace it. All this is accomplished without the need to retrieve the entire book.

Another mechanism is file reference variables. These are used to retrieve a large object directly to a file or to update a large object in a table directly from a file. File reference variables are used to reduce the storage requirements for the applications since they do not need to hold the large object data in memory.

3.2.2 Reserved Schemas

The following schema names are reserved:

- SYSIBM
  - Base catalogs
  - Built-in types
  - Built-in functions
- SYSCAT
  - Select GRANT to PUBLIC
  - Catalog read-only views
  - User-defined types - SYSFUN schema
- SYSSTAT
  - Updateable catalog views
  - Influences the optimizer
- SYSFUN
  - UDFs provided by IBM

Avoid schema names beginning with SYS. In general, you should avoid schema names that begin with SYS to avoid potential migration problems in the future. The database manager will not allow you to create triggers, user-defined types or user-defined functions using a schema name beginning with SYS.

3.2.3 DB2 Relational Extenders

DB2 Relational Extenders define and implement new complex data types in DB2. The DB2 Relational Extender paradigm is essentially to extend corporate relational data with new data types. Relational Extenders encapsulate the attributes, structure, and behavior of new data types and store them in a column of a DB2 table, such that they can be processed through the SQL language as natural additions to the standard set of DB2 data types. Relational Extenders are separate from the database system itself, in the sense that they can be developed, installed and used independently of full database product releases, but they are also part of the database in that they appear to application developers as seamless extensions to SQL and the DB2 product.

The foundation for the DB2 Relational Extenders is a set of object relational facilities introduced by DB2 V2. These are the fundamental enablers, or building blocks, that make Relational Extenders possible. The object relational facilities are an initial implementation by IBM of selected parts of the emerging SQL3 standard, and comprise user defined types (UDTs), user defined functions (UDFs), large objects (LOBs), and triggers and checks (check constraints). The White Paper ″Object Support in the DB2 family″ discusses these extensions in detail.

The unifying architectural characteristic of DB2 Relational Extenders is that they are based on the DB2 object relational facilities. Generally, the new data type is defined as a UDT, and its behavior is defined as one or more UDFs. It is this basic feature that allows all extender data types to be utilized as extensions to the SQL language. Triggers and checks are applied where necessary to apply constraints or to maintain internal data structure, and their use is generally transparent to the application.

DB2 Relational Extenders help DB2 users extend their business applications with non-standard data types in a controlled manner. The Relational Extender concept is evolutionary. Adding the capability to handle new data types it is not necessary to install a new data base, and concerns about migration or impact on existing systems are minimized. Extenders can be developed in-house, or obtained from IBM, or (over time) from third-party vendors and installed individually when they are required. Development staff need only learn the characteristics of the selected extenders, and add this knowledge to their existing experience in SQL and the DB2 database environment.

3.2.4 Referential Integrity

Referential integrity lets you define required relationships between and within tables. The database manager maintains these relationships which are expressed as referential constraints and require that all values of a given attribute or column of a table also exist in some other table or column. For example, a typical referential constraint might require that every employee in the EMPLOYEE table must be in a department that exists in the DEPARTMENT table. No employee can be in a department that does not exist.

You can build referential constraints into a database to ensure that referential integrity is maintained. When planning for referential integrity, identify the relationships to be established between database tables. You can identify a relationship by defining a primary key and referential constraints.

Constraints are specified with create and alter table statements. The database services enforce constraints on insert, updates, and deletes.

The following definitions are useful for understanding referential integrity.

- A unique key is a set of columns where no two values are duplicated in any other row. You may define one unique key for each table as the primary key. The unique key may also be known as a parent key when referenced by a foreign key.
- A primary key is a unique key that is part of the definition of the table. Each table can only have one primary key. In the preceding tables DEPTNO and EMPNO are the primary keys of the DEPARTMENT and EMPLOYEE tables.
- A foreign key is a column or set of columns in a table that refer to a unique key or primary key of the same or another table. A foreign key is used to

establish a relationship with a unique key or primary key to enforce referential integrity among tables. The column WORKDEPT in the EMPLOYEE table is a foreign key because it refers to the primary key, column DEPTNO, in the DEPARTMENT table.

- A composite key is a key that has more than one column. Unique primary and foreign keys can be composite keys. For example, if departments were uniquely identified by the combination of division number and department number, two columns would be needed to comprise the key to the DEPARTMENT table.
- A parent key is a primary key or unique key of a referential constraint.
- A parent table is a table containing a parent key that is related to at least one foreign key in the same or another table. A table can be a parent in an arbitrary number of relationships. For example, the DEPARTMENT table, which has a primary key of DEPTNO, is a parent of the EMPLOYEE table, which contains the foreign key WORKDEPT.
- A parent row is a row of a parent table whose parent key value matches at least one foreign key value in a dependent table. A row in a parent table is not necessarily a parent row. The fourth row (D11) of the DEPARTMENT table is the parent row of the third and sixth rows in the EMPLOYEE table. The second row (B01) of the DEPARTMENT table is not the parent of any other rows.
- A dependent table is a table containing one or more foreign keys. A dependent table can also be a parent table. A table can be a dependent in an arbitrary number of relationships. For example, the EMPLOYEE table contains the foreign key WORKDEPT, which is dependent on the DEPARTMENT table that has a primary key.
- A dependent row is a row of a dependent table that has a non-null foreign key value that matches a parent key value. The foreign key value represents a reference from the dependent row to the parent row. Since foreign keys may accept null values, a row in a dependent table is not necessarily a dependent row.
- A table is a descendent of a table if it is a dependent table or if it is a descendent of a dependent table. A descendent table contains a foreign key that can be traced back to the parent key of some table.
- A referential cycle is a path that connects a table to itself. When a table is directly connected to itself, it is a self-referencing table. If the EMPLOYEE table has another column called MGRID that contains the EMPNO of each employee′s manager, then the EMPLOYEE table would be a self-referencing table. MGRID would be a foreign key for the EMPLOYEE table.
- A referential constraint is an assertion that non-null values of a designated foreign key are valid only if they also appear as values of a unique key of a designated table. The purpose of referential constraints is to guarantee that database relationships are maintained and data entry rules are followed.
- A self-referencing table is both a parent and a dependent in the same relationship. A self-referencing row is a row that is a parent and a dependent of itself. The constraint that exists in this situation is called a self-referencing constraint. For example, if the value of the foreign key in a row of a self-referencing table matches the value of the unique key in that row, then the row is self-referencing.

Referential integrity removes the burden of constraint checking from application programs.

Enforcement of referential constraints has special implications for some SQL operations that depend on whether the table is a parent or a dependent. This segment describes the effects of referential integrity on the SQL INSERT, DELETE, UPDATE, and DROP operations.

The database manager does not automatically enforce referential constraints across systems. As a result, if you wish to enforce referential constraints across systems, your application programs must contain the necessary logic.

The following referential integrity rules are discussed:

- INSERT Rules
- DELETE Rules
- UPDATE Rules

3.2.4.1 INSERT Rules

You can insert a row at any time into a parent table without any action being taken in the dependent table. However, you cannot insert a row into a dependent table, unless there is a row in the parent table with a parent key value equal to the foreign key value of the row that is being inserted, unless the foreign key value is null. The value of a composite foreign key is null if any component of the value is null.

This rule is implicit when a foreign key is specified.

When you try to insert a row into a table that has referential constraints, the INSERT operation is not allowed if any of the non-null foreign key values are not present in the parent key. If the INSERT operation fails for one row during an attempt to insert more than one row, all rows in the statement are backed out.

3.2.4.2 DELETE Rules

When you delete a row from a parent table, the database manager checks if there are any dependent rows in the dependent table with matching foreign key values. If any dependent rows are found, several actions could be taken. You can determine which action will be taken by specifying a delete rule when you create the dependent table.

The delete rules for a dependent table (the table containing the foreign key) when a primary key is deleted are:

- **RESTRICT**

  Prevents any row in the parent table from being deleted if any dependent rows are found. If you need to remove both parent and dependent rows, delete dependent rows first. Deleting the parent row first would violate the referential constraint and is not allowed.

- **NO ACTION**

  Enforces the presence of a parent row for every child after all the referential constraints are applied.

- **CASCADE**

Implies that deleting a row in the parent table automatically deletes any related rows in the dependent table. This rule is useful when a row in the dependent table has no significance without a row in the parent table.

Deleting the parent row first would automatically delete the dependent rows referencing a primary key. Therefore, the dependent rows would not need to be deleted first. If some of these dependent rows have dependents of their own, the delete rule for those relationships will be applied. In other words, the database manager can handle cascading deletions.

- **SET NULL**

  Ensures that deletion of a row in the parent table sets the values of the foreign key in any dependent rows to null. Other parts of the row are unchanged.

If no delete rule is explicitly defined when the table is created, the NO ACTION rule will be applied.

Any table that can be involved in a delete operation is said to be delete-connected. The following restrictions apply to delete-connected relationships.

- A table cannot be delete-connected to itself in a referential cycle of more than one table.
- When a table is delete-connected to another table through more than one dependent relationship, these relationships must have the same delete rule, either CASCADE or NO ACTION.
- When a self-referencing table is a dependent of another table in a CASCADE relationship, the delete rule of the self-referencing relationship must also be CASCADE.

You can, at any time, delete rows from a dependent table without taking any action on the parent table. For example, in the department-employee relationship, an employee could retire and have his row deleted from the employee table with no effect on the department table. (Ignore, for the moment, the reverse relationship of employee-department, in which the department manager ID is a foreign key referring to the parent key of the employee table. If a manager retires, there is an effect on the department table).

3.2.4.3 UPDATE Rules

The database manager prevents the update of a unique key of a parent row. When you update a foreign key in a dependent table, and the foreign key is not null, it must match some value of the parent key of the parent table of the relationship. If any referential constraint is violated by an UPDATE operation, an error occurs and no rows are updated.

When a value in a column of the parent key is updated:

- If any row in the dependent table matches the original value of the key, the update is rejected when the update rule is RESTRICT.
- If any row in the dependent table does not have a corresponding parent key when the update statement is completed (excluding after triggers), the update is rejected when the update rule is NO ACTION.

To update the value of a parent key that is in a parent row, you must first remove the relationship to any child rows in the dependent tables by either:

- Deleting the child rows, or
- Updating the foreign keys in dependent tables to include another valid key value.

When there is no dependency to the key value in the row, the row is no longer a parent in a referential relationship and can be updated.

If part of a foreign key is being updated and no part of the foreign key value is null, the new value of the foreign key must appear as a unique key value in the parent table. If there is no foreign key dependent on a given unique key, that is, the row containing the unique key is not a parent row, then part of the unique key may be updated. However, no more than one row can be selected for updating in this case, because you are working with a unique key where duplicate rows are not allowed.

3.2.5 Check Constraints

Business rules identified within your design can be enforced through table check constraints. Table check constraints specify search conditions that are enforced for each row of a table. These constraints are automatically activated when an update or insert statement runs against the table. They are defined when using either CREATE TABLE or ALTER TABLE statements.

A table check constraint can be used for validation. For example; the values of a department number must lie within the range 10 to 100; the job title of an employee can only be ′Sales′, ′Manager′, or ′Clerk′; or an employee who has been with the company for more than 8 years must earn more than $40,500.

3.2.6 Triggers

DB2 triggers allow automatic dispatch of specific procedures whenever a relational table is the subject of an update, deletion, or insertion. Multiple triggers are allowed, with user control over their order of execution. Checks implement constraints on values stored in tables. A trigger or a check might be used to implement a business rule on a table, since it will always be invoked whenever the table is changed, without the knowledge or cooperation of the application code. In a Relational Extender context, triggers might be used to maintain the internal structures or indexes of a complex data type.

A trigger is a defined set of actions that are executed when a delete, insert, or update operation is carried out against a specified table. To help support business rules, triggers can be defined. Triggers are stored in the database, therefore application development is faster because you do not have to code the actions in every application program. The trigger is coded once, stored in the database and automatically called by the database manager, as required, when an application uses the database. This ensures that the business rules related to the data are always enforced. If a business rule does change, only a modification to the trigger is required instead of to each application program.

For example, triggers can be used to automatically update summary or audit data.

A user-defined function (UDF) can be called within a triggered SQL statement. This allows the triggered action to perform a non-SQL operation when the trigger is fired. For example, e-mail can be sent as an alert mechanism. Triggers may be dropped at any time.

Adding a trigger to an already populated table does not cause the trigger to be activated.

3.2.7 User Defined Data Types and Functions

3.2.7.1 User-Defined Data Types (UDTs)

A UDT allows a new data type, derived from a standard DB2 data type, to be defined to the database engine. The database engine subsequently treats this UDT in a strongly typed manner, and ensures that it is only used where a data item of this particular type is expected.

There are several benefits associated with UDTs:

1. Extensibility

   By defining new types, you can indefinitely increase the set of types provided by DB2 to support your applications.

2. Flexibility

   You can specify any semantics and behavior for your new type by using user-defined functions (UDFs) to augment the diversity of the types available in the system.

3. Consistent behavior

   Strong typing insures that your UDTs will behave appropriately. It guarantees that only functions defined on your UDT can be applied to instances of the UDT.

4. Encapsulation

   The behavior of your UDTs is restricted by the functions and operators that can be applied on them. This provides flexibility in the implementation since running applications do not depend on the internal representation that you chose for your type.

5. Extensible behavior

   The definition of user-defined functions on types can augment the functionality provided to manipulate your UDT at any time.

6. Performance

   Distinct types are highly integrated into the database manager. Because distinct types are internally represented the same way as built-in data types, they share the same efficient code used to implement built-in functions, comparison operators, indexes, and so forth, for built-in data types.

7. Foundation for object-oriented extensions

   UDTs are the foundation for most object-oriented features. They represent the most important step towards object-oriented extensions.

These commands create the UDTs kph and mph:

```
create distinct type kph as integer with comparisons
create distinct type mph as integer with comparisons
create table speed_limits(route_num smallint, canada_sl kph, us_sl mph)
```

Values using these new types cannot be compared to each other. The following command will fail since no function exists to compare these new data types.

```
select route_num from speed_limits where canada_sl > us_sl
```

3.2.7.2 User-Defined Functions (UDFs)

UDFs allow new functions to be defined, written in C, or sourced from built-in functions, Through the signature of its parameter list, a UDF may be associated with a particular a UDT or standard type. UDFs may be used anywhere in an SQL expression that a standard DB2 built-in function can be used, and thus provide the mechanism for the seamless extension to the SQL language that is a goal of Relational Extenders. UDFs are also required to handle UDT operations.

This is an example of a UDF declaration:

```
create function kph_to_mph(kph) returns mph
language c
extername name
'conversions ! kph_to_mph'
```

3.2.7.3 Sourced UDFs

Note that a sourced UDF, which is different from an external UDF, does not require an implementation in the form of a separate piece of code. Such a UDF uses the same implementation as its source function, along with many of its other attributes.

This is how sourced UDFs can be used to implement addition and average functions for the UDTs mph and kph:

```
create function "=" (mph, mph) returns mph
source "+" (integer, integer)
create function avg_us_speed(mph) returns mph
source avg(integer)
```

3.2.7.4 Fenced UDFs

A UDF can run as a not-fenced or fenced UDF. A not-fenced UDF runs in the same address space as the database manager (the DB2 Agent's address space). A fenced UDF called from the server, runs in an address space (the application's address space) that is isolated from the database manager's address space. A fenced UDF called from a remote machine runs in a special DB2 process whose address space is distinct from the DB2 System Controller. Running your UDF as not-fenced results in increased performance when compared with running it as fenced.

This performance increase is realized regardless of whether the client application runs locally or remotely from the server machine where you are running your UDF application. Note that while performance improvements can be expected when running a not-fenced UDF, there is the possibility that user code could accidentally or maliciously damage the database control structures. In addition, local fenced UDFs are easier for you to debug since the UDFs run in the application's address space. Thus, when debugging your application, the debugger will have access to the UDF code. With not-fenced UDFs, the debugger will also have access to the database manager's address space, thus complicating your debugging activity.

Note: Due to the associated risk of damaging your database, you should only use not-fenced UDFs when you need to maximize the performance benefits. In addition, ensure that all your UDFs are thoroughly tested prior to running them as not-fenced. If a severe error does occur while you are running a not-fenced UDF, the database manager determines whether the error occurred in the UDF code or the database code, and attempts an appropriate recovery.

Keep in mind when you are writing a not-fenced UDF, that it may run in a threaded environment, depending on the operating system. Thus, the UDF must either be completely re-entrant, or manage its static variables so that access to these variables is serialized.

CREATE_NOT_FENCED allows a user to create a user-defined function (UDF) or procedure that is not fenced. UDFs, or procedures that are not fenced, must be extremely well tested because the database manager does not protect its storage or control blocks from these UDFs or procedures. (As a result, a poorly written and tested UDF or procedure that is allowed to run not fenced can cause serious problems for your system.)

3.3 Data movement

This section will cover changes to the Load utility.

3.3.1 Load

The Load utility is intended for the initial load or an append of a table where large amounts of data are moved. There are no restrictions on the data types used by the Load utility including large objects (LOBs) and user-defined types (UDTs). The Load utility speeds up the task of loading large amounts of data into a database. Load is faster than Import because Load writes formatted pages directly into the database while Import does SQL INSERTs. The data being loaded must be local to the server (unlike Import and Export where data can be passed from the client).

In addition to the overview to the Load utility above, here are some details about the Load utility that may be of interest to you. The Load utility also almost completely eliminates the logging associated with the loading of data. In place of logging, you have the option of making a copy of the loaded portion of the table. Load does not fire triggers; nor does it perform referential, and table, constraint checking (other than validating the uniqueness of the unique indexes). Tables with such options defined may be populated faster, or more simply, using Import. If you have a recoverable database, you can do one of the following:

- Use the non-recoverable Load option (and not require a backup).
- Explicitly request a copy of the loaded portion of the table be made.
- Take a backup of the table space(s) in which the table resides after the completion of the load.

The Load utility can take advantage of a hardware configuration where multiple processors and/or multiple storage devices are used such as in a symmetric multiprocessor (SMP) environment. There are several ways in which parallel processing of large amounts of data can take place using the Load utility. One way is through the use of multiple storage devices which allows for I/O parallelism during the load process. Another way involves the use of the multiple processors in an SMP environment which allows for intra-partition

parallelism. Both can be used together to provide even faster loading of the data.

Note: Load will quiesce the tablespace holding the table being loaded. This means all tables in that table space are inaccessible for the duration of the load. Loading a table in DB2/PE did not affect the access to other tables.

These are the steps to take when loading a table into a partitioned database:

- Create tables, indexes
- Create exception tables
- Create input file in sorted format
- Split data
- Backup table space or database
- Consider free space
- For each node:
  - Load ... DUMPFILE ... for exception ... warningcount ... remote file
  - Load query yy.rmt (during execution of Load)
  - Examine xx.msg and dumpfile (after Load completes)
- Examine exception tables (after load completes)
- Backup tablespace if LOGRETAIN=YES and COPY NO
- Set constraints for ...(only if table in check-pending state)
- Update statistics if necessary

3.3.1.1 Load phases

There are multiple phases to the load process: Load, where the data is written into the table; Build, where the indexes are created; and Delete, where the rows that caused a unique key violation are removed from the table. You must run the SET CONSTRAINTS SQL statement after the load completes if there are tables left in the check pending state to validate the table for referential integrity and check constraints. The Load utility generates messages about the progress of each phase. If a failure occurs during the Load process, these messages will assist you in deciding how to recover.

1. **Load phase**

   During the Load phase, data is loaded into the table; index keys and table statistics are collected if necessary. Save points, or points of consistency, are established at intervals specified by you in the SAVECOUNT parameter on the LOAD command. These points of consistency are not established exactly on the number of rows specified with the SAVECOUNT parameter; rather the number of rows are converted to a page count, and rounded up to intervals of the extent size. Messages let you know how many input rows were successfully loaded at the time of the save point. If a failure occurs, you should use the number of input rows at the last successful consistency point with the RESTARTCOUNT parameter during a restart. If the failure occurs near the beginning of the Load process and you were doing a REPLACE, you might consider restarting the load using the REPLACE option.

2. **Build phase**

During the Build phase, indexes are created based on the index keys collected in the Load phase. The index keys are sorted during the Load phase and index statistics are collected. The statistics collected are similar to those collected during RUNSTATS. If a failure occurs, the Build phase restarts from the beginning.

Unique key violations are placed into the exception table, if one was specified, and messages on rejected rows are put into the message file. Following the completion of the load process: review these messages, correct any problems, and insert the corrected rows into the table.

Note: The recording of warnings has a detrimental effect on the performance of the load. If performance is important, and you anticipate a large number of warnings, you should consider using the NOROWWARNINGS filetype modifier. If this filetype modifier was specified, these warnings are suppressed.

3. **Delete phase**

   During the Delete phase, all rows causing a unique key violation are deleted. If a failure occurs, this phase should be restarted by you from the beginning. Information on the rows containing the invalid keys is stored in a temporary file. After you request a restart to begin at the Delete phase, the violating rows are deleted based on the information in a temporary file. You must not modify any data in any temporary files. Also, you must restart the LOAD command with the same parameters, otherwise the Delete phase will fail. If the temporary file has been modified, or does not exist, you should restart the LOAD command at the Build phase. Once the index is re-built, any invalid keys are placed in the exception table if it exists, and duplicate keys are deleted.

3.3.1.2 Exception Table

The exception table is a user-created table which mimics the definition of the table being loaded. It is specified by the FOR EXCEPTION option on the LOAD command. The table is used to store copies of rows that violate unique index rules.

Note: Any rows rejected before the building of the index on the loaded table because of invalid data are not inserted into the exception table.

Rows are added to the existing information in the exception table. The existing information may include rows listing check constraint or foreign key violations; or invalid rows from a previous load. If you want only the invalid rows from this load, you will need to remove the existing rows before invoking load.

The exception table used with the Load utility is identical to the exception table(s) used by the SET CONSTRAINTS statement. An exception table should be created to perform a load which has a unique index and may have duplicate records. If an exception table is not provided for the load, and duplicate records are found, then the load will continue. However, only a warning message is issued about the deleted duplicate records and the deleted duplicate records are not placed anywhere.

After completing the load, information in the exception table can be used any way you wish. You may want to use the information to correct any data that was in error and insert the rows into the original table.

Table 7 on page 100 shows the structure of the exception table message column:

FIELD NUMBER	CONTENTS	SIZE	COMMENTS
1	Number of violations	5 characters	Right justified padded with '0'
2	Type of first violation. Only "I" is used by load	1 character	'I' - Unique Index violation
3	Length of constraint/ index token	5 characters	Right justified padded with '0'
4	Constraint/index token	Length from the previous field	
Note: Only Unique Index violations will be reported by load. The Check Constraint and Foreign Key violations will be reported by running the SET CONSTRAINTS statement with the IMMEDIATE CHECKED FOR EXCEPTION options. Only one unique index violation in a row is reported. LONG or Large Object (LOB) data is not inserted into the exception table. Index token is the "IID" value from SYSCAT.INDEXES that identifies the index.			

Table 7. Exception Table Message Column Structure for Load

The Exception table has the following rules for creation:

- First n columns are the same
- No constraints and no trigger definitions
- n+1 column is of type timestamp
- n+2 column is a CLOB (32 k)
- User must have INSERT privilege

The suggested method for the creation of the Exception table is as follows:

1. Export with select that qualifies No Rows to IXF.
2. Import to Exception table with CREATE option.
3. Drop indexes in exception table.
4. Optionally alter table to add timestamp and CLOB message columns.
5. Exception table uses same PMAP as target table.

Load can also be used with a non-recoverable option. This allows you to perform a non-recoverable load without affecting the recoverability of all other tables in the database. When this type of load is run, there is no requirement for either using the COPY YES option or having a backup taken.

3.3.1.3 Table Space Pending States

Since regular logging is not performed, load uses pending states to preserve consistency of the database. The Load and Build phases of the load process place any associated table spaces into a load pending state. The Delete phase of the load process places any associated table spaces into a delete pending state. If you complete the load process but you do not have either LOGRETAIN or USEREXIT set to on; and, you have not specified the COPY YES option or the NONRECOVERABLE option, then any associated table spaces are placed in a

backup pending state. These states can be checked by using the LIST TABLESPACES command. One last possible state associated with the load process is concerned with referential and check constraints. Dependent tables may be placed in a check pending state following the completion of the load process.

Load pending: If a load fails, the table space(s) involved could be in an inconsistent state because there is no logging. For this reason, the table spaces are left in a load pending state. To remove the load pending state, you will have to restart the load, perform a LOAD REPLACE on the same table on which the load failed, or recover the table space(s) using a RESTORE with the most recent backup (either table space or database backup) and then carry out further recovery actions. (You could also drop the table space and then re-create it).

If a failure occurs while loading data, you can restart the load from the last save point or point of consistency; or reload the entire table by using the REPLACE option.

The remote file specified in the load restart operation should be the one that was specified in the LOAD command that failed.

There are a number of options available should you decide to restart the load.

1. If you decide to restart using the RESTARTCOUNT number option, then you must use the number of rows at the last successful consistency point. To determine that value, use the LOAD QUERY command with either the name specified with the REMOTE FILE option or the default name db2utmp. By choosing RESTARTCOUNT number, the Load restarts from the row following the row identified by the number and attempts to finish the load.

   The load should be restarted at *n+1* where *n* is last SAVECOUNT point.

   Note: Only person doing load can access table if load pending state.

   The RESTARTCOUNT number can only be used with the last successfully completed consistency point. If the last consistency point started but did not complete (that is, SQL3519W is not followed by SQL3520W), then you must carry out the action as described in the help for message SQL3519W.

2. Restartcount B

   If you do not want to continue loading rows, or if the failure was during the Build phase, you can use the RESTARTCOUNT B option. The Load process brings the table to the state of the last save point or point of consistency and then restarts the Build phase. By choosing this option the load restarts, does not attempt to load additional rows, and builds the indexes for the rows already loaded.

3. Restartcount D

   If the message file states that the Build phase completed and all temporary files are unmodified as left by the load, you can use the RESTARTCOUNT D option. The information on the rows containing duplicate keys stored in the temporary files is used to delete those rows.

   The restarted LOAD command should continue until the completion of the load process.

   Note: For minor errors such as nonexistent data files or an invalid dcoldata, the load will clean up and take the table out of the load pending state. You must do the load again in either REPLACE or INSERT mode with correct parameter

4. To return to prior state before load:

- Create a backup copy of existing tablespace before loading data
- Attempt a load with RESTART option
- Load with TERMINATE option to change to recover pending state
- Restore tablespace backup image created before load
- Roll-forward to end of logs

Backup pending: If forward recovery is enabled (LOGRETAIN or USEREXIT is set to ON) and the COPY option was not used, all table spaces in which the loaded table resides are left in a backup pending state. A backup of the database or the table space(s) is required to remove this pending status. The backup is done before any other units of work against the database or table space can be started.

Note: The COPY YES/NO option specifies whether to create a copy of the input data during LOAD or not. If YES is chosen, performance is reduced because all the data being loaded is copied at the same time. This choice is faster than accepting a backup pending state and having to do a backup later before accessing the table. If NO is chosen, and forward recovery in enabled, then the table is placed in a backup pending state.

Check pending state: The loaded table may be in the check pending state if it has table check constraints or referential integrity constraints defined on it. The STATUS flag of the SYSCAT.TABLES in the row corresponding to the loaded table indicates the check pending state of the table. For the loaded table to be usable, the STATUS must have a value of *N* indicating the normal state of the table.

To remove a table from the check pending state, use the SET CONSTRAINT statement. One or more tables may be submitted to be checked in a single invocation. For a dependent table to be checked, the parent table must not be in the check pending state. In the case of a referential integrity cycle, all the tables involved in the cycle must be included in a single invocation of the SET CONSTRAINTS statement.

To manage the loading of several tables, consider the position of each within referential relationships along with the table size and time windows available to carry out the load. It may be convenient, for example, to check the parent table for check constraint violations while the dependent table is loaded. This can only occur if the two tables are not in the same table space.

Exception tables are convenient for a consolidated report of all the rows that have constraints violated. If the exception table option is not used, only the first violation is reported. This may be a cause for frustration when dealing with large tables having more than one constraint violation. The same exception table used for the Load utility may be used for checking constraint violations. As with the Load utility, there is no checking done when running the SET CONSTRAINTS statement to ensure that the exception table is empty. The extra timestamp column in the exception table may be used to distinguish newly-inserted rows from the old ones, if necessary.

The SET CONSTRAINTS statement does not activate any DELETE triggers as a result of deleting rows that violate constraints. It must be noted, however, that

once the table is removed from the check pending state, triggers are active. This implies that, if we correct data and INSERT rows from the exception table into the loaded table, any INSERT trigger defined on the table will be activated. The implications of this on the data should be considered and, if necessary, suitable action should be taken. One option is to drop the INSERT trigger, INSERT rows from the exception table, and then recreate the INSERT trigger.

3.3.1.4 Load Considerations

Performance: The performance of load depends on the nature and size of the data, the number of indexes, the options used, and whether the SET CONSTRAINTS statement is required. Use of SET CONSTRAINTS lengthens the total time needed to load the table and make it usable again.

The Load utility performs almost equally well in either INSERT mode or REPLACE mode.

Index creation will reduce the performance of the load process, especially when data is added to a table already containing data. If there are many indexes on a table which already has a large amount of data and only a small percentage of data to be loaded, you should consider using the IMPORT utility instead of the Load utility. Unique indexes also reduce the performance of the load process if duplicates are encountered. In most cases it is still more efficient to create the index during the load than to complete the load and then use the CREATE INDEX command for each of the indexes.

The Load utility automatically attempts to provide the best performance possible by determining the best values for DISK_PARALLELISM, CPU_PARALLELISM, DATA_BUFFER, and SORT_BUFFER if these parameters have not be specified by the user at the time the utility is run. Optimization of these values is done based on the size and the free space available in the utility heap. Consider allowing the Load utility to choose the values for these parameters and then attempt to tune the parameters for your particular needs.

Performance of the load can be improved by:

- Using the FASTPARSE option in the MODIFIED BY parameter reduces the data checking done on the user-supplied column values, and performance is enhanced. This option should only be used when the data being loaded is known to be valid. This may improve the performance of the load process by about 10 or 20 percent.
- Specifying as many devices for the temporary sort directories as you can so that there is opportunity for parallel I/O during sorting. When very large sort areas are required on systems with restricted file sizes, then multiple directories are required.
- Specifying as large a value for the sort buffer as possible. The sort buffer memory is allocated from the utility heap. Ensure that the utility heap is defined to be large enough.
- Using CPU_PARALLELISM during the load so that you can choose to preserve or not preserve the order of the input data being loaded. The default is to preserve order. Specify ANYORDER on the LOAD command to override this default and gain additional performance improvements.

  Note: The additional performance improvements would be to the create index time, but may hurt in the query performance time if the original data was in clustered index sequence.

- Using the SORT BUFFER and DATA BUFFER parameters. The SORT BUFFER will improve performance only if indexes are created when using load. The DATA BUFFER parameter specifies the total amount of memory allocated to the Load utility as a buffer. The SORT BUFFER parameter specifies the amount of memory allocated for sorting index keys. (This assumes there is real storage to handle the increased buffer size and prevent increased paging.) Both buffer allocations come from the utility heap. You can modify the UTIL_HEAP_SZ database configuration parameter accordingly.
- Specifying multiple directories in different file systems for the USING parameter. Specifying multiple directories improves performance only if indexes are created during the load.
- Controlling the parallelism used during the load in a machine environment where symmetric multi-processor (SMP) exploitation is possible. You can control parallelism in two ways:
  - By using the parameter CPU_PARALLELISM. This parameter controls the degree of parallelism used when parsing, converting, and formatting records.

    Note: When tables include either LOB or LONG VARCHAR data, CPU_PARALLELISM is set to one. Parallelism is not supported in this case.
  - By using the parameter DISK_PARALLELISM. This parameter controls the degree of parallelism used when writing data to the table space containers.
- Using the BINARY NUMERICS and PACKED DECIMAL parameters when loading binary data in ASC data files, to improve the load time involving numeric data.
- Installing high performance sorting libraries from third party vendors to create indexes during the load. Examples of third party sort products include: SMARTsort and SyncSort. This is done through the use of the DB2SORT environment variable.

Other Considerations: When creating indexes during the load, you require at least as much disk space as the sum of the index sizes, and possibly twice as much. The space used is temporary; that is, it is located outside the database in the directories specified for the USING parameter or in the temporary directory defined by the DB2INSTPROF environment variable.

3.4 Backup and Recovery

This section will cover the following topics:

- Table space Backup
- Recovery History file
- Tablespace Recovery: General Considerations
- Roll-Forward Pending
- Disaster Recovery Considerations

3.4.1 Table Space Backup

A table space level backup contains one or more table spaces for a database, specified when BACKUP is executed. A table space level backup can be taken online or offline.

Table space level backup can be used to recover from problems that only affect specific table spaces. While this recovery is taking place, all other table spaces are available for processing.

To ensure that restored table spaces are synchronized with the rest of the database, the table spaces must be rolled forward to the end of the log (or to the point where the table spaces were last used). For this reason, table space level backup and restore can only be performed if roll-forward recovery is enabled. If roll-forward recovery is disabled at any time after a table space level backup is executed, it will not be possible to restore from the backup, and then to roll the table space forward to the current point in time. In this case, all table space level backups taken prior to that time are no longer restorable. The restore operation will fail if the user tries to restore from such a backup. In cases where it cannot be determined that the backup is not valid (if, for instance, the database has been restored and rolled forward, thus creating a new log sequence), the restore may be successful, and the broken restore set will be detected during roll-forward recovery.

The user may choose to separate data, index, long field (LONG), and large objects (LOB) into different table spaces. Long field and LOB data for the same table must reside in the same table space.

Each component of a table may be backed up and restored with the table space in which it resides, independently of the other components of the table.

It is not necessary to back up table spaces for temporary tables. If a list of table spaces to be backed up contains such a table space, BACKUP fails.

Table space level backup and restore cannot be run concurrently.

In a partitioned database system, if you are rolling forward a table space to a point in time, you do not have to supply the list of nodes (database partitions) on which the table space resides. The database manager submits the ROLLFORWARD request to all database partitions. If you are rolling forward a subset of the table spaces to the end of the logs, you must supply the list of nodes. If you want to roll-forward all table spaces to the end of the logs, you do not have to supply the list of nodes. By default, the ROLLFORWARD request is sent to all database partitions.

3.4.2 Recovery History File

A recovery history file (RHF) is created with each database and is automatically updated whenever there is a:

- Database or table space backup
- Database or table space restore
- Database or table space roll-forward
- Table space quiesce
- Load of a table

Figure 56 on page 106 shows creation and updates of the RHF:

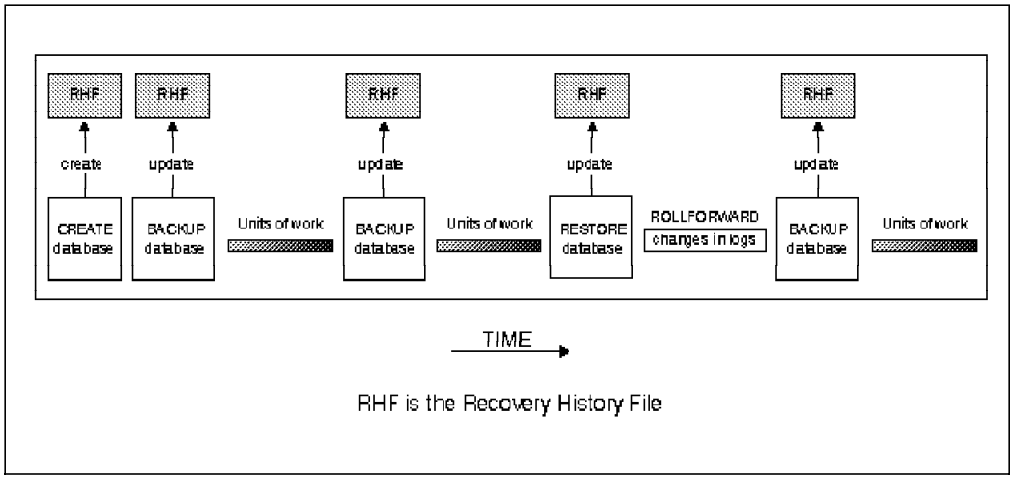


Figure 56. Creating and Updating the Recovery History File

The file contains a summary of the backup information that can be used in the event that all or part of the database must be recovered to a given point in time. The information in the file includes:

- The part of the database that was copied and how.
- The time the copy was made.
- The location of the copy (stating both the device information and the logical way to access the copy).
- The last time a restore was done.

Every backup operation (both table space and full database) includes a copy of the recovery history file. The recovery history file is linked to the database. Dropping a database deletes the recovery history file. Restoring a database to a new location restores the recovery history file. Restoring does not overwrite the existing history recovery file.

If the current database is unusable or not available and the associated recovery history file is damaged or deleted, an option on the RESTORE command allows only the recovery history file to be restored. The recovery history file can then be reviewed to provide information on which backup to use to restore the database.

The size of the file is controlled by the rec_his_retentn configuration parameter that specifies a retention period (in days) for the entries in the file. Even if the number for this parameter is set to zero (0), the most recent full database backup plus its restore set is kept. (The only way to remove this copy is to use the PRUNE with FORCE option). The retention period has a default of 366 days. The period can be set to an indefinite number of days by using -1. In this case, explicit pruning of the file is required.

You can query and run commands against the recovery history file by using an API function call, the command line processor, or the Control Center. The five (5) basic queries and commands are: OPEN, CLOSE, GET NEXT, UPDATE, and PRUNE.

3.4.3 General Considerations for Table Space Recovery

- Roll-forward must be enabled

  Roll-forward recovery is not enabled by the default setting (*Off*) of the LOGRETAIN and USEREXIT configuration parameters. The default for both parameters is set to *Off* because, initially, there is no backup that you can use to recover the database. Initially, the database cannot be recovered, so you cannot perform forward recovery on it.

  To enable a new database for roll-forward recovery, you must enable at least one of these configuration parameters before taking the first backup of the database. When you change the value of one or both parameters, the database will be put into the backup pending state, which requires that you take an offline backup of the database. After the backup operation completes successfully, the database can be used.

- All log records must be applied if only restore table space on subset of nodes

  You can perform a partial or subset restore of a backup created using Version 5 of DB2. This cannot be done with earlier versions of DB2. A partial or subset restore is only possible with a recoverable database. A recoverable database has either the logretain or userexit (or both) configuration parameters enabled to allow roll-forward recovery to occur. With a recoverable database, once the restore is complete, the table space (or table spaces) must be rolled forward (at the end of the restore, each table space that was restored is in the roll-forward pending state).

3.4.4 Roll-Forward Pending

If the database is enabled for forward recovery, you have the option of backing up, restoring, and rolling forward table spaces independent of the database. You may want to implement a recovery strategy for individual table spaces because this can save time: it takes less time to recover a portion of the database than it does to recover the entire database. For example, if a disk is bad and it only contains one table space, the table space can be restored and rolled forward without having to recover the entire database (and without impacting user access to the rest of the database). Also, table-space-level backups allow you to back up critical portions of the database more frequently than other portions, which requires less time than backing up the entire database.

You can issue QUIESCE TABLESPACES FOR TABLE to create a transaction-consistent point in time that you can use for rolling forward table spaces. When you quiesce table spaces for a table (in share, intent to update or exclusive), the request will wait (through locking) for all running transactions that are accessing objects in the table spaces to complete while blocking new requests against the table spaces. When the quiesce request is granted, all outstanding transactions are already completed (committed or rolled back) and the table spaces are in a consistent state. You can look in the recovery history file to find quiesce points and check whether they are past the minimum roll-forward time to determine a desirable time for a ROLLFORWARD STOP.

Different states are associated with a table space to indicate its current status:

- A table space will be placed in the roll-forward pending state after it is restored, or following an I/O error. When the I/O error is corrected, the table space must be rolled forward to remove the roll-forward pending state. If the table space has been restored, it must be rolled forward.

- A table space will be placed in the roll-forward-in-progress state when a roll-forward operation is in progress on that table space. The table space will be removed from the roll-forward-in-progress state when ROLLFORWARD completes successfully.
- A table space will be placed in the restore pending state after a ROLLFORWARD CANCEL or a ROLLFORWARD in which an unrecoverable error occurs on that table space. The table space must be restored and rolled forward again.
- A table space will be placed in the backup pending state after a ROLLFORWARD to a point in time, or after a LOAD NO COPY operation. The table space must be backed up.

If the data and long objects of a table are in separate table spaces, and the table has been reorganized, the table spaces for both the data and long objects must be restored and rolled forward together. You should take a back up of the affected table spaces after the table is reorganized.

After a table space is restored, it is always in the roll-forward pending state (that is, if you restore a table space and specify the WITHOUT ROLLING FORWARD parameter, the WITHOUT ROLLING FORWARD is ignored). To make the table space usable, you must perform roll-forward recovery on it. You have the option of rolling forward to the end of the logs, or rolling forward to a point in time. If you want to roll-forward a table space to a point in time, you should be aware of the following:

- You cannot roll forward system catalog tables to a point in time. These must be rolled forward to the end of the logs to ensure that all table spaces in the database remain consistent.
- A table space that is to be rolled forward to a point in time must have been restored from a backup that is earlier than the point in time specified for the roll-forward.
- If a table is contained in multiple table spaces, all table spaces that contain the table must be rolled forward simultaneously. If, for example, the table data is contained in one table space, and the index for the table is contained in another table space, you must roll forward both table spaces simultaneously to the same point in time.
- Before rolling forward a table space, use the LIST TABLESPACES SHOW DETAIL command. This command returns information on the Minimum Recover Time, which is the earliest point in time to which the table space can be rolled forward. The table space must be rolled forward to at least the Minimum Recover Time so that is synchronized with the information in the system catalog tables.

  The Minimum Recover Time is updated when DDL statements are executed against the table space, or against tables in the table space.

  Because the recovered table space must be consistent with the system catalog tables, you cannot perform a table space roll-forward to recover a dropped table space or table, because the catalog table will indicate that the object was previously dropped. This means that you should not create dummy tables in those table spaces that you want to recover separately from the database.
- If you want to roll forward a table space to a point in time and a table in the table space participates in a referential integrity relationship with another

table that is contained in another table space, you should roll forward both table spaces simultaneously to the same point in time. If you do not, both table spaces will be in the check pending state at the end of the point-in-time roll forward operation. If you roll forward both table spaces at the same time, the constraint will remain active at the end of the point-in-time roll forward operation.

- You should be careful that a point-in-time table space roll-forward operation does not cause a transaction to be rolled back in some table spaces, and committed in others. This can happen when:
  - Point-in-time roll-forward is performed on a subset of the table spaces that were updated by a transaction, and the point in time is before the time that the transaction committed.
  - Any table contained in the table space being rolled forward to a point in time has an associated trigger, or is updated by a trigger that affects table spaces other than the one that is being rolled forward.

  You should find a stop time that will prevent this from happening.
- After a table space point-in-time roll-forward operation completes, the table space (or table spaces) is placed in the backup pending state. You must take a backup of the table space because the log records for the table space that are between the point in time that you rolled forward to and the current time are not applied. The following example shows why the table space backup is required, and how it is used. To make the table space available, you can either back up the entire database, the table space that is in the backup pending state, or a set of table spaces that includes the table space that is in the backup pending state:

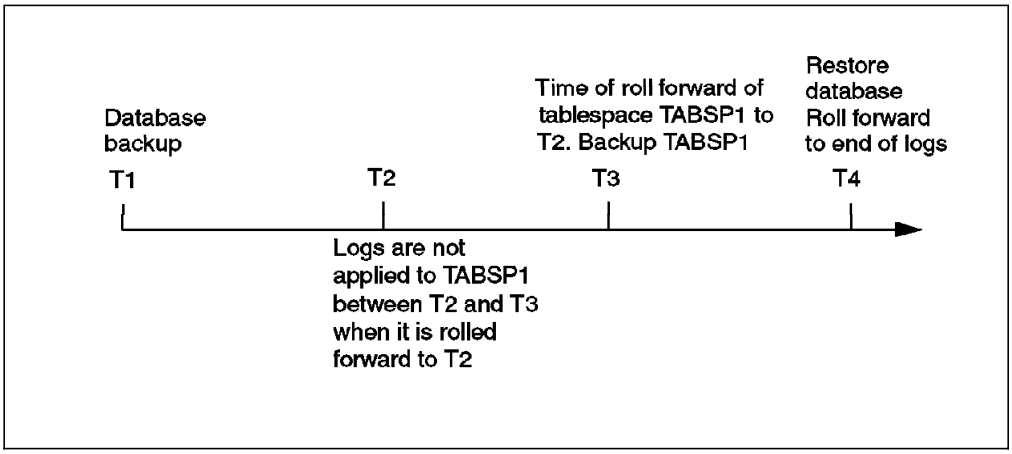


Figure 57. Restore Scenario

In the example shown in Figure 57, you back up the database at time T1. Then, at time T3, you roll forward table space TABSP1 to the point in time T2, then take a back up of the table space after T3. (Because the table space is in the backup pending state, you must take a backup of it. The timestamp of the table space backup is after T3, but the table space is at time T2. Log records are not applied to TABSP1 from between T2 and T3.) At time T4, you restore the database with the backup you took at T1 and roll forward to the end of the logs. The table space TABSP1 will be placed into the restore pending state when time T3 is reached.

The table space is put into the restore pending state at T3 because the database manager assumes that operations were performed on TABSP1

between T3 and T4 without the log changes between T2 and T3 having been applied to the table space. If the log changes between T2 and T3 were reapplied as part of the roll-forward on the database, this assumption would be violated. The required backup of a table space that must be taken after the table space is rolled forward to a point in time allows you to roll that table space forward past the time of the point-in-time roll-forward (T3 in the example).

Assuming that you want to recover table space TABSP1 to T4, you would restore the table space from a backup that was taken after T3 (either the required backup, or a later one) then roll forward TABSP1 to the end of the logs.

In the preceding example, the most efficient way of restoring the database to time T4 would be to perform the required steps in the following order. Because you restore the table space before rolling forward the database, resource is not used to apply log records to the table space when the database is rolled forward, which would happen if you rolled forward the database before you restored the table space.

1. Restore the database.
2. Restore the table space.
3. Roll forward the database.
4. Roll forward the table space.

If you cannot find a back up image of TABSP1 that is after time T3, or you want to restore TABSP1to T3 or before, you can:

- Roll forward the table space to the T3 point in time (in this situation, you do not need to restore the table space again).
- Restore the table space again from the backup of the database that you took at time T1, then roll forward the table space to a time that precedes time T3.
- Drop the table space.

Notes:

1. If, in a partitioned database environment, some database partitions are in the roll-forward pending state, and, on other database partitions, some table spaces are in the roll-forward pending state (but the database partition is not), you must first roll forward the database partitions, then roll forward the table spaces.
2. In a partitioned database environment, you must roll forward all portions of the table space to the same point in time at the same time. This ensures that the table space is consistent at each database partition.

3.4.5 Disaster Recovery Considerations

The term disaster recovery is used to describe the activities that need to be performed to restore the database in the event of a fire, earthquake, vandalism, or another catastrophic event. A plan for disaster recovery can include one or more of the following:

- A site to be used in the event of an emergency
- A different machine on which to recover the database
- Off-site storage of database backups and archived logs

If your plan for disaster recovery is to recover the entire database on another machine, you require at least one full database backup and all the archived logs for the database. When operating your business with this consideration, you may choose to keep a standby database up-to-date by applying the logs to it as they are archived. Or, you may choose to keep the database backup and log archives in the standby site, and perform a restore/roll-forward only after a disaster has occurred. (In this case, a recent database backup is clearly desirable). With a disaster, however, it is generally not possible to recover all of the transactions up to the time of the disaster.

The usefulness of a table space backup for disaster recovery depends on the scope of the failure. When a major disaster occurs, a full database backup is needed on a standby site. If the disaster is a damaged disk, then a table space backup (for each table space using that disk) can be used to recover. If you have lost access to a container because of a disk failure (or for any other reason), you can restore the container to a different location.

With critical business data being stored in your database, you should plan for the possibility of a natural or man-made disaster affecting your database. Both table space backups and full database backups can have a role to play in any disaster recovery plan. The DB2 facilities available for backing up, restoring, and rolling forward data changes provide a foundation for a disaster recovery plan. You should ensure that you have tested the recovery procedures in place to protect your business.

If we compare of database versus table space backup and recovery:

Database recovery:

- Does not require application of all logs
- Reduces backup management

Table space recovery:

- Requires application of all logs if restore only subset of nodes
- Increases the number of backup images

3.5 Concurrency

This section covers the following topics:

- Isolation levels
- Locks

The integrity of the data in a relational database must be maintained as multiple users access and change the data. Concurrency is the sharing of resources by multiple interactive users or application programs at the same time. The database manager controls this access to prevent undesirable effects, such as:

- Lost updates. Two applications, A and B, might both read the same row from the database and both calculate new values for one of its columns based on the data these applications read. If A updates the row with its new value and B then also updates the row, the update performed by A is lost.
- Access to uncommitted data. Application A might update a value in the database, and application B might read that value before it was committed. Then, if the value of A is not later committed, but backed out, the

calculations performed by B are based on uncommitted (and presumably invalid) data.

- Nonrepeatable reads. Some applications involve the following sequence of events: application A reads a row from the database, then goes on to process other SQL requests. In the meantime, application B either modifies or deletes the row and commits the change. Later, if application A attempts to read the original row again, it receives the modified row or discovers that the original row has been deleted.
- Phantom Read Phenomenon. The phantom read phenomenon occurs when:
  1. Your application executes a query that reads a set of rows based on some search criterion.
  2. Another application inserts new data or updates existing data that would satisfy your application's query.
  3. Your application repeats the query from step 1 (within the same unit of work).

  When the query is repeated (step 3), some additional *phantom* rows are returned as part of the result set that were not returned when the query was initially executed (step 1).

3.5.1 Isolation Levels

An isolation level determines how data is locked or isolated from other processes while the data is being accessed. The isolation level will be in effect for the duration of the unit of work. Applications that use a cursor declared using the WITH HOLD clause will keep the chosen isolation level for the duration of the unit of work in which the OPEN CURSOR was performed.

DB2 UDB supports the following isolation levels:

- Repeatable Read (RR)
- Read Stability (RS) - new in UDB for PE users
- Cursor Stability (CS)
- Uncommitted Read (UR)

To discuss the new read stability isolation level, we must make the comparison with repeatable read.

3.5.1.1 Repeatable Read

Repeatable read (RR) locks all the rows an application references within a unit of work. Using repeatable read, a SELECT statement issued by an application twice within the same unit of work, in which the cursor was opened, gives the same result each time. With repeatable read, lost updates, access to uncommitted data, and phantom rows are not possible.

The repeatable read application can retrieve and operate on the rows as many times as needed until the unit of work completes. However, no other applications can update, delete, or insert a row that would affect the result table until the unit of work completes. Repeatable read applications cannot see uncommitted changes of other applications.

With repeatable read, every row that is referenced is locked, not just the rows that are retrieved. Appropriate locking is performed so that another application

cannot insert or update a row that would be added to the list of rows referenced by your query, if the query was re-executed. This prevents phantom rows from occurring. This means that if you scan 10,000 rows and apply predicates to them, locks are held on all 10,000 rows, even though only 10 rows qualify.

Note: The repeatable read isolation level ensures that all returned data remains unchanged until the time the application sees the data, even when temporary tables or row blocking are used.

Since repeatable read may acquire and hold a considerable number of locks, these locks may exceed the number of locks available as a result of the locklist and maxlocks configuration parameters. In order to avoid lock escalation, the optimizer may elect to immediately acquire a single table level lock for an index scan, if it believes that lock escalation is very likely to occur. This functions as though the database manager has issued a LOCK TABLE statement on your behalf. If you do not want a table level lock to be obtained, ensure that enough locks are available to the transaction or use the read stability isolation level.

3.5.1.2 Read Stability

Read stability (RS) locks only those rows that an application retrieves within a unit of work. It ensures that any qualifying row read during a unit of work is not changed by other application processes until the unit of work completes, and that any row changed by another application process is not read until the change is committed by that process. That is, nonrepeatable read behavior is not possible.

Unlike repeatable read, with read stability, if your application issues the same query more than once, you may see additional phantom rows (the phantom read phenomenon). Recalling the example of scanning 10,000 rows, read stability only locks the rows that qualify. Thus, with read stability, only 10 rows are retrieved, and a lock is held only on those ten rows. Contrast this with repeatable read, where in this example, locks would be held on all 10,000 rows. The locks that are held can be share, next share, update, or exclusive locks.

Note: The read stability isolation level ensures that all returned data remains unchanged until the time the application sees the data, even when temporary tables or row blocking are used.

One of the objectives of the read stability isolation level is to provide both a high degree of concurrency as well as a stable view of the data. To assist in achieving this objective, the optimizer ensures that table level locks are not obtained until lock escalation occurs.

The read stability isolation level is best for applications that include all of the following:

- Operate in a concurrent environment.
- Require qualifying rows to remain stable for the duration of the unit of work.
- Do not issue the same query more than once within the unit of work.
- Do not require that the query get the same answer when issued more than once in the same unit of work.

3.5.2 Locks

The database manager provides concurrency control and prevents uncontrolled access by means of locks. A lock is a means of associating a database manager resource with an application to control how other applications can access the same resource. The application with which the resource is associated is said to hold or own the lock.

The database manager imposes locks to prohibit applications from accessing uncommitted data written by other applications (unless the uncommitted read isolation level is used). This principle protects data integrity (that is, the consistency and security of data). Locks can also prohibit the updating of rows (such as for a repeatable read application).

To satisfy data integrity, the database manager acquires locks implicitly, under database manager control. Except for the uncommitted read isolation level, it is never necessary for an application to request a lock explicitly to ensure that uncommitted data is hidden from other processes.

Database manager locks have the following basic attributes:

- Object

  The resource being locked. The only types of explicitly lockable objects are tables. The database manager also imposes locks on other types of resources, such as rows, tables and table spaces (also indexes). The object being locked represents the granularity of the lock.

- Duration

  The length of time a lock is held. Lock durations are affected by isolation levels.

- Mode

  The type of access allowed for the lock owner, as well as the type of access permitted for concurrent users of the locked object. It is sometimes referred to as the state of the lock.

  Modes and their effects are shown in order of increasing control over resources:

  – IN (Intent None)

    The lock owner can read any data in the table, including uncommitted data, but cannot change any of it. No row locks are acquired by the lock owner. Other concurrent applications can read or update the table. Both table spaces and tables can be locked in this mode.

  – IS (Intent Share)

    The lock owner can read data in the locked table, but can not change this data. When an application holds the IS table lock, the application acquires an S or NS lock on each row read. In either case, other applications can read or update the table. Both table spaces and tables can be locked in this mode.

  – NS (Next Key Share) - new in UDB for PE users

    This lock is acquired on rows of a table, instead of a share lock. The lock owner and all concurrent applications can read, but not change, the locked row. Only individual rows can be locked in NS mode. This lock is

acquired in place of a share (S) lock on data that is read with the RS or CS isolation levels.

- S (Share)

  The lock owner and any concurrent applications can read, but not change, the locked data. Individual rows can be Share locked. If a table is share locked, no row locks are acquired by the lock owner. Other concurrent applications can read the table. Both rows and tables can be locked in this mode.

- IX (Intent Exclusive)

  The lock owner and concurrent applications can read and change data in the table. When the owner reads data, it acquires an S, NS, X, or U lock on each row. It also acquires an X lock on each row that it updates. Other concurrent applications can both read and update the table. Both table spaces and tables can be locked in this mode.

- SIX (Share with Intent Exclusive)

  The lock owner can both read and change data in the table. The lock owner acquires X locks on the rows it updates, but does not acquire locks on rows that it reads. Other concurrent applications can read the table. Only a table object can be locked in this mode.

- U (Update)

  The lock owner can update data in the locked object and acquire X locks on the rows prior to updates. Other units of work can read the data, but cannot attempt to update it. Both rows and tables can be locked in this mode. When a table is locked in U mode, X row locks are obtained.

- NX (Next Key Exclusive) - new in UDB to PE users

  This lock is acquired on the next row when a row is deleted from an index or inserted into the index of a table. The lock owner can read but not change the locked row. Only individual rows can be locked in NX mode. This is similar to an X lock except that it is compatible with the NS lock.

- NW (Next Key Weak Exclusive)

  This lock is acquired on the next row when a row is inserted into the index of a non-catalog table. The lock owner can read but can not change the locked row. Only individual rows can be locked in NW mode. This is similar to X and NX locks except that it is compatible with the W and NS locks.

- X (Exclusive)

  The lock owner can both read and change data in the locked object. Tables can be Exclusive locked, meaning that no row locks will be acquired. Only uncommitted read applications can access the locked table. Both rows and tables can be locked in this mode.

- W (Weak Exclusive)

  This lock is acquired on the row when a row is inserted into a non-catalog table. The lock owner can change the locked row. Only individual rows are locked in W mode. This lock is similar to an X lock except that it is compatible with the NW lock. Only uncommitted read applications can access the locked row.

– Z (Superexclusive)

  This lock is acquired on a table in certain conditions, such as when the table is altered or dropped, an index on the table is created or dropped, or a table is reorganized. No other concurrent application can read or update the table. Both table space and table objects can be locked in this mode.

Note that only tables and table spaces will obtain the *intent* lock modes. That is, intent locks are not obtained for rows.

3.6 Distributed Management

This section will cover the following topics:

- Remote Unit of Work (RUOW)
- Distributed Unit of Work (DUOW)

In the DB2 database manager, a transaction is commonly referred to as a unit of work. A unit of work is a recoverable sequence of operations within an application process, and is the basic building block used by the database manager to ensure that a database is in a consistent state. Any reading or writing to the database is done within a unit of work. A point of consistency (or commit point) is a time when all recoverable data that an application accesses is consistent with related data.

A unit of work starts when the first SQL statement is issued against the database. The application must end the unit of work by issuing either a COMMIT or a ROLLBACK statement. The COMMIT statement makes permanent all changes made within a unit of work, whereas the ROLLBACK statement removes these changes from the database. If the application ends normally without either of these statements, the unit of work is automatically committed. If it ends abnormally while in the middle of a unit of work, the unit of work is automatically rolled back. Once issued, a COMMIT or ROLLBACK cannot be stopped. With some multi-threaded applications, if the application ends normally without either of these statements, the unit of work is automatically rolled back. Similarly on some operating systems, if the application ends normally without either of these statements, the unit of work is automatically rolled back. The recommendation when writing your applications is to always explicitly COMMIT or ROLLBACK your completed unit of work.

3.6.1 Remote Unit of Work (RUOW)

The simplest form of database usage is to read and write to only one database within a single transaction (unit of work). This type of database access is called remote unit of work.

Pre-compile your application program to specify a type 1 connection, that is, specify CONNECT(1) on the PREP command, as described in the Embedded SQL Programming Guide manual.

Here is an example of this type of RUOW:

```
connect db1
select/insert/update db1
commit
connect db2
select/insert/update db2
commit
```

3.6.2 Distributed Unit of Work (DUOW)

When using multiple databases in a single transaction, the requirements for setting up and administering your environment are different, depending on the number of databases that are being updated in the transaction.

3.6.2.1 Updating a Single Database

If your data is distributed across multiple databases, you may wish to update one database while reading from one or more other databases. This type of access can be performed within a single unit of work (transaction). This type of database access is called distributed unit of work.

Performance Tip: You should note that unlike the scenario described in ″Updating Multiple Databases,″ updating a single database while reading multiple databases only requires a one-phase commit (SYNCPOINT[ONEPHASE] on PREP command). Using a one-phase commit process requires less overhead than a two-phase commit process. Therefore, performance is better when using SYNCPOINT(ONEPHASE) rather than SYNCPOINT(TWOPHASE) for applications that only update a single database within a unit of work.

Pre-compile your application program, as described in the Embedded SQL Programming Guide to specify:

1. A type 2 connection, that is, specify CONNECT(2) on the PREP command
2. One-phase commit, that is SYNCPOINT(ONEPHASE) on the PREP command.

3.6.2.2 Updating Multiple Databases

If your data is distributed across multiple databases, you may also wish to read and update several databases in a single transaction. This type of database access is called distributed unit of work. This type of environment is more complex than that described in 3.6.2.1, “Updating a Single Database.” These considerations are dealt with after this section.

- Pre-compile your application program, as described in the Embedded SQL Programming Guide to specify:
  1. A type 2 connection, that is, specify CONNECT(2) on the PREP command
  2. Two-phase commit, that is SYNCPOINT(TWOPHASE) on the PREP command.
- Configure the DB2 transaction manager (TM), as described in 3.6.2.7, “DB2 Transaction Manager” on page 122. This section also provides information about how the two-phase commit process works.

Here is an example of this type of DUOW:

```
connect db1
select/insert/update db1
connect db2
select/insert/update db2
commit
```

3.6.2.3 Design considerations for DUOW applications

The additional complexity involved in multiple updates requires an understanding of the necessary actions, and corresponding SQL statements required to perform DUOW transactions.

DESIRED ACTION	SQL STATEMENT NEEDED
establish connection	connect to <dbname>
switch connection to another database (re-establish a dormant connection)	set connection <dbname>
mark connection for release at next successful commit	release <all>|<current>|<dbname>
sever transaction connection	disconnect <all>|<current>|<dbname>

Table 8. DUOW Actions and Corresponding SQL Statements

These actions require consideration of the following:

- SQLRULES
- Type 1 versus type 2 connect
- Syncpoint 1 phase versus 2 phase commit
- DB2 Transaction Manager

3.6.2.4 SQLRULES

This parameter specifies whether type 2 CONNECTs are to be processed according to the DB2 rules or the Standard (STD) rules based on ISO/ANSI SQL92. The syntax for setting SQLRULES is as follows:

```
db2 set client syncpoint twophase connect 2 sqlrules DB2|STD
```

- SQLRULES DB2

  Allow the use of the SQL CONNECT statement to switch the current connection to another established (dormant) connection. For example:

  ```
  connect to db1
  <sql statements>...
  connect to db2
  <sql statements>...
  connect to db1
  ```

  (You can also use *release current*; *set connection db1*).

- SQLRULES STD

Allow the use of the SQL CONNECT statement to establish a new connection only. The SQL SET CONNECTION statement must be used to switch to a dormant connection. For example:

```
connect to db1
<sql statements>...
connect to db2
<sql statements>...
set connection db1
```

(You can also use *release all*; *connect to db1*).

3.6.2.5 Difference between Type 1 and Type 2 Connects

- Type 1

  The CONNECT (Type 1) statement connects an application process to the identified application server according to the rules for remote unit of work.

  Connect reset disconnects the current connection

- Type 2

  The CONNECT (Type 2) statement connects an application process to the identified application server and establishes the rules for =pplication-directed distributed unit of work. This server is then the current server for the process.

  Connect reset immediately connects to default database (indicated by the DB2DBDFT database environment variable), and the previous connection is now dormant.

  Table 9 on page 120 summarizes the difference between Type 1 and Type 2 connections using CONNECT TO.

TYPE 1	TYPE 2
Each unit of work can only establish connection to one application server.	Each unit of work can establish connection to multiple application servers.
The current unit of work must be committed or rolled back before allowing a connection to another application server.	The current unit of work need not be committed or rolled back before connecting to another application server.
The CONNECT statement establishes the current connection. Subsequent SQL requests are forwarded to this connection until changed by another CONNECT.	Same as Type 1 CONNECT if establishing the first connection. If switching to a dormant connection and SQLRULES is set to STD, then the SET CONNECTION statement must be used instead.
Connecting to the current connection is valid and does not change the current connection.	Same as Type 1 CONNECT if the SQLRULES precompiler option is set to DB2. If SQLRULES is set to STD, then the SET CONNECTION statement must be used instead.
Connecting to another application server disconnects the current connection. The new connection becomes the current connection. Only one connection is maintained in a unit of work.	Connecting to another application server puts the current connection into the dormant state. The new connection becomes the current connection. *Multiple connections* can be maintained in a unit of work. If the CONNECT is for an application server on a dormant connection, it becomes the current connection. Connecting to a dormant connection using CONNECT is only allowed if SQLRULES(DB2) was specified. If SQLRULES(STD) was specified, then the SET CONNECTION statement must be used instead.
SET CONNECTION statement is supported for Type 1 connections, but the only valid target is the current connection.	SET CONNECTION statement is supported for Type 2 connections to change the state of a connection from dormant to current.

Table 9. Differences between Type 1 and Type 2 Connections for CONNECT TO

If you use CONNECT...USER...USING to establish the connection, then Type 1 and Type 2 connections differ as shown in Table 10:

TYPE 1	TYPE 2
Connecting with the USER...USING clauses disconnects the current connection and establishes a new connection with the given authorization name and password.	Connecting with the USER/USING clause will only be accepted when there is no current or dormant connection to the same named server.

Table 10. Differences between Type 1 and Type 2 Connections for CONNECT...USER...USING

When using Implicit CONNECT, CONNECT RESET, and Disconnecting, then Type 1 and Type 2 connections differ as shown in Table 11 on page 121:

TYPE 1	TYPE 2
CONNECT RESET can be used to disconnect the current connection.	CONNECT RESET is equivalent to connecting to the default application server explicitly if one has been defined in the system. Connections can be disconnected by the application at a successful COMMIT. Prior to the commit, use the RELEASE statement to mark a connection as release-pending. All such connections will be disconnected at the next COMMIT. An alternative is to use the precompiler options DISCONNECT(EXPLICIT), DISCONNECT(CONDITIONAL), DISCONNECT(AUTOMATIC), or the DISCONNECT statement instead of the RELEASE statement.
After using CONNECT RESET to disconnect the current connection, if the next SQL statement is not a CONNECT statement, then it will perform an implicit connect to the default application server if one has been defined in the system.	CONNECT RESET is equivalent to an explicit connect to the default application server if one has been defined in the system.
It is an error to issue consecutive CONNECT RESETs.	It is an error to issue consecutive CONNECT RESETs ONLY if SQLRULES(STD) was specified because this option disallows the use of CONNECT to existing connection.
CONNECT RESET also implicitly commits the current unit of work.	CONNECT RESET does not commit the current unit of work.
If an existing connection is disconnected by the system for whatever reasons, then subsequent non-CONNECT SQL statements to this database will receive an SQLSTATE of 08003.	If an existing connection is disconnected by the system, COMMIT, ROLLBACK, and SET CONNECTION statements are still permitted.
The unit of work will be implicitly committed when the application process terminates successfully.	Same as Type 1.
All connections (only one) are disconnected when the application process terminates.	All connections (current, dormant, and those marked for release pending) are disconnected when the application process terminates.

Table 11. Differences between Type 1 and Type 2 Connections for Implicit Connect, Connect Reset and Disconnect

Connection failure characteristics differ between Type 1 and Type 2 connections as shown in Table 12 on page 122:

TYPE 1	TYPE 2
Regardless of whether there is a current connection when a CONNECT fails (with an error other than server-name not defined in the local directory), the application process is placed in the unconnected state. Subsequent non-CONNECT statements receive an SQLSTATE of 08003.	If there is a current connection when a CONNECT fails, the current connection is unaffected. If there was no current connection when the CONNECT fails, then the program is then in an unconnected state. Subsequent non-CONNECT statements receive an SQLSTATE of 08003.

Table 12. Differences between Type 1 and Type 2 Connections for Connection Failures

3.6.2.6 Syncpoint 1 versus 2-Phase Commit

ONEPHASE: ONEPHASE specifies that no transaction manager (TM) is to be used to perform a two-phase commit. A one-phase commit is to be used to commit the work done by each database in multiple database transactions.

TWOPHASE: TWOPHASE specifies that the TM is required to coordinate two-phase commit among those databases that support this protocol.

Multi-site update with two-phase commit:

- Maintains data integrity across databases
- Transactions are all or nothing
- Centrally controlled by coordinator
- Two step process
  1. Prepare phase
  2. Commit phase

The components involved in two-phase commit are:

- The application (AP)
- Resource manager (RM)
- Transaction manager (TM, also SPM - syncpoint manager)

NONE: NONE specifies that no TM is to be used to perform a two-phase commit, and does not enforce single updater, multiple reader. A COMMIT is sent to each participating database. The application is responsible for recovery if any of the commits fail.

3.6.2.7 DB2 Transaction Manager

The database manager provides transaction manager functions that can be used to coordinate updating several databases within a single unit of work. The database client automatically coordinates the unit of work and uses a transaction manager database to register each transaction (unit of work) and to track the completion status of that transaction.

The database that will be used as the transaction manager database is determined at the database client by the database manager configuration parameter TM_DATABASE. Consider the following factors when setting this configuration parameter:

- The transaction manager database can be:

  - Another DB2 database.
  - A DB2 for OS/390 Version 5 or later database.

- The transaction manager database can be any database.
- Catalog databases and nodes to allow the following:
  - All database manager instances participating in a distributed transaction must be able to connect to the transaction manager database that was specified by the client's TM_DATABASE configuration parameter. An instance participates in a distributed transaction if the transaction connects to one or more databases contained in that instance. If, for example, the TM_DATABASE parameter is set to DB2TRMGR at the database client, you should be able to issue the following command from each participating instance:

    ```
    CONNECT TO DB2TRMGR
    ```

    The result of this command should connect you to the same database, on the same node from every participating instance, as well as the database client.
  - The database manager instance containing the transaction manager database must be able to connect to all other databases participating in the distributed transaction. If, for example, the client connects to the SAVINGS_DB, CHECKING_DB and FEE_DB, the instance containing the transaction manager database must also be able to connect to those databases using the same names or aliases that the database client uses.

    Note: The transaction manager database must not be cataloged using the alias option to specify an alternate name.
- If the keyword 1ST_CONN is defined for the TM_DATABASE parameter, the first database to which the application connects in the transaction will be used as the transaction manager database. In this case, all databases used in any transaction initiated from the database client must be able to connect to one another using the same database aliases as are used at the database client. This effectively means that each database within a network must have a unique alias across the network.

  Care must be taken when using 1ST_CONN and you should only use this configuration if it is easy to maintain, for example, in the following situations:
  - The database client initiating the transaction is in the same instance that contains the participating databases, including the transaction manager database.
  - You are using DCE directory services to catalog and manage access to your databases.

  Note that if your application attempts to disconnect from the database being used as the transaction manager database, you will receive a warning message and the connection will be held until the unit of work is committed.

3.6.2.8 Resynchronization Process

Recovering from Problems During Two-Phase Commit: Recovering from error situations is a normal task associated with application programming, system administration, database administration and system operation. Distributing databases over several remote servers increases the potential for error situations resulting from network or communication failures. To ensure data integrity, the database manager provides the two-phase commit process. The following explain how the database manager handles errors during this two-phase commit process:

- **First Phase Error**

  If a database responds that it failed to prepare to commit the unit of work, the database client will roll back the unit of work during the second phase of the commit process. A prepare message will not be sent to the transaction manager database in this case.

  During the second commit-phase, the client sends a rollback message to all participating databases that successfully prepared to commit in the first phase. Each database then writes ABORT record to their log file and releases the locks held for this unit of work.

- **Second Phase Error**

  Error handling at this stage is dependent on whether the second phase will commit or roll back the transaction. The second phase will only roll back the transaction if the first phase encountered an error.

  If one of the participating databases fails to commit the unit of work (possibly due to a communications failure), the transaction manager database will retry the commit on the failed database. The database manager configuration parameter RESYNC_INTERVAL (Transaction Resync Interval [RESYNC_INTERVAL]) is used to determine how long the transaction manager database will wait between attempts to commit the unit of work.

  If the transaction manager database fails, it will resynchronize the unit of work when the database is restarted. The resynchronization process will attempt to complete all indoubt transactions, that is, those transactions that have finished the first phase and have not completed the second phase of the commit. The database manager, where the transaction manager database resides, will perform the resynchronization by:

  1. Connecting to the databases that replied that they were PREPARED to commit during the first phase of the commit process.
  2. Attempting to commit the indoubt transaction at that database. (If the indoubt transaction cannot be found, the database manager assumes that the database successfully committed the transaction during the second phase of the commit process.)
  3. Committing the indoubt transaction in the transaction manager database after all indoubt transactions have been committed in the participating databases.

  If one of the participating databases fails and is restarted, the database manager for this database will check the log of the transaction manager database to determine whether the transaction should be rolled back. If the transaction is not found in the log, the database manager assumes the transaction was rolled back and will roll back the indoubt transaction for this database. Otherwise, the database will wait for a commit request from the transaction manager database.

3.6.2.9 Lock Timeout Avoidance

Without lock timeout detection, in an abnormal situation, your application may have to wait for a lock to be released. This might occur, for example, when a transaction is waiting for a lock held by another user's application, and the other user has left their workstation without performing some interaction to allow their application to commit their transaction which would release the lock. Obviously, this results in poorer application performance. To avoid stalling your program in such a case, you can use the LOCKTIMEOUT CONFIGURATION parameter to set the maximum time that any application waits to obtain a lock.

Using this parameter helps avoid global deadlocks, especially in distributed unit of work (DUOW) applications. If the lock times out, that is, if the time that the lock request is pending is greater than the locktimeout value, your application receives an error and your transaction is rolled back. For example, if program1 tries to acquire a lock which is already held by program2, program1 returns SQLCODE -911 (SQLSTATE 40001) with reason code 68 if the timeout is expired.

3.7 Security

This section will cover the following:

- Authorities
- Privileges
- Controlling Access

Authorization is the process in which DB2 obtains information about an authenticated DB2 user that indicates the database operations a user may perform and what data objects may be accessed. With each user request there may be more than one authorization check depending on the objects and operations involved.

Authorization is performed using DB2 facilities. DB2 tables and configuration files are used to record the permissions associated with each authorization name. The authorization name of an authenticated user, and those of groups in which the user is a member, are compared against the recorded permissions. Based on the comparison, DB2 decides whether to allow the user the requested access.

There are two types of permissions recorded by DB2: privileges and authority levels. A privilege defines a single permission for an authorization name, enabling a user to create or access database resources. Privileges are stored in the database catalogs for a given database. Authority levels provide a method of grouping privileges and control over higher level database manager maintenance and utility operations. Database-specific authorities are stored in the database catalogs for each database; system authorities are recorded by group membership and are stored in the database manager configuration file for a given instance.

Groups provide a convenient means of performing authorization for a collection of users without having to grant or revoke privileges for each user individually. Unless otherwise specified, group authorization names can be used anywhere authorization names are used for authorization purposes. In general, group membership is considered for dynamic SQL and non-database object authorizations (such as instance level commands and utilities) and is not

considered for static SQL. Specific cases where group membership does not apply are noted throughout DB2 documentation, where applicable.

3.7.1 Authorities

Two new authorities, SYSCTRL and SYSMAINT are introduced in UDB for PE users as shown in Figure 58:

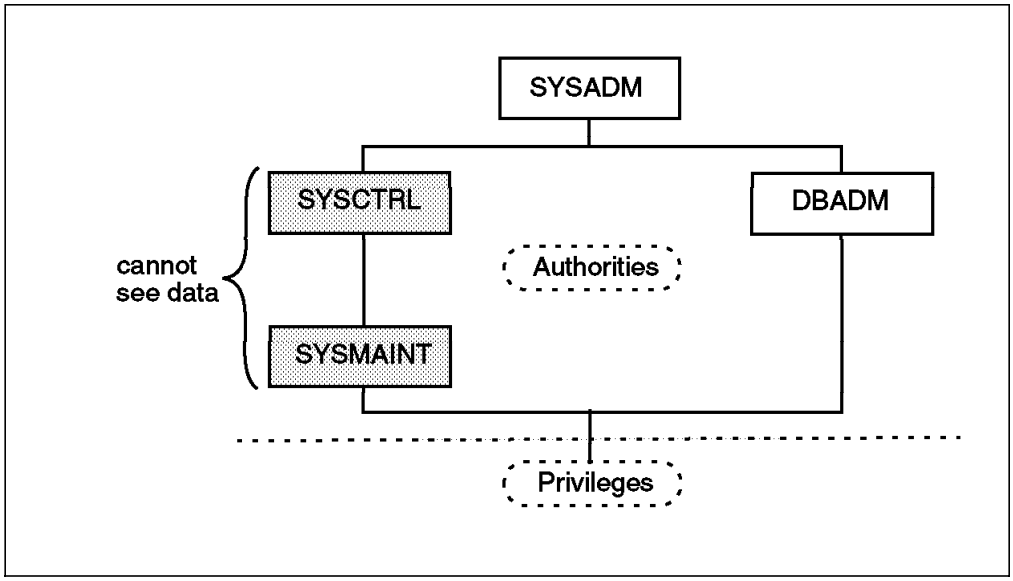


Figure 58. Authorities

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.
- 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. It is also possible to grant a table, view, or schema privilege to another user if that privilege is held WITH GRANT OPTION. However, the WITH GRANT OPTION does not allow the person granting the privilege to revoke the privilege once granted. You must have SYSADM authority, DBADM authority, or CONTROL privilege to revoke the privilege.

A user or group can be authorized for any combination of individual privileges or authorities. When a privilege is associated with a resource, that resource must exist. For example, a user cannot be given the SELECT privilege on a table unless that table has previously been created.

Note: Care must be taken when an authorization name is given authorities and privileges and there is no user created with that authorization name. At some later time, a user can be created with that authorization name and automatically receive all of the authorities and privileges associated with that authorization name.

3.7.1.1 System Administration Authority (SYSADM)

SYSADM authority is the highest level of administrative authority. Users with SYSADM authority can run utilities, issue database and database manager commands, and access the data in any table in any database within the database manager instance. It provides the ability to control all database objects in the instance, including databases, tables, views, indexes, packages, schemas, aliases, data types, functions, procedures, triggers, table spaces, nodegroups, buffer pools, and event monitors.

SYSADM authority is assigned to the group specified by the SYSADM_GROUP configuration parameter. Membership in that group is controlled outside the database manager through the security facility used on your platform.

Only a user with SYSADM authority can perform the following functions:

- Migrate a database
- Change the database manager configuration file (including specifying the groups having SYSCTRL or SYSMAINT authority)
- Grant DBADM authority

In addition, a user with SYSADM authority can perform the functions of users with the following authorities:

- System Control Authority (SYSCTRL)
- System Maintenance Authority (SYSMAINT)
- Database Administration Authority (DBADM)

Note: When users with SYSADM authority create databases, they are automatically granted explicit DBADM authority on the database. If the database creator is removed from the SYSADM group, and if you want to also prevent them from accessing that database as a DBADM, you must explicitly revoke this DBADM authority.

3.7.1.2 System Control Authority (SYSCTRL)

SYSCTRL authority is the highest level of system control authority. This authority provides the ability to perform maintenance and utility operations against the database manager instance and its databases. These operations can affect system resources, but they do not allow direct access to data in the databases. System control authority is designed for users administering a database manager instance containing sensitive data.

SYSCTRL authority is assigned to the group specified by the SYSCTRL_GROUP configuration parameter. If a group is specified, membership in that group is controlled outside the database manager through the security facility used on your platform.

Only a user with SYSCTRL authority or higher can do the following:

- Update a database, node, or distributed connection services (DCS) directory

- Force users off the system
- Create or drop a database
- Drop, create, or alter a table space
- Restore to new database

In addition, a user with SYSCTRL authority can perform the functions of users with System Maintenance Authority (SYSMAINT) authority.

Users with SYSCTRL authority also have the implicit privilege to connect to a database.

Note: When users with SYSCTRL authority create databases, they are automatically granted explicit DBADM authority on the database. If the database creator is removed from the SYSCTRL group, and if you want to also prevent them from accessing that database as a DBADM, you must explicitly revoke this DBADM authority.

3.7.1.3 System Maintenance Authority (SYSMAINT)

SYSMAINT authority is the second level of system control authority. This authority provides the ability to perform maintenance and utility operations against the database manager instance and its databases. These operations can affect system resources, but they do not allow direct access to data in the databases. System maintenance authority is designed for users maintaining databases within a database manager instance that contains sensitive data.

SYSMAINT authority is assigned to the group specified by the SYSMAINT_GROUP configuration parameter. If a group is specified, membership in that group is controlled outside the database manager through the security facility used on your platform.

Only a user with SYSMAINT or higher system authority can do the following:

- Update database configuration files.
- Backup a database or table space.
- Restore to an existing database.
- Perform roll-forward recovery.
- Start or stop a database instance.
- Restore a table space.
- Run trace.
- Take database system monitor snapshots of a database manager instance or its databases.

A user with SYSMAINT, DBADM, or higher authority can do the following:

- Query the state of a table space.
- Update log history files.
- Quiesce a table space.
- Reorganize a table.
- Collect catalog statistics using the RUNSTATS utility.

Users with SYSMAINT authority also have the implicit privilege to connect to a database.

3.7.1.4 Database Administration Authority (DBADM)

DBADM authority is the second highest level of administrative authority. It applies only to a specific database, and allows the user to run certain utilities, issue database commands, and access the data in any table in the database. When DBADM authority is granted, BINDADD, CONNECT, CREATETAB, CREATE_NOT_FENCED, and IMPLICIT_SCHEMA privileges are granted as well. Only a user with SYSADM authority can grant or revoke DBADM authority. Users with DBADM authority can grant privileges on the database to others and can revoke any privilege from any user regardless of who granted it.

Only a user with DBADM or higher authority can do the following:

- Read log files.
- Create, activate and drop event monitors.
- Run the Load utility.

A user with DBADM, SYSMAINT, or higher authority can do the following:

- Query the state of a table space.
- Update log history files.
- Quiesce a table space.
- Reorganize a table.
- Collect catalog statistics using the RUNSTATS utility.

Note: A DBADM can only perform the above functions on the database for which DBADM authority is held.

Table 13 on page 130 summarizes which functions the different authorities have permission to execute.

FUNCTION	SYSADM	SYSCTRL	SYSMAINT	DBADM
migrate database	Y			
update dbm cfg	Y			
grant/revoke DBADM	Y			
specify SYSCTRL group	Y			
specify SYSMAINT group	Y			
add/drop node verify	Y			
catalog/uncatalog db dir	Y	Y		
catalog/uncatalog node dir	Y	Y		
catalog/uncatalog DCS dir	Y	Y		
force users	Y	Y		
create/drop db	Y	Y		
create/drop/alter tablespace	Y	Y		
restore to new database	Y	Y		
update db cfg	Y	Y	Y	
backup database or tablespace	Y	Y	Y	
restore to existing database	Y	Y	Y	
perform roll-forward recovery	Y	Y	Y	
start/stop database instance	Y	Y	Y	
restore tablespace	Y	Y	Y	
run trace	Y	Y	Y	
take dbm or db snapshots	Y	Y	Y	
activate/deactivate database	Y	Y	Y	Y
query tablespace state	Y	Y	Y	Y
update log history files	Y	Y	Y	Y
quiesce tablespace	Y	Y	Y	Y
reorg table	Y	Y	Y	Y
run runstats utility	Y	Y	Y	Y
load tables	Y			Y
read log files	Y	Y		
create/activate/drop event monitors	Y	Y		

Table 13. Database Authorities

3.7.2 Database Privileges

Database privileges involve actions on a database as a whole:

- CONNECT allows a user to access the database
- BINDADD allows a user to create new packages in the database
- CREATETAB allows a user to create new tables in the database
- CREATE_NOT_FENCED allows a user to create a user-defined function (UDF) or procedure that is *not fenced*. UDFs or procedures that are *not fenced*

must be extremely well tested because the database manager does not protect its storage or control blocks from these UDFs or procedures. (As a result, a poorly written and tested UDF or procedure that is allowed to run *not fenced* can cause serious problems for your system).

- IMPLICIT_SCHEMA allows any user to create a schema implicitly by creating an object using a CREATE statement with a schema name that does not already exist. SYSIBM becomes the owner of the implicitly created schema and PUBLIC is given the privilege to create objects in this schema.

Only users with SYSADM or DBADM authority can grant and revoke these privileges to and from other users.

Note: When a database is created, the following privileges are automatically granted to PUBLIC:

- CREATETAB
- BINDADD
- CONNECT
- IMPLICIT_SCHEMA
- SELECT privilege on the system catalog views.

To remove any privilege, a DBADM or SYSADM must explicitly revoke the privilege from PUBLIC.

The hierarchy of authorities and privileges is shown in Figure 59.

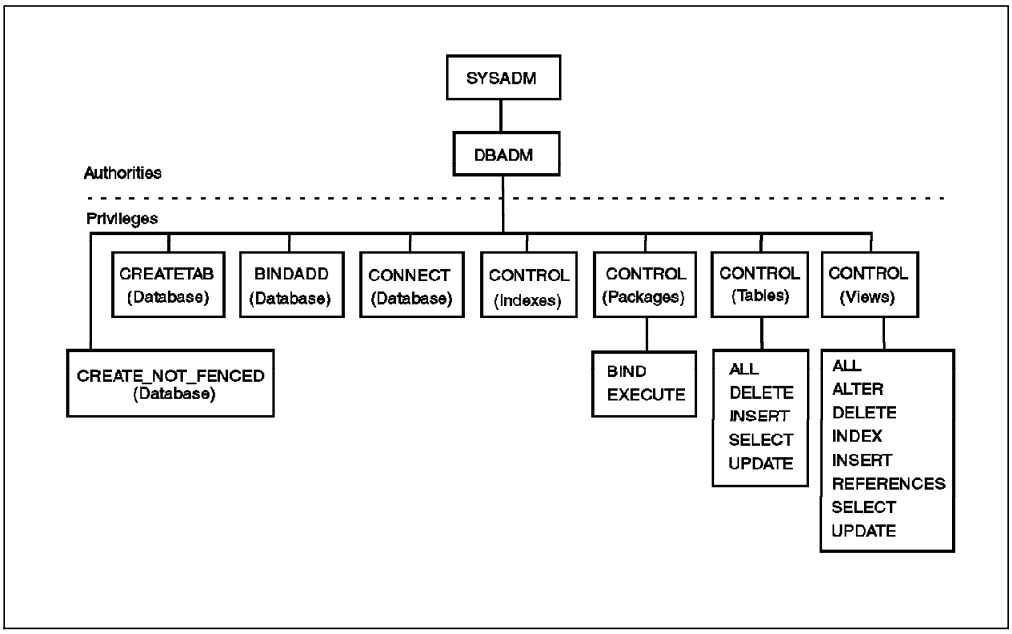


Figure 59. Authorities and Privileges

3.7.2.1 Resources: Privileges required

Table 14 summarizes what privileges are required for each resource:

RESOURCE	NEEDED TO CREATE	NEEDED TO CONTROL	OTHER PRIVILEGES
database	SYSADM SYSCTRL	DBADM	CONNECT BINDADD CREATETAB NOFENCE
package	BINDADD	CONTROL	BIND EXECUTE
table(T) view(V)	CREATETAB (T) CONTROL OR SELECT (V)	CONTROL	SELECT (T/V) INSERT (T/V) DELETE (T/V) UPDATE (T/V) ALTER (T) INDEX (T) REFERENCES (T)
index	INDEX	CONTROL	none
alias	If schema differs from current authid, requires DBADM	CONTROL	none

Table 14. Resources and Required Privileges

3.7.2.2 Table and View Privileges

Table and view privileges involve actions on tables or views in a database. A user must have CONNECT privilege on the database to use any of the following privileges:

- CONTROL provides the user with all privileges for a table or view including the ability to drop it, and to grant and revoke individual table privileges. You must have SYSADM or DBADM authority to grant CONTROL. The creator of a table automatically receives CONTROL privilege on the table. The creator of a view automatically receives CONTROL privilege only if they have CONTROL privilege on all tables and views referenced in the view definition, or they have SYSADM or DBADM authority.
- ALTER allows the user to add columns to a table, to add or change comments on a table and its columns, to add a primary key or unique constraint and to create or drop a table check constraint. The user can also create triggers on the table, although additional authority on all the objects referenced in the trigger (including SELECT on the table if the trigger references any of the columns of the table) is required. A user with ALTER privilege on all the descendent tables can drop a primary key; a user with ALTER privilege on the table and REFERENCES privilege on the parent table, or REFERENCES privilege on the appropriate columns, can create or drop a foreign key. A user with ALTER privilege can also COMMENT ON a table.
- DELETE allows the user to delete rows from a table or view.
- INDEX allows the user to create an index on a table. Creators of indexes automatically have CONTROL privilege on the index.

- INSERT allows the user to insert an entry into a table or view, and to run the IMPORT utility.
- REFERENCES allows the user to create and drop a foreign key, specifying the table as the parent in a relationship. The user may have this privilege only on specific columns.
- SELECT allows the user to retrieve rows from a table or view, to create a view on a table, and to run the EXPORT utility.
- UPDATE allows the user to change an entry in a table, a view, or for one or more specific columns in a table or view. The user may have this privilege only on specific columns.

The privilege to grant these privileges to others may also be granted using the WITH GRANT OPTION on the GRANT statement.

Note: When a user or group is granted CONTROL privilege on a table, all other privileges on that table are automatically granted WITH GRANT OPTION. If you subsequently revoke the CONTROL privilege on the table from a user, that user will still retain the other privileges that were automatically granted. To revoke all the privileges that are granted with the CONTROL privilege, you must either explicitly revoke each individual privilege or specify the ALL keyword on the REVOKE statement, for example:

```
REVOKE ALL ON EMPLOYEE FROM USER HERON
```

3.7.2.3 Package Privileges

A package is a database object that contains the information needed by the database manager to access data in the most efficient way for a particular application program. Package privileges enable a user to create and manipulate packages. The user must have CONNECT privilege on the database to use any of the following privileges:

- CONTROL provides the user with the ability to rebind, drop, or execute a package as well as the ability to extend those privileges to others. The creator of a package automatically receives this privilege. A user with CONTROL privilege is granted the BIND and EXECUTE privileges, and can grant BIND and EXECUTE privileges to other users as well. To grant CONTROL privilege, the user must have SYSADM or DBADM authority.
- BIND allows the user to rebind an existing package.
- EXECUTE allows the user to execute a package.

In addition to these package privileges, the BINDADD database privilege allows users to create new packages or rebind an existing package in the database.

Table 15 on page 134 summarizes the privileges required to perform actions against packages:

ACTION	PRIVILEGES REQUIRED
precompile to bindfile	CONNECT on database
create a new package	CONNECT on database BINDADD on database privileges need to execute each static SQL stmt
modify an existing package	CONNECT on database BIND on package privileges need to execute each static SQL stmt
recreate an existing package	CONNECT on database BIND on package
execute a package	CONNECT on database EXECUTE on package
drop a package	CONNECT on database CONTROL on package or creator of packages

Table 15. Package Privileges

3.7.2.4 Index Privileges

The creator of an index automatically receives CONTROL privilege on the index. CONTROL privilege on an index is really the ability to drop the index. To grant CONTROL privilege on an index, a user must have SYSADM or DBADM authority.

The table-level INDEX privilege allows a user to create an index on that table.

3.7.3 Controlling Access to Database Objects

Controlling data access requires an understanding of direct and indirect privileges, administrative authorities, and packages. This section explains these topics and provides some examples.

Directly granted privileges are stored in the system catalog tables.

Authorization is controlled in three ways:

- Explicit authorization is controlled through privileges controlled with the GRANT and REVOKE statements.
- Implicit authorization is controlled by creating and dropping objects.
- Indirect privileges are associated with packages.

3.7.3.1 Granting Privileges

The GRANT statement allows an authorized user to grant privileges. A privilege can be granted to one or more authorization names in one statement; or to PUBLIC, which makes the privileges available to all users. Note that an authorization name can be either an individual user or a group.

On operating systems where users and groups exist with the same name, you should specify whether you are granting the privilege to the user or group. Both the GRANT and REVOKE statements support the keywords USER and GROUP. If these optional keywords are not used, the database manager checks the operating system security facility to determine whether the authorization name

identifies a user or a group. If the authorization name could be both a user and a group, an error is returned.

The following example grants SELECT privileges on the EMPLOYEE table to the user HERON:

```
GRANT SELECT ON EMPLOYEE TO USER HERON
```

The following example grants SELECT privileges on the EMPLOYEE table to the group HERON:

```
GRANT SELECT ON EMPLOYEE TO GROUP HERON
```

To grant privileges on most database objects, the user must have SYSADM authority, DBADM authority, or CONTROL privilege on that object; or, the user must hold the privilege WITH GRANT OPTION. Privileges can be granted only on existing objects. To grant CONTROL privilege to someone else, the user must have SYSADM or DBADM authority. To grant DBADM authority, the user must have SYSADM authority.

3.7.3.2 Revoking Privileges

The REVOKE statement allows authorized users to revoke privileges previously granted to other users. To revoke privileges on database objects, you must have DBADM authority, SYSADM authority, or CONTROL privilege on that object. Note that holding a privilege WITH GRANT OPTION is not sufficient to revoke that privilege. To revoke CONTROL privilege from another user, you must have SYSADM or DBADM authority. To revoke DBADM authority, you must have SYSADM authority. Privileges can only be revoked on existing objects.

Note: A user without DBADM authority or CONTROL privilege on a table or view is not able to revoke a privilege that they granted through their use of the WITH GRANT OPTION. Also, there is no cascade on the revoke to those who have received privileges granted by the person being revoked.

If a privilege has been granted to both a user and a group with the same name, you must specify the GROUP or USER keyword when revoking the privilege. The following example revokes the SELECT privilege on the EMPLOYEE table from the user HERON:

```
REVOKE SELECT ON EMPLOYEE FROM USER HERON
```

The following example revokes the SELECT privilege on the EMPLOYEE table from the group HERON:

```
REVOKE SELECT ON EMPLOYEE FROM GROUP HERON
```

Note that revoking a privilege from a group may not revoke it from all members of that group. If an individual name has been directly granted a privilege, it will keep it until that privilege is directly revoked.

If a table privilege is revoked from a user, privileges are also revoked on any view created by that user which depends on the revoked table privilege. However, only the privileges implicitly granted by the system are revoked. If a privilege on the view was granted directly by another user, the privilege is still held.

If an explicitly-granted table (or view) privilege is revoked from a user with DBADM authority, privileges will not be revoked from other views defined on that

table. This is because the view privileges are available through the DBADM authority and are not dependent on explicit privileges on the underlying tables.

If you have defined a view based on one or more underlying tables or views and you lose the SELECT privilege to one or more of those tables or views, then the view cannot be used.

Note: When CONTROL privilege is revoked from a user on a table or a view, the user continues to have the ability to grant privileges to others. When given CONTROL privilege, the user also receives all other privileges WITH GRANT OPTION. Once CONTROL is revoked, all of the other privileges remain WITH GRANT OPTION until they are explicitly revoked.

All packages that are dependent on revoked privileges are marked invalid, but can be validated if rebound by a user with appropriate authority. Packages can also be rebuilt if the privileges are subsequently granted again to the binder of the application; running the application will trigger a successful implicit rebind. If privileges are revoked from PUBLIC, all packages bound by users having only been able to bind based on PUBLIC privileges are invalidated. If DBADM authority is revoked from a user, all packages bound by that user are invalidated including those associated with database utilities. Attempting to use a package that has been marked invalid causes the system to attempt to rebind the package. If this rebind attempt fails, an error occurs (SQLCODE -727). In this case, the packages must be explicitly rebound by a user with:

- Authority to rebind the packages
- Appropriate authority for the objects used within the packages

These packages should be rebound at the time the privileges are revoked.

If you have defined a trigger based on one or more privileges and you lose one or more of those privileges, then the trigger cannot be used.

Chapter 4. DB2 Universal Database GUI Tools

The tools integrated with the DB2 Universal Database Version 5.0 include support for the following functions:

- Database Administration
- Performance monitoring and tuning
- Advanced database systems management
- Additional tools integrated with DB2 Universal Database
- Bidirectional data replication
- Object Relational Extensions
- Web Connectivity (Net.Data)
- Querying, reporting and charting (Lotus Approach)

This chapter will explain the GUI tools that are included in the first five items listed above.

4.1 Presenting the GUI Tools

After finishing the installation of DB2 Universal Database, the DB2 Desktop Folder in Figure 60 will be displayed on Windows NT machines.

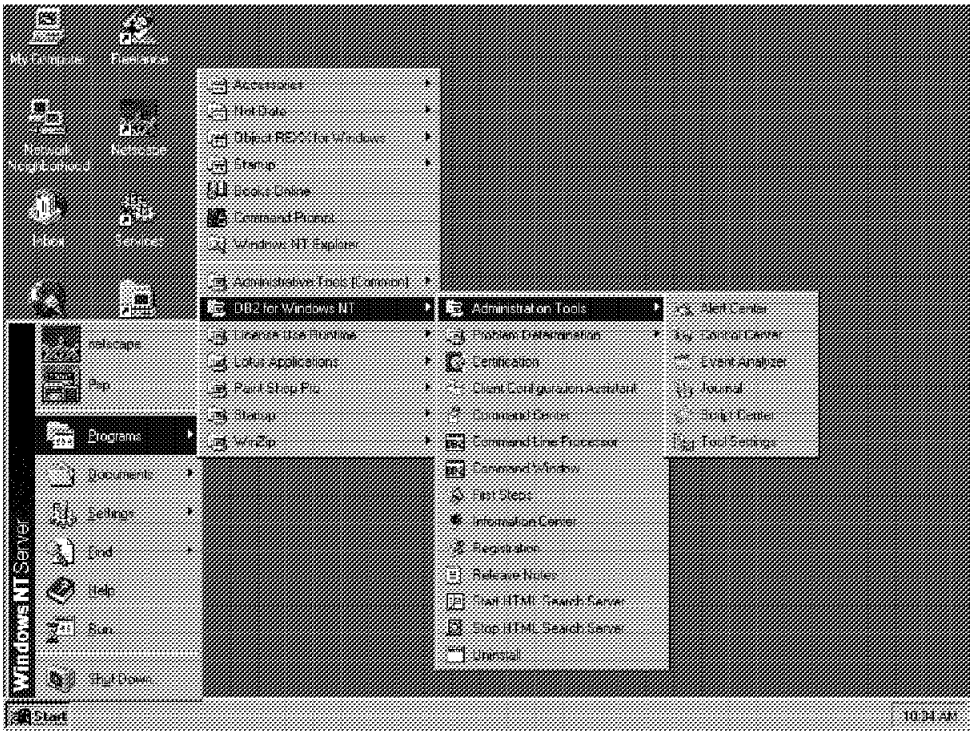


Figure 60. Windows NT Desktop Showing Contents of DB2 Folders

From the DB2 Desktop Folder the user can access any tool by clicking on the icon that represents the function to be performed.

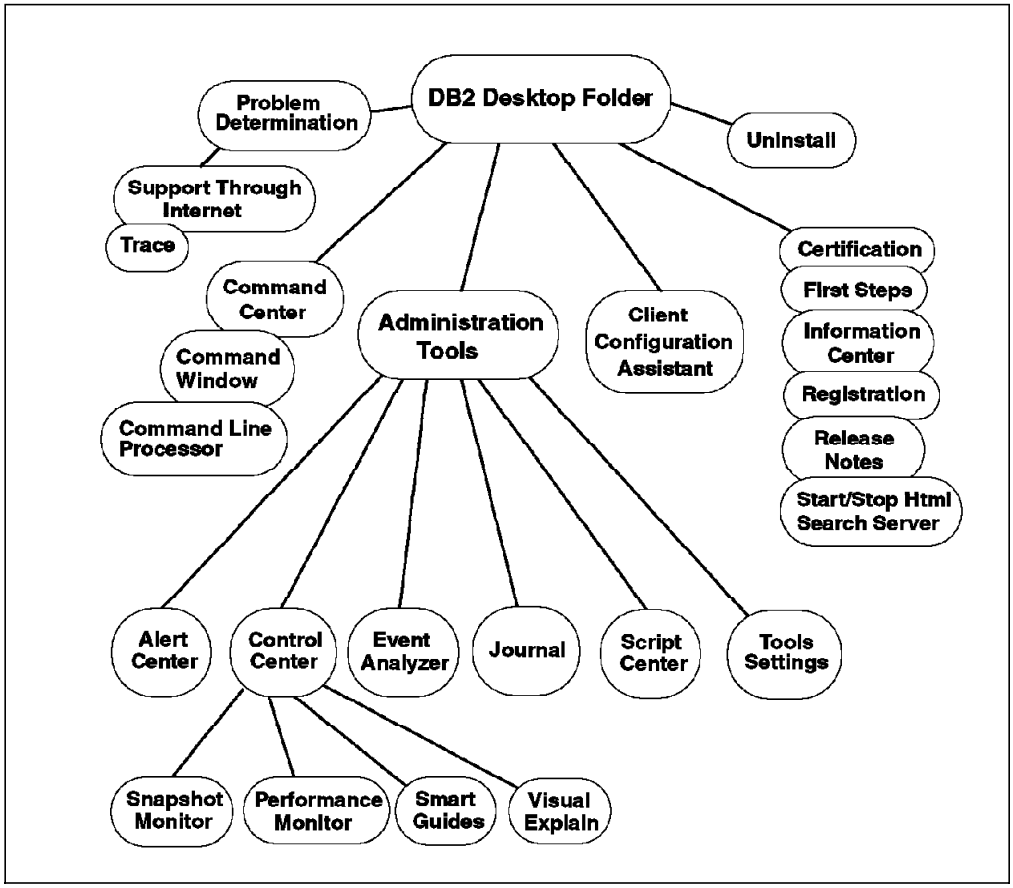


Figure 61. Roadmap to the GUI Tools in DB2 UDB V5

Figure 61 shows a roadmap to the GUI tools displayed on the desktop folder.

Problem Determination is a folder that contains support information and utilities, as well as the trace utility. The trace utility should be used only when directed to do so by an IBM Support Center representative or by a technical support representative. Information gathered with this tool should be sent to IBM for analysis only upon direction from the IBM Support Center.

Command Window is an option on Windows NT and Windows 95 platforms. It is intended to support non-interactive processing of DB2 commands and SQL statements from a DOS Window. It can be invoked at command prompt with the `db2cmd` command.

Command Line Processor invokes an interactive session of DB2. It can be invoked with the `db2` command from any command prompt on OS/2 or DB2 Command Window on Windows NT and Windows 95.

Command Center is the graphical version of Command Line Processor. It can be invoked with the `db2cc -p` command from any command prompt.

Administration Tools is a folder that contains icons for the following tools: (the command to invoke each tool from a command prompt is shown in brackets)

- **Alert Center** is a tool to monitor your system for early warnings of potential problems. (`db2cc -a`)
- **Control Center** is the main tool for administration tasks. (`db2cc`)

- **Event Analyzer** is a tool that helps to analyze the event monitor files generated for a database. (db2cc -e)
- **Journal** is a tool which displays jobs, recovery history, alerts, and DB2 error messages. (db2cc -j)
- **Script Center** is a tool to create mini applications called scripts, which can be stored and invoked at a later time. (db2cc -s)
- **Tools Settings** is a notebook used to change the settings for Control Center, Alert Center, and Replication. (db2cc -t)

Client Configuration Assistant is a tool for easy configuration of access to remote databases. It can be invoked from any command prompt with the db2cca command.

Certification displays information about the IBM DB2 Certification Program.

First Steps is a program configured to be run automatically the first time after DB2 installation. It allows the user to create a sample database and preview some information about the product.

Information Center opens a notebook that allows the user to search information by topics, view the DB2 Universal Database List of Books as well as code examples and other documentation.

Registration allows the user to register as a DB2 UDB V5 customer and receive information about new releases, upgrades or new versions of the product.

Release Notes displays the release notes issued with DB2 UDB V5.

Start/Stop Html Search Server allows the user to start or start the Html Search Server. This utility enables searches to be performed on the DB2 UDB Books which are in html format.

Uninstall is the utility used to safely remove the DB2 programs copied to the system at installation time. Before proceeding with the uninstall process, the user will be prompted for confirmation. The process will remove the following components:

- Shared program files
- Standard program files
- Folder items
- Program folders
- Program directories
- Program registry entries

Other utilities, such as **Performance Monitor**, **Snapshot Monitor** and **Visual Explain** are invoked from the Control Center tool. Performance Monitor and Snapshot Monitor can also be invoked from any command prompt with the db2pm command, and Visual Explain with db2vexp command (additional parameters are required in each case). Also, NT Performance Monitor can be used to monitor NT databases.

Some of the administration functions have **SmartGuides**, that are step-by-step notebooks that prompt the user to fill in the information necessary for the task to be performed.

4.2 Configuration Required to Enable the Client GUI Tools

This section will detail the configuration tasks necessary in order to enable the automation of some functions achieved by the GUI tools, such as remote administration of DB2 servers, scheduler of jobs and the automatic cataloging of nodes and databases (*Discovery*).

4.2.1 DB2 Administration Server

DB2 UDB introduces the DB2 Administration Server for managing servers both locally and remotely. The DB2 Administration Server is code that runs in the DB2 Server in order to enable certain client GUI tools functions.

In general, the type of operations performed using the DB2 Administration Server are:

- Query the operating system configuration information
- Query the operating system for user and group information
- Start and stop other DB2 instances
- Execute scheduled jobs locally or remotely
- Collect information for Discovery to be returned to remote clients

The DB2 Administration Server is a special DB2 instance enabled to perform administration tasks. During installation, it is created and can be configured to start when the operating system is booted. All remote administration tasks will be sent to the DB2 Administration Server for local execution.

On the server, the `db2admin start` command is used to start the DB2 Administration Server during system initialization.

4.2.2 Connectivity and Protocol Configuration

DB2 UDB simplifies the task of connecting a client to databases on servers. It automates many of the steps performed on the client and the server.

On the client The user asks DB2 to find databases that are available for access. DB2 will search the network and return a list to the client system. The user can select the database to connect to and leave it to DB2 to configure the connection.

This automation is available on the client via the Client Configuration Assistant.

On the server The customer is presented with a list of protocols that DB2 has found are installed on the workstation. The user can select the ones that the client will use to connect to the server and leave DB2 to configure them.

This automation is available on the server in two forms:

- **During the installation process**
  - On OS/2 and Windows NT this facility is used to set up the default instance for communications, so it can be connected to after reboot.

- On UNIX, a new installation tool, db2setup, provides the facilities to accomplish initial setup of server communications as well as common tasks, such as instance creation.

- **After installation**

For configuration of instances created after the installation and ongoing maintenance, the user is able to:

- Configure new protocol and change default parameter values.
- Maintain existing protocols by changing parameter values.
- Remove a protocol from configuration.

These tasks are performed in different ways depending on the platform:

- Using the Control Center on OS/2, Windows NT and Windows 95 platforms. Reconfiguration of protocol information is invoked by selecting *Setup Communications* from the instance level object in the Control Center. On OS/2 and Windows NT is able to maintain local and remote instances on all server platforms. On Windows 95 is able to maintain only remote instances.
- Using *db2 update admin cfg* on UNIX platforms. This is invoked from an OS command and is able to manage only local instances.

4.3 GUI Tools in DB2 UDB V5

This section will describe the functionality of the GUI tools presented in the Roadmap.

It is possible to configure DB2 to be started automatically (assuming DB2 is not running) the first time a user starts a DB2 UDB GUI tool after system restart. This is done using the Tools Settings.

4.3.1 Client Configuration Assistant

This tool is an enhanced version of Client Setup in DB2 V2. It is intended to automate the process of adding a database to the list of DB2 data sources that a client can connect to in the network.

Additionally, it enables the user to perform ODBC administration, application binding, and so on, all from the single graphical tool for client administrative tasks presented on Figure 62 on page 142.

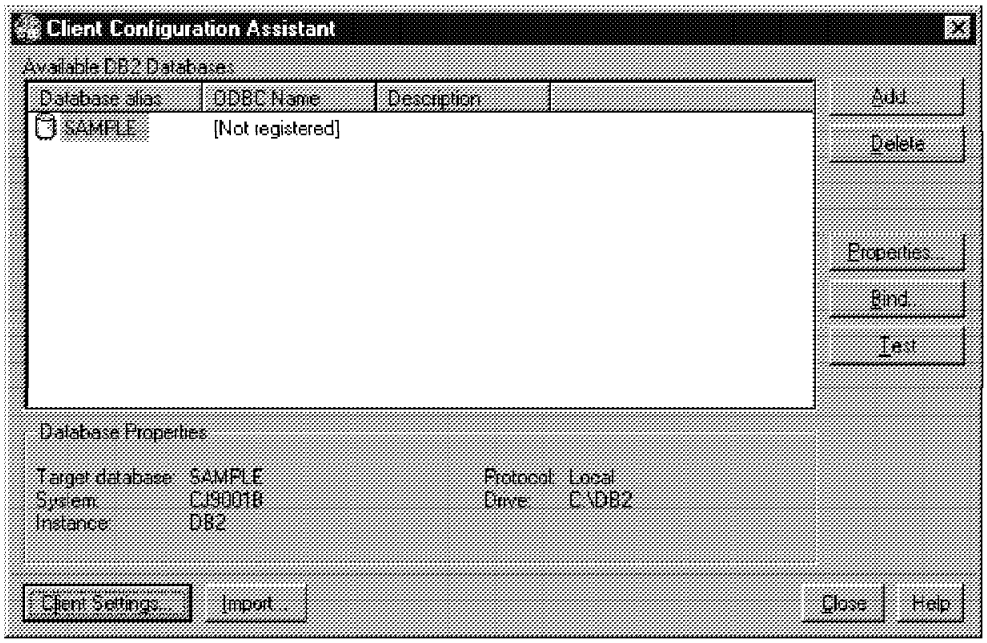


Figure 62. Client Configuration Assistant

It is important to remember that the Client Configuration Assistant tool can configure only the client it is running on. It cannot be used to configure remote clients.

The following functions can be performed with the Client Configuration Assistant:

- Configuration of connections to DB2 UDB and DRDA databases. Discovery functionality is only supported by DB2 UDB databases. Configuration of connections to DRDA databases is supported if:
  - **DB2 Connect Enterprise Edition** is installed on a remote server system. In this case, the user is able to see DRDA databases in the list of databases that can be added along with other DB2 databases on the LAN (as they are already cataloged on the remote server system).
  - **DB2 Connect Personal Edition** is installed on the local system. In this case, the user is able to connect directly to a DRDA database by manually specifying APPC or TCPIP communications information to catalog the DRDA database.
- DB2 Connection testing allows the user to test the connection to the database using the CLI connect function.
- Connection Information maintenance. The user is able to change:
  - General information, such as database alias or description.
  - Connection information, such as the remote database name or communication protocol and parameter settings. If other databases are using the configuration to be changed, the user is presented with a panel that gives the option to change connection information for all databases at the same time. If a global change is not accepted, other databases will continue to use the old configuration.

- CLI/ODBC Administration allows the user to catalog selected databases as ODBC data sources. The user can select an optimized configuration from a list or manually configure ODBC data source parameters.
- Bind applications to database. This option allows the user to perform bind operations against the database via two options:
  - Bind utilities, where the user is presented with a list of the DB2 UDB utilities that can be bound against the server (the user no longer needs to know the bind file names).
  - Bind applications, that support application binding using a bind file or list.
- Remove database entries from the database directory and maintain ODBC data source information and node directories as required.
- Maintenance of database manager configuration parameters for the client instance.
- The changing of MVS passwords on systems where the user's DB2 MVS databases reside.
- Maintenance of server lists that will be used by directed discovery. Each entry in the list contains information needed to connect the DB2 Administration Server on a machine, such as protocols and parameters to be used.

4.3.2 Administration Tools

In the Administration Tools folder are a set of graphical administration tools specially designed to help database administrators manage and administer DB2 databases.

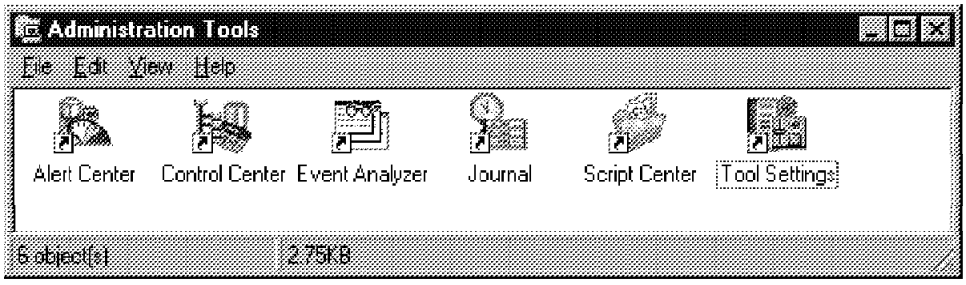


Figure 63. Administration Tools Folder

The administration tools shown in Figure 63 may be installed on any OS/2 or Windows NT DB2 Server system.

The same tools are also optionally installable on any OS/2, Windows NT or Windows 95 DB2 Client. This client system can administer remote DB2 UDB databases on OS/2, Windows NT, AIX, HP-UX or Solaris Operating Environment.

The following sections will explain the functionality of each tool included in the Administration Tools folder.

4.3.2.1 Alert Center

The Alert Center is used to monitor your system for early warnings of potential problems or to automate actions to correct problems.

The Alert Center window is a simplified view of the Snapshot Monitor that shows the results of monitoring. It groups alarm warnings together to tell the user where the problems are. It will automatically open to display any monitored objects that are in a state of alert if an object's threshold has been exceeded.

When an object is in a state of alert in the Alert Center window as shown in Figure 64:

- Double-click on the icon to open the Monitoring Details View window which shows all the performance variables being monitored.
- Determine the color of the icon. This color always reflects the condition of the most severe alert within a group of performance variables. A red icon indicates an alarm while a yellow icon indicates warning.
- Double-click on the performance variable to see the data returned. Analyze the data and take the action to correct or solve the problem.

See 4.5.2, "Monitoring Performance" on page 190 for more details on how to use Event Monitors.

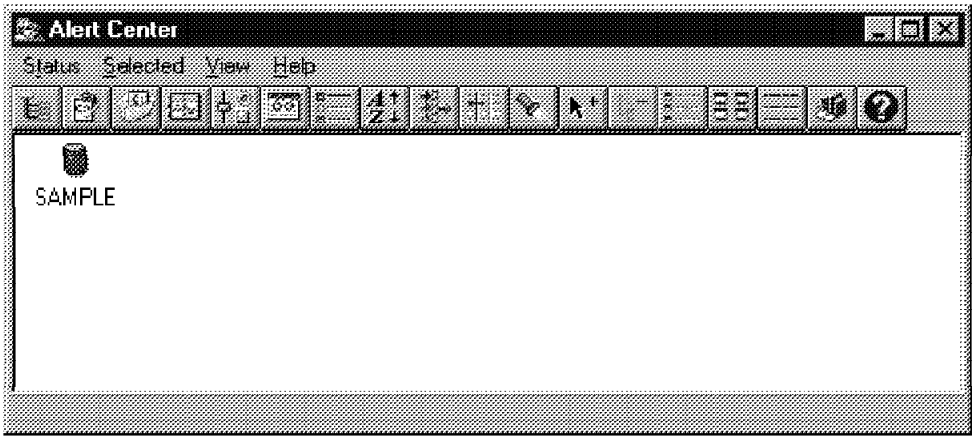


Figure 64. Alert Center

4.3.2.2 Event Analyzer

This tool, shown in Figure 65 on page 145, allows the user to trace performance data produced by DB2 event monitors that have their data directed to files.

There are two methods for reading event monitor traces:

- Specifying the directory where the trace files are located. This allows the user to move trace files from a server and analyze them locally. This can be done even if the event monitor has been dropped.
- Specifying the database and event monitor names. This allows automatic location of the trace files. The event analyzer connects to the database and issues a *select target from sysibm.syseventmonitors* to locate the directory where the event monitor writes its trace files. This method cannot be used if the event monitor has been dropped.

See 4.5.2, “Monitoring Performance” on page 190 for more details on how to use Event Monitors.

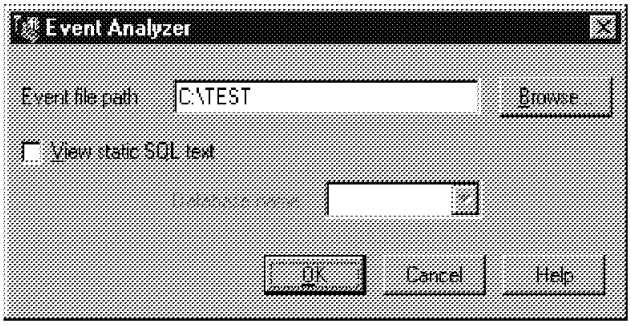


Figure 65. Event Analyzer

4.3.2.3 Tools Settings

The Tools Settings notebook is used to modify preferences for the GUI Tools.

From the Control Center toolbar click on the **Tools Settings** icon or double-click on the icon in the Administration Tools folder. It opens the Tools Settings notebook as shown in Figure 65. You can change settings for:

General To enable or disable hover help, and automatically start local DB2 on tools start-up.

Alert Center To specify whether to display the Alert Center window on new warnings and alarms, on new alarms only or never.

Replication To specify capture of source table columns before and after image, or capture after image only. Also to define creation of target tables at subscription, drop target tables if subscription is removed, and drop table space when empty.

Node Status (Only for partitioned databases). To enable node status reporting. Also to define actions to take if a node is down or its state is unknown.

After modifying the Control Center settings, close the Tools Settings window and changes will take place immediately.

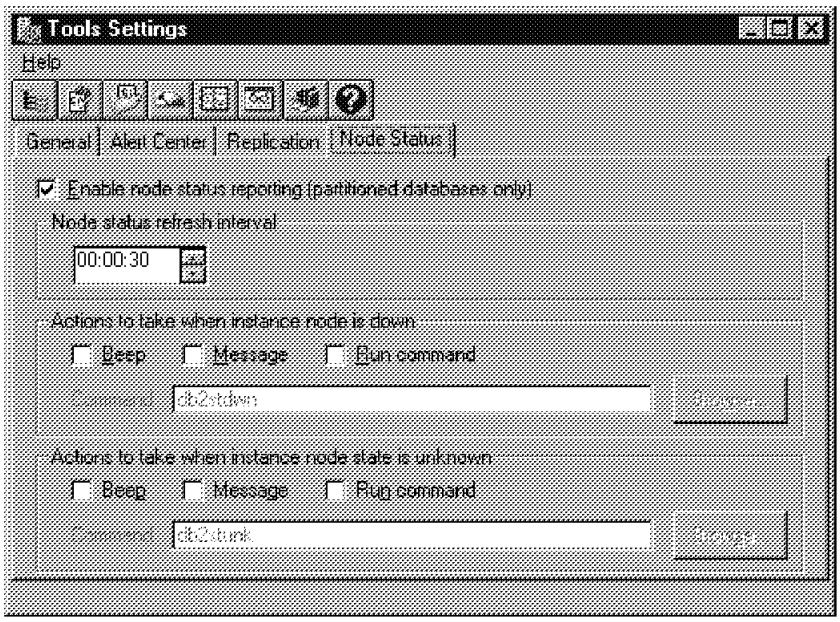


Figure 66. Tools Settings

4.3.2.4 Control Center

The Control Center provides the user with the tools necessary to perform common database administration tasks. This tool:

- Provides a seamless integration of the administration tools
- Gives a clear overview of the entire system
- Enables remote database management
- Provides step-by-step assistance for complex tasks when using the SmartGuides

You can choose to have the Control Center installed on a Windows NT or OS/2 DB2 Server or on a Windows NT, Windows 95, or OS/2 DB2 Client.

The main components of the Control Center panel are shown in Figure 67 on page 147.

Menu Bar Used to access Control Center functions and online help using menus.

Tool Bar These icons may be used to access the other tools, such as Journal, Script Center and so on. These options can also be selected in the View menu.

Objects Tree Shown on the left side of the Control Center window and contains all the objects that can be managed from the Control Center as well as their inter-relationship.

Contents Pane On the right side of the Control Center and contains the objects that belong or correspond to the object selected in the Objects tree.

Contents Pane Toolbar These icons are used to tailor the view of objects and information in the Contents pane. These functions can also be selected in the View menu.

Hover Help Provides a short description for each icon on the toolbars.

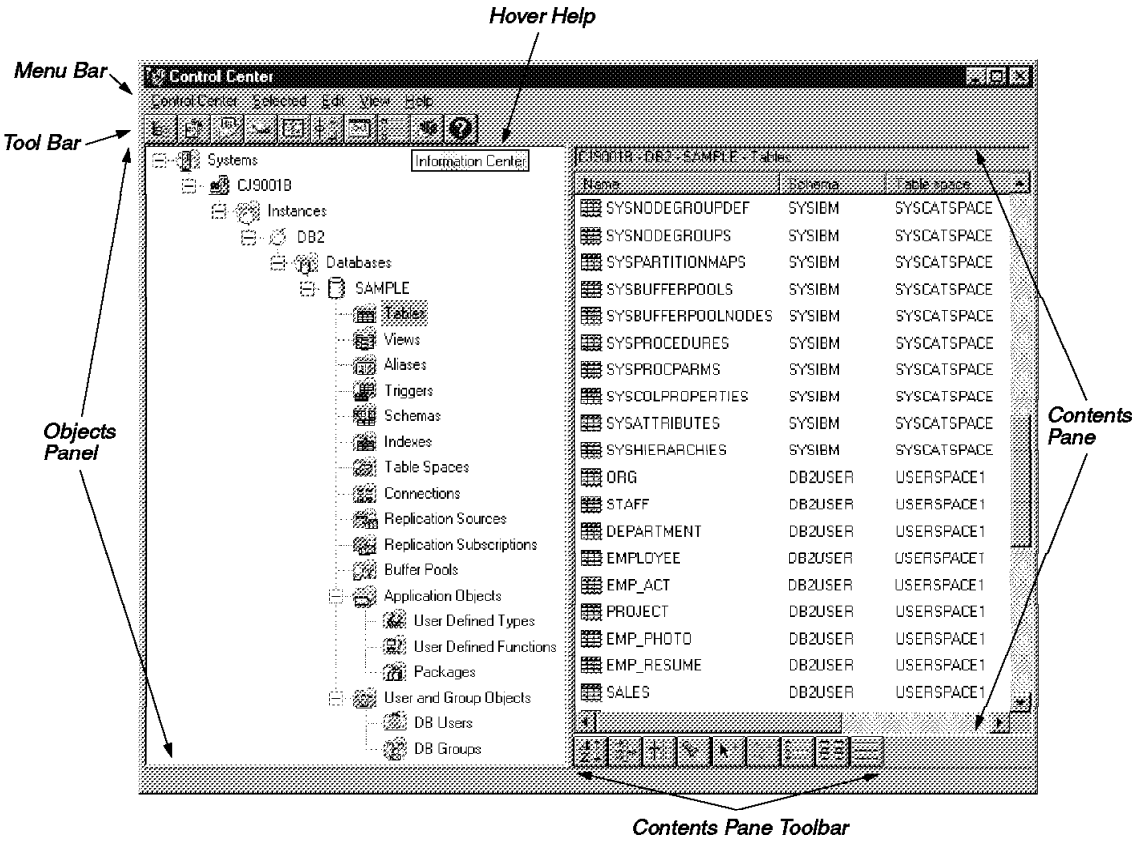


Figure 67. Main Components of the Control Center

The Systems icon represents a machine. To display all the systems that your system is connected to and which have DB2 installed, expand the object tree by clicking on the **plus sign (+)** on Systems. The local workstation is represented by a System icon labeled with your machine's name and other remote systems defined are labeled with their names.

An instance is a logical database manager environment similar to an image of the actual database manager environment, and is a directory on your machine. To display the instance(s) on each system, expand the object tree by clicking on the **plus sign (+)** on your machine's name or remote system name. A default local instance named DB2 is created at installation time.

Several instances can be created on the same workstation, and can be used to separate the development environment from the production environment, tune the database manager to a particular environment, and protect sensitive information from a particular group of people.

A relational database is a collection of data that is stored in tables and belongs to a unique instance. To display the database(s) created on a particular instance, expand the object tree by clicking on the **plus sign (+)** on DB2 instance or other instance name.

At the database level the user can manage objects such as tables, views, aliases, triggers, schemas, indexes, tables paces, connections, replication sources, replication subscriptions, buffer pools, application objects and user & group objects. The management of some of these objects from the Control Center is explained in 4.4, “Common Tasks Using GUI Tools” on page 153.

4.3.2.5 Command Center

The Command Center allows the user to process SQL statements and DB2 commands using a graphical front-end as seen in Figure 68. It is the GUI version of Command Line Processor that provides an interactive window for the entry and execution of SQL statements or DB2 commands.

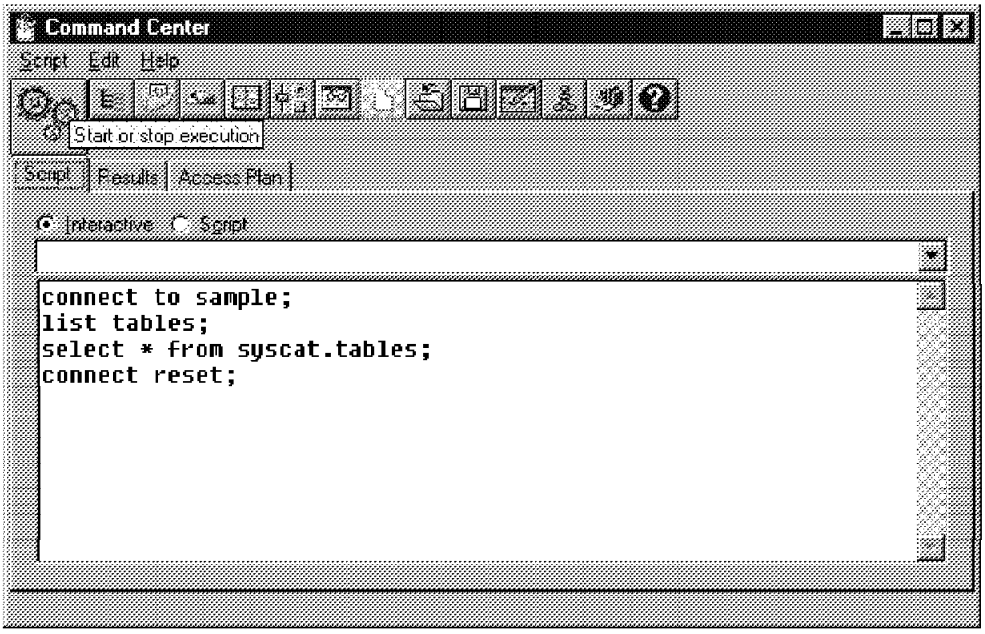


Figure 68. Command Center

The result of the one or many SQL statements and DB2 commands is displayed in Figure 69 on page 149. The user is able to scroll through the results and generate a report if needed. The user can also generate the access plan of any command by clicking on the **Visual Explain** icon.

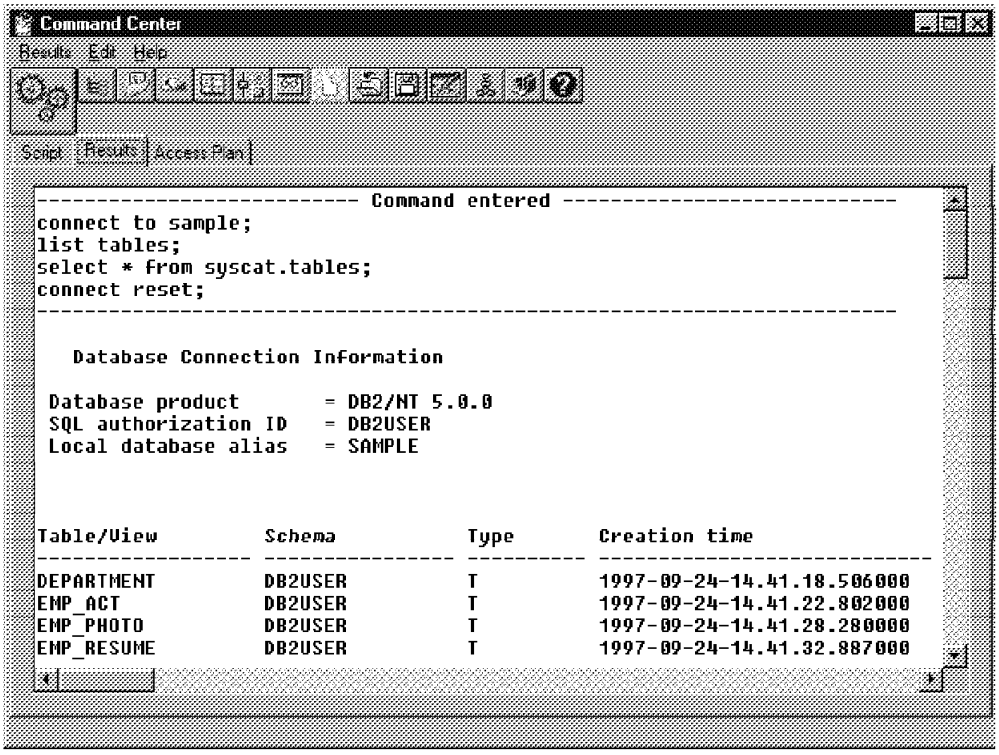


Figure 69. Command Processor Showing Results on Scrollable Window

A save option allows the user to save a sequence of SQL statements or DB2 commands to a script file. This script file can be scheduled using the Script Center and monitored in the Journal.

This GUI version of Command Line Processor uses the same options as traditional CLP for autocommit, SQLCA and SQLCODE display, output file, stop on error, and so on.

Both the GUI and traditional CLP are installed and shown as icons in the DB2 folder. For NT, there is also a Command Window to enable users to write SQL statements and DB2 commands in a non-interactive DOS Windows session.

4.3.2.6 Script Center

The Script Center is a GUI tool that allows the user to create or import previously created scripts, containing OS or DB2 commands, to be immediately executed or scheduled.

Once the script is created, it is registered in the script list which contains a script ID number, the instance where it was defined on, the script name and description, the command type (OS or DB2) and the last modification timestamp.

The user can select a script from the list and edit its contents to be modified and saved, copy the script to a new file or remove it. A script can also be run immediately as shown in Figure 70 on page 150 or scheduled to run later by specifying the job description, execution frequency and actions to be taken when the job succeeds or fails.

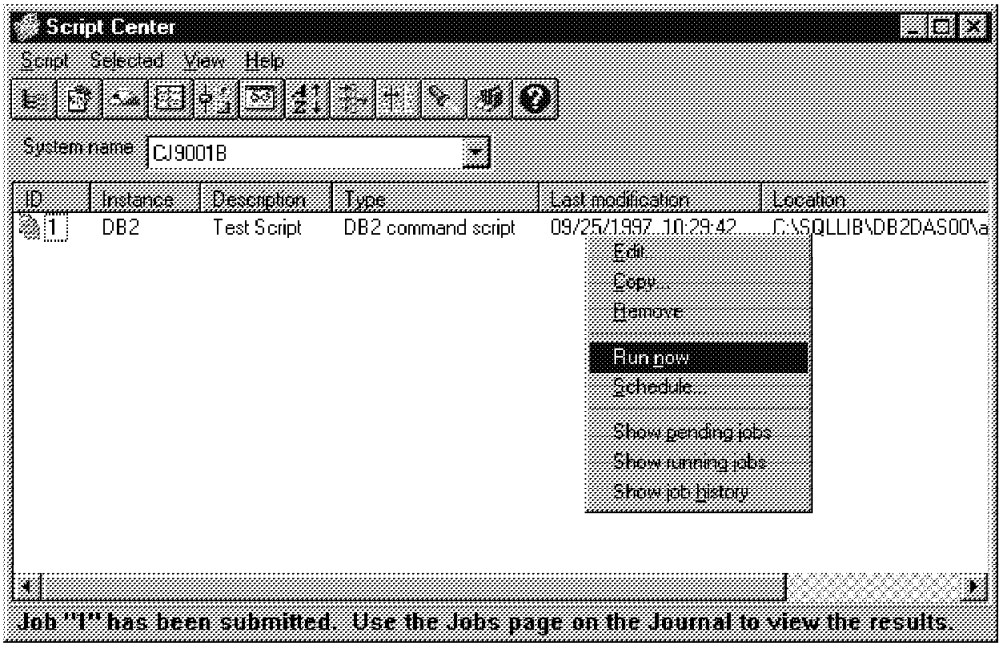


Figure 70. Script Center Showing How to Run a Script

The Show pending jobs, Show running jobs and Show job history options open the **Journal** tool to display the status of scheduled jobs.

4.3.2.7 Journal

The Journal is used to schedule jobs to run unattended. These jobs can be run once or set up to run on a repeating schedule; a repeating schedule is particularly useful for tasks like backups. The Journal also allows you to review the results of jobs that are running unattended. The Journal also displays the Recovery History, Alert Messages and DB2 Error Messages.

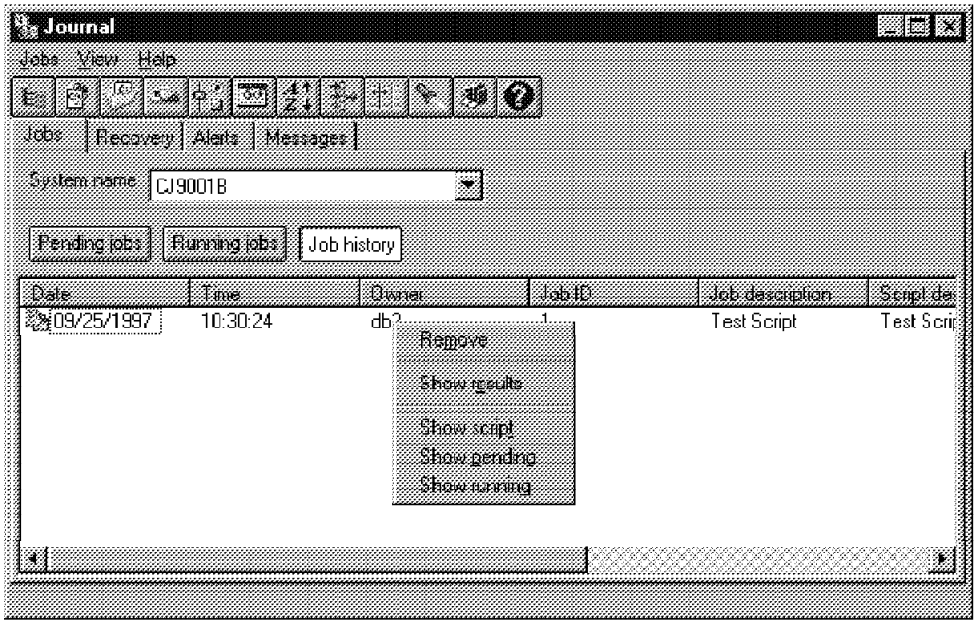


Figure 71. Journal Utility Showing How to Remove a Job

The journal notebook presents the information in four pages:

Jobs To see a list of jobs, all the information about the job and to perform actions on each job, such as remove it, show the scripts associated with it or run it immediately.

Recovery Displays the recovery history. For example, details from a restore operation.

Alerts Shows the log of all alert messages.

Messages Shows the log of all DB2 error messages, as in Figure 72.

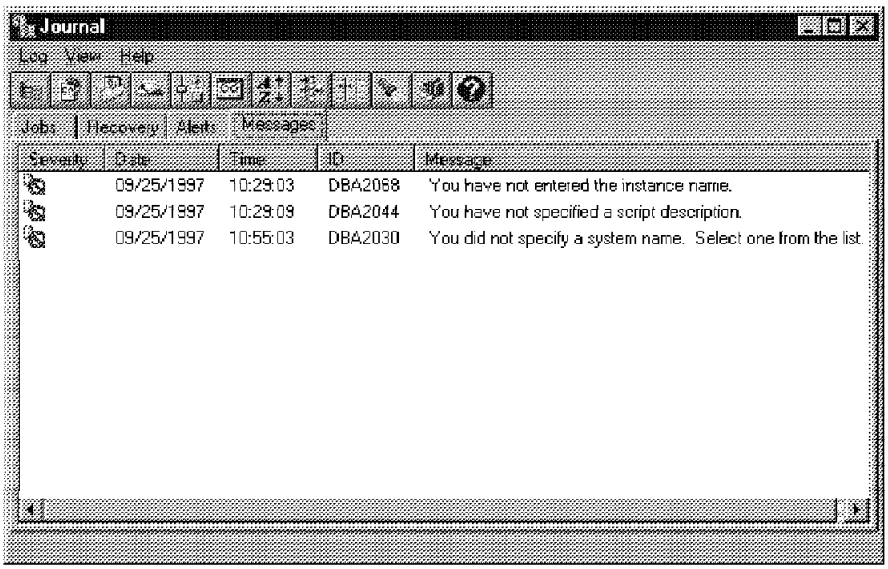


Figure 72. Journal Page Showing DB2 Error Messages

4.3.3 Product Information and Documentation

- **Information Center**

This tool provides the user with quick access to DB2 product documentation. It launches a viewer to display the information. This viewer could be the system help viewer, an editor, or a web browser, depending on the kind of information requested.

From the DB2 desktop folder, select the **Information Center** icon. Use the appropriate tab to look at the information needed, such as tasks, reference information, on-line books in HTML format, troubleshooting information, sample programs or sources of DB2 information on the web.

- **First Steps**

As shown in Figure 73 on page 152, DB2 First Steps is a program that was first available in DB2 V2 as a sequence of recommended steps after installation. Each step is optional and includes:

Create a sample database This step creates a sample database called SAMPLE that can be used to verify that the system is working correctly. The product documentation and sample programs refer to the tables and columns of this database.

Preview product information This step runs the Information Center, from which the user can view the product's information library.

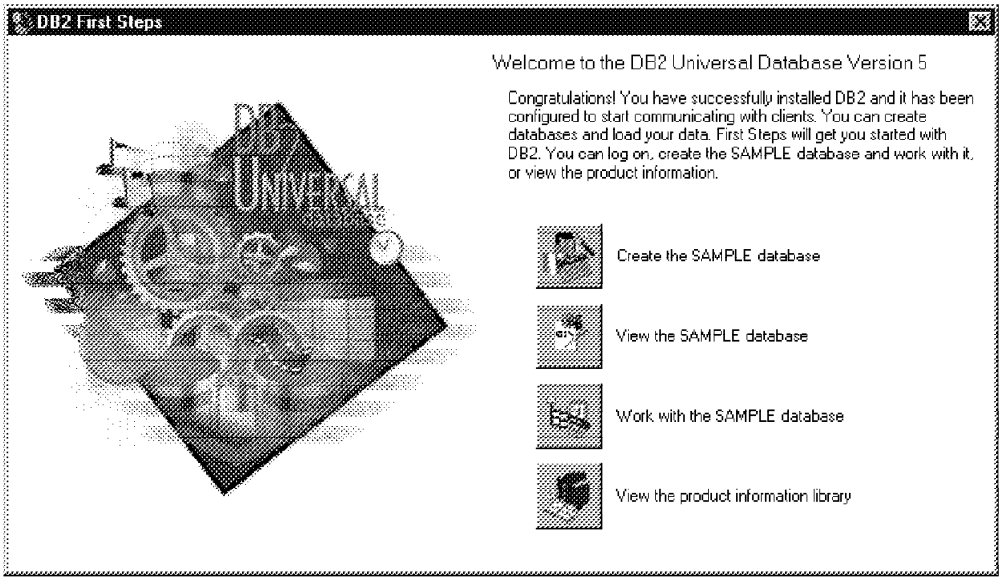


Figure 73. DB2 First Steps

This program is set up to be run at system start-up after the installation of DB2 Universal Database. After that, it can be invoked at any time from the DB2 folder.

4.3.4 Problem Determination

Information about support is provided by *Support Through Internet.* If a user selects this icon, a browser is started and displays links to several sources of on-line support.

For OS/2 users, several service-related tools are packaged together in the Problem Determination folder. Several of these tools require that you have an Internet connection established before you can use them.

A graphical version of the trace utility is also available

4.3.4.1 Trace Facility

DB2 provides a trace facility to trace events, dump trace data to a file, and format trace data into a readable form. You may be requested by DB2 Customer Service to take a trace if the db2diag.log file does not contain enough information to diagnose a problem.

DB2 trace information is either in memory or on disk. If trace collection was activated, trace information is recorded in a chronological order. Each entry in the trace file is called a trace point and is sequential.

The amount of information gathered by a trace grows rapidly. The goal of performing the DB2 trace is to capture only the error situation. Any other activities that do not reproduce the error situation should be avoided (for example, starting the database manager instance with db2start or connecting to

a database with db2 CONNECT). When taking a trace, reproduce the smallest, recreatable scenario and capture it for further analysis.

You should consider that the process of performing a trace will have a global effect on the behavior of the DB2 instance. How much degradation you experience is dependent on the type of problem and what resources are being used to gather the trace information.

From the DB2 Desktop Folder select Problem Determination and double-click on the **Trace Utility** icon. To start the trace utility you should specify the options for trace execution (all are optional):

-m *mask*	It specifies trace record types. The mask variable consists of four byte-masks separated by periods, and correspond to products, event types, components and functions.
-e *max_sys_errors*	Limits the number of system errors the trace will hold.
-r *max_record_size*	Limits the size of trace records. Longer trace records are truncated.
-l *buffer_size*	It specifies the retention of the last trace records, the first records are overwritten when the buffer is full.
-i *buffer_size*	It specifies retention of the initial trace records, no more records are written to the buffer if it is full. Also can be used to specify buffer size.

Click the **Start** push button and repeat all the steps to recreate the problem scenario.

To save the trace output, click on the **Save as** push button. Select **Generate formatted trace file** if you want the full trace information or **Generate control-flow trace file** if you want only the control flow of the trace information in a nested format. You can also specify some trace save options and click on **Save**. You must save the trace output before stopping the trace utility.

The user can activate the trace utility several times by turning tracing on, producing a dump file for a particular scenario, formatting the dump file, and turning tracing off again. Then the user can repeat these steps for a different scenario to be analyzed.

4.4 Common Tasks Using GUI Tools

This section will show how to perform some common database administration tasks using the DB2 UDB GUI Tools and SmartGuides.

4.4.1 Configure Access to Remote Databases

The Client Configuration Assistant is used to configure access to remote databases.

4.4.1.1 Use an access profile

There are two kinds of access profile:

Server Profile	Contains the information held at the server regarding communications.
Client Profile	Contains information about the remote databases configured at the client.

- **Server Access Profile**

To generate a Server Access Profile:

1. From the Control Center click the **mouse button two** on the system that contains the database you want to connect to;
2. Select **Generate access profile**.
3. Specify a directory and file name to be generated.

To use a Server Access Profile:

1. From the Client Configuration Assistant, select **Add**.
2. Select the **Use an Access Profile** radio button, and click on the **Next** push button.
3. Click on the **Browse** push button to select the server profile that you want to access, or enter the path and filename in the File field.
4. Select a database to be added.
5. Select whether to register this database for ODBC as a user or system data source if needed.

- **Client Access Profile**

To generate a Client Access Profile:

1. Enter `db2cca admin` at a command prompt to start the Client Configuration Assistant in administrator mode;
2. Click on the **Export** push button;
3. Select the databases to be exported from the Available DB2 Databases window, and add them to the Databases to be exported window;
4. Enter a path and file name for the Client profile.

To use a Client Access Profile:

1. From the Client Configuration Assistant, Click on the **Import** push button.
2. Select the path and filename of the client profile you want to import and click on **OK**.
3. The Import Client Profile window opens. Select the items you want to import and click on **OK**.

Now you are ready to test your database connection.

4.4.1.2 Search the network

This option will use the discovery utility to look for available servers on the network in one of two ways:

- If discovery is set to search, the CCA will look for the database servers available for the protocol defined for discovery. As in the case of an access profile, you will be presented with a list of databases to select from.
- If discovery is set to known, you must first add the system specifying the protocol to be used. Depending on the protocol specified, you will be prompted for:

  TCP/IP Hostname

  NetBios Workstation name and adapter number

  IPX/SPX Internetwork Address

  Named Pipe Computer name and instance

Now select the target database, specify an alias, description, whether you want it to be registered for ODBC or not and test the connection.

In order to be able to use the discovery facility to gather information at network search for configuring remote databases, it is a requirement that:

- On the Server:
  - DB2 Administration Server is installed and configured on the server workstation.
  - DISCOVER at the Administration Server level must be set to KNOWN or SEARCH.
  - DISCOVER_COMM at the Administration Server level must be configured to support one or more protocols.
  - DISCOVER_INST at the instance level must be set to ENABLE.
  - DISCOVER_DB at the database level must be set to ENABLE.
- On the Client:
  - DB2 Client code is installed on the client workstation.
  - DISCOVER at the instance level must be set to KNOWN or SEARCH.
  - DISCOVER_COMM at the instance level must be configured to support one or more protocols.
- Protocol stack(s) on client and server are installed and configured, so that they are fully functional.

4.4.1.3 Manually configure a connection to a DB2 database

The SmartGuide will prompt you for the protocol to be used and some additional information depending on your protocol selection. Add the target database name, alias and description, select to register for ODBC or not and test the connection. Use this option to setup a connection to a DB2 Common Server V2 system.

4.4.1.4 Manually configure a DRDA connection to a database

The SmartGuide will prompt you for the operating system (MVS/ESA, OS/400 or VM/VSE) and protocol (TCP/IP or APPC) to be used to connect to the DRDA database. Depending on your protocol selection you will need to know some definitions, add the target database name, alias and description, select to register for ODBC or not and test the connection.

4.4.2 Creating Database Objects

This section details the steps necessary to create the principal database objects, using notebooks and the SmartGuides when available.

4.4.2.1 Database

A database comprises information which is logically connected.

- **Create Database SmartGuide**

The Create Database SmartGuide helps you to create a database, assign storage and select basic performance options. From the Control Center, click the mouse button two on the **Databases** icon and select **Create**, then **New** from the pop-up menu.

The Create Database SmartGuide appears. There are seven pages to complete:

Database name	Specify the new database name, default drive, comment and database alias.
User tables	Specify how the data files will be managed, either by the operating system (low maintenance) or by the database manager (high performance). Based on your selection, you should specify the container characteristics.
Catalog tables	Allocate disk space for the system catalog tables which may be stored in SMS or DMS table spaces.
Temporary tables	Allocate disk space for temporary data which may also be stored in SMS or DMS table spaces.
Performance	Modify the default parameter values for how the database reads and writes data to the disk. These parameters can be modified later, except for the extent size.
Region	Specify the territory and code page set the database will use, also how the characters will be compared. These definitions cannot be changed once the database is created.
Other tasks	Includes launching of other SmartGuides, such as Create Table or Backup Database SmartGuides after creating the new database.

Figure 74 on page 157 shows one page of data requested at database creation time in the Create Database SmartGuide.

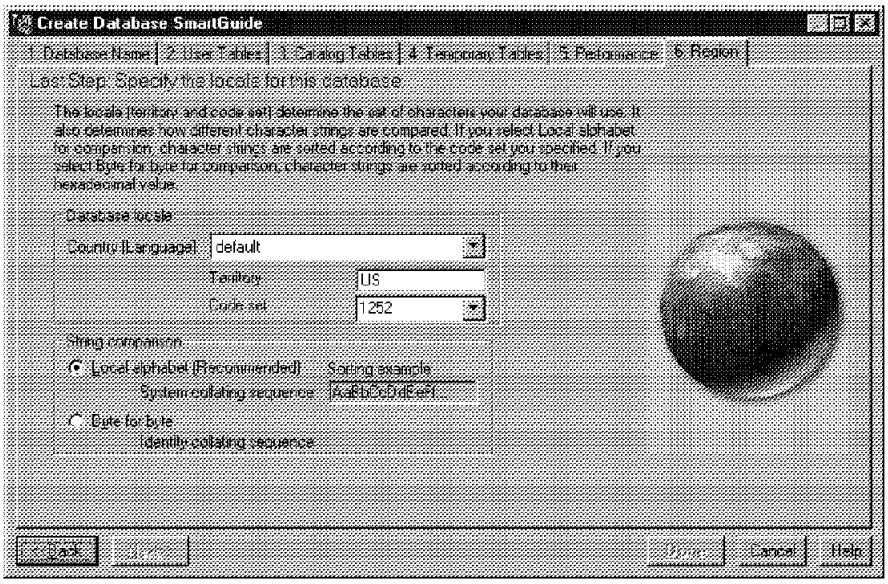


Figure 74. Create Database SmartGuide Showing Region Parameters Definition

- **Create Database from Backup Notebook**

The Create Database from Backup notebook allows you to create a database from a backup image, redefining table spaces and specifying some performance options. From Control Center, click the mouse button two on the **Databases** icon and select **Create**, then **Create from backup** from the pop-up menu.

The Create Database from Backup notebook shown in Figure 75 on page 158 will appear. There are five pages to complete:

Description Specify the new database name and default drive.

Backup Image Specify manually the backup image information or select from a list of backup images, if found.

Containers In case you need to redefine table spaces and redirect containers from a previous backup image.

Roll-forward Specify whether to leave the database in roll-forward pending state or not by applying the logs. Logs can be applied to a point in time or up to the end of logs.

Options Specify if the backup will be taken off-line or on-line and the performance parameters to be used.

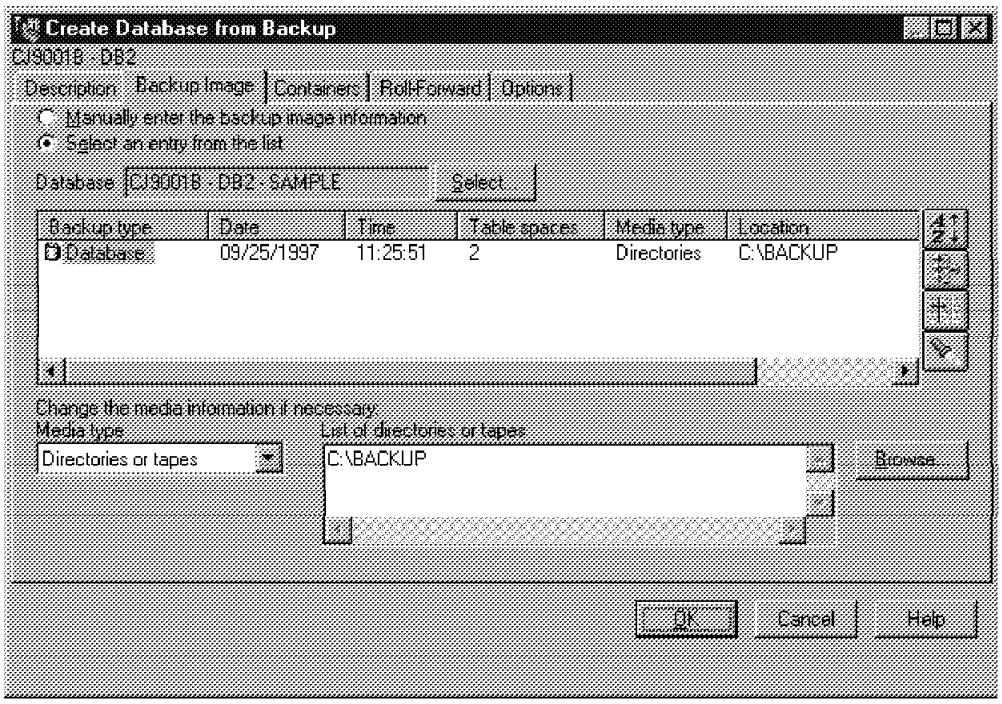


Figure 75. Creating a Database from Backup Image

4.4.2.2 Table spaces

Databases are made up of table spaces. A table space can be either a system managed space (SMS) or a database managed space (DMS).

For an SMS table space, each container is a directory in the file system of the operating system. For a DMS table space, each container is either a fixed size, pre-allocated file, or a physical device such as a disk.

- **Create Table Space Notebook**

The Create Table Space notebook is a GUI tool to help you to create a new table space. It will request some basic information and is intended for experienced users. From the Control Center, click the mouse button two on the **Table Spaces** icon and select **Create**, then **Table space** from the pop-up menu.

Figure 76 on page 159 shows the minimum data required to create a table space.

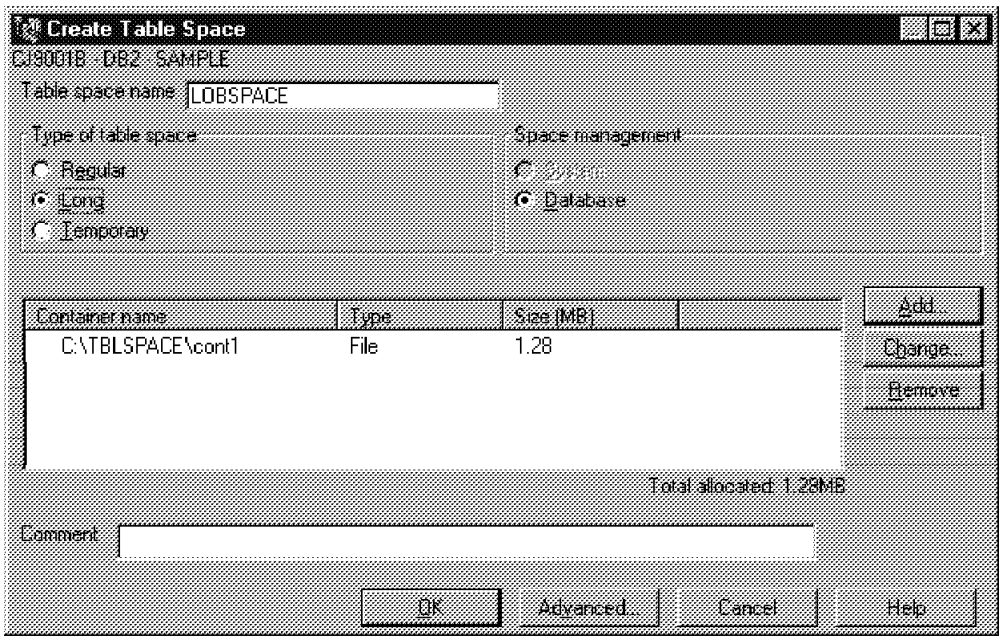


Figure 76. Create Table Space Notebook

- **Create Table Space SmartGuide**

The Create Table Space SmartGuide helps you create a new table space and set basic performance options. From the Control Center, click the mouse button two on the **Table spaces** icon and select **Create**, then **Table space using SmartGuide** from the pop-up menu.

The Create Table Space SmartGuide shown in Figure 77 on page 160 will appear. There are six pages to complete:

Name	Specify the new table space name.
Type	Choose the type of data that will be stored in this table space, as shown in Figure 77 on page 160.
Management	Specify either system or database managed space.
Containers	Define the containers.
Read/Write	Change the default parameters for number of containers, extent size and prefetch size.
Drive Speed	Specify additional information about the disk drive characteristics.

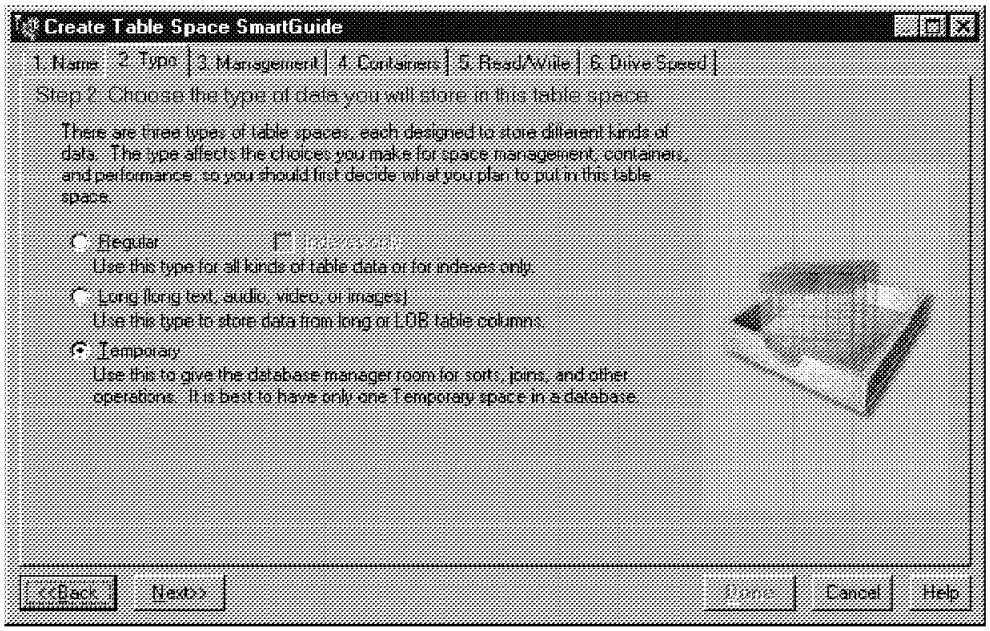


Figure 77. Create Table Space SmartGuide

4.4.2.3 Tables

A table consists of data logically arranged in columns and rows. The data in the table is logically related, and relationships can be defined between tables.

Before you create a table, you must ensure the table space in which you are to store the table exists and has sufficient free space. If a table space is not explicitly defined, the default USERSPACE1 table space will be used.

After you have determined how you intend to organize your data into tables, the next step is to create a table. This can be done using the Create Table notebook or the Create Table SmartGuide.

- **Create Table Notebook**

The Create Table notebook is a GUI tool to help you design your new table. It is intended for experienced users. From the Control Center, click the mouse button two on the **Tables** icon and select **Create**, then **Table** from the pop-up menu.

The Create Table notebook in Figure 78 on page 161 is displayed. There are five pages to complete:

Table	Specify the table description and physical storage, such as table schema and table name, table spaces for data, index and long data types. The user can also specify whether or not to enable the table for Data Replication.
Columns	Specify details of the columns in the table.
Primary Key	Optional: Specify the columns that make up the primary key. You can select columns to be part of the primary key from a list of available columns and specify the constraint name.

Foreign Key Optional: Specify referential constraints with other tables. You can Add, Change and Remove references to parent tables, specify rules on update or delete and a constraint name.

Check constraints Optional: Specify check constraints on the table. You can Add, Change and Remove definitions of check constraints.

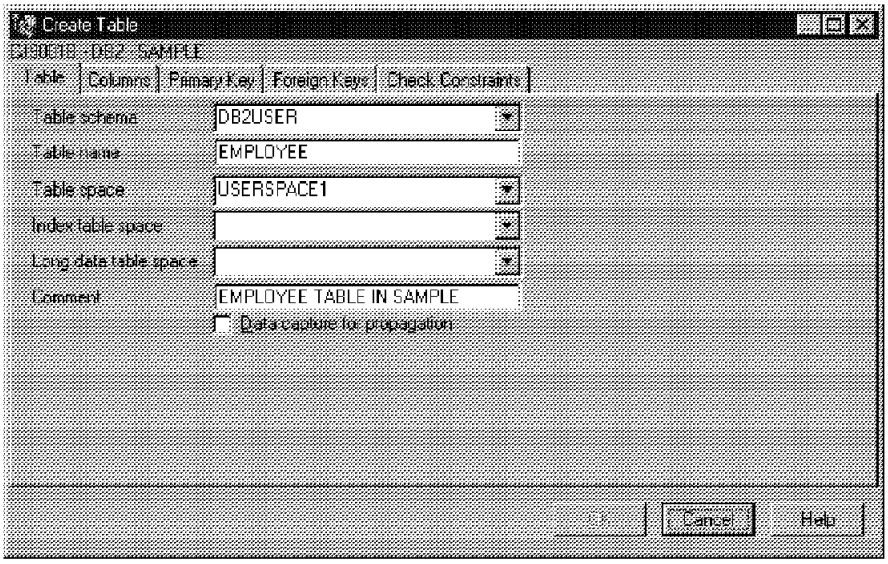


Figure 78. Create Table Notebook

When you have finished, select **Done** to create the table.

- **Create Table SmartGuide**

The Create Table SmartGuide helps you design columns, create a primary key for the table and assign it to one or more table spaces. From the Control Center, click the mouse button two on the **Tables** icon and select **Create**, then **Table using SmartGuide** from the pop-up menu.

The Create Table SmartGuide is displayed. There are six pages to complete:

Name Specify the table schema and table name.

Columns Specify details of the columns in the table. You can Add, Change, Remove columns or Select from a list of available columns as shown in Figure 79 on page 162.

Order Optional: Specify the order of the columns in the table.

Primary Key Optional: Specify the columns that make up the primary key. You can select columns to be part of the primary key from a list of available columns and specify the constraint name.

Table space Optional: Specify the use of a new or existing table space for data, index and long data types.

Edit Columns Optional: Edit and modify the columns definitions.

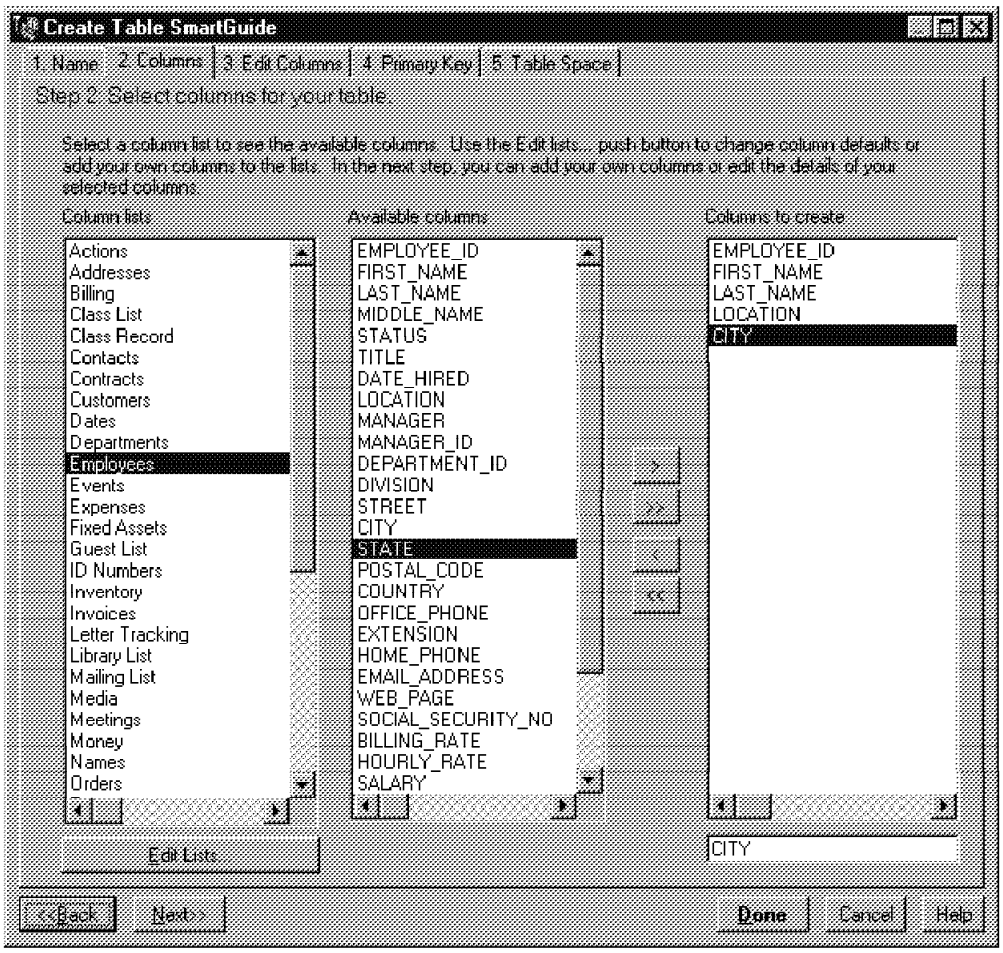


Figure 79. Create Table SmartGuide

When you have finished, select **Done** to create the table.

4.4.2.4 Views

A view is an alternate representation of data from one or more tables. It can include all or some of the columns or rows contained in the tables on which it is defined.

From the Control Center, click the mouse button two on the **Views** icon and select **Create** from the pop-up menu.

The Create View notebook in Figure 80 on page 163 will prompt you for the view schema and view name, the SQL statement to define the data in the view, check options and a comment.

Complete this information and click **OK** to create the view definition.

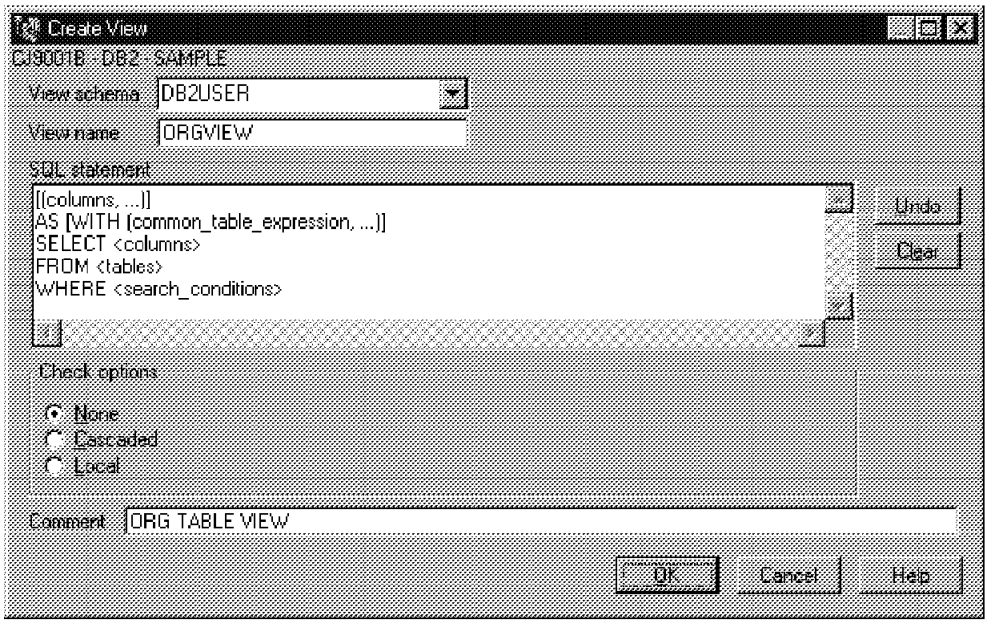


Figure 80. Create View Notebook

4.4.2.5 Indexes

An index is a set of pointers that are logically ordered by the values of a key. Indexes provide quick access to data and can enforce uniqueness on the rows in the table.

From the Control Center, click the mouse button two on the **Indexes** icon and select **Create** from the pop-up menu.

The Create Index notebook in Figure 81 on page 164 will prompt you for the index schema and name, the table schema and name, to select one or more columns as key from the list and to specify if this is a unique index or not. Add a comment and click **OK** to create the index.

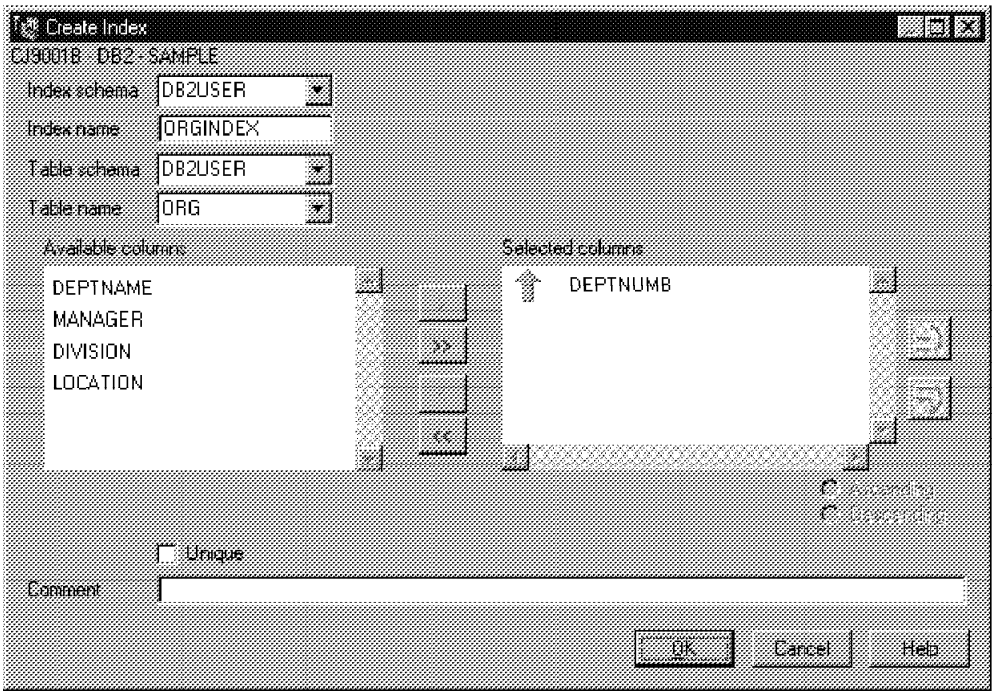


Figure 81. Create Index Notebook

4.4.3 Moving data

DB2 provides utilities to help you move data in and out of your database. These utilities are Copy, Import, Export and Load. Another method of moving data between databases is data replication.

4.4.3.1 Copy

You can copy a table into the same database or a different database in the network.

From the Control Center, click the mouse button two on the table you want to copy and select **Copy** from the pop-up menu.

In the Copy notebook shown in Figure 82 on page 165, you specify some information for the new table, such as the system name (local or remote system name), the instance name where the target database is, the database where you want to create the new table in, a table schema and a table name.

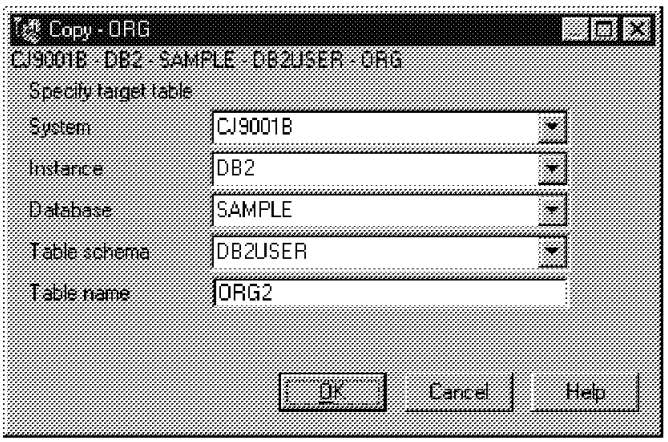


Figure 82. Copying a Table

When you have finished, select **OK** to copy the table.

4.4.3.2 Import

You can use the Import utility to insert data from an input file into a table you selected or create a new table and populate it with data from the input file. You need SYSADM or DBADM authority and CONTROL or INSERT privileges on the table.

From the Control Center, click the mouse button 2 on the table you want to import data into and select **Import** from the pop-up menu.

The Import notebook shown in Figure 83 on page 166 is displayed. There are three pages to complete:

File Specify the input and message file names, import file types such as ASC, DEL, WSF or IXF (each with their own options), import mode (if you are populating an existing table you can choose between Insert, Insert_Update or Replace, or if you are creating a new table, choose between Create or Replace_Create).

Also, specify the scope of records between each commit and if restarting, the number of record to begin at. If using the compound option, specify the scope of records in the block.

Large Objects Optional: Specify if LOBs are stored in separate files. If large objects (LOBs) are in separate files (LOBSINFILE) specify the LOB paths and LOB file names.

Columns Specify columns to be included depending on the import file type. Also, define the schema and name for the new table.

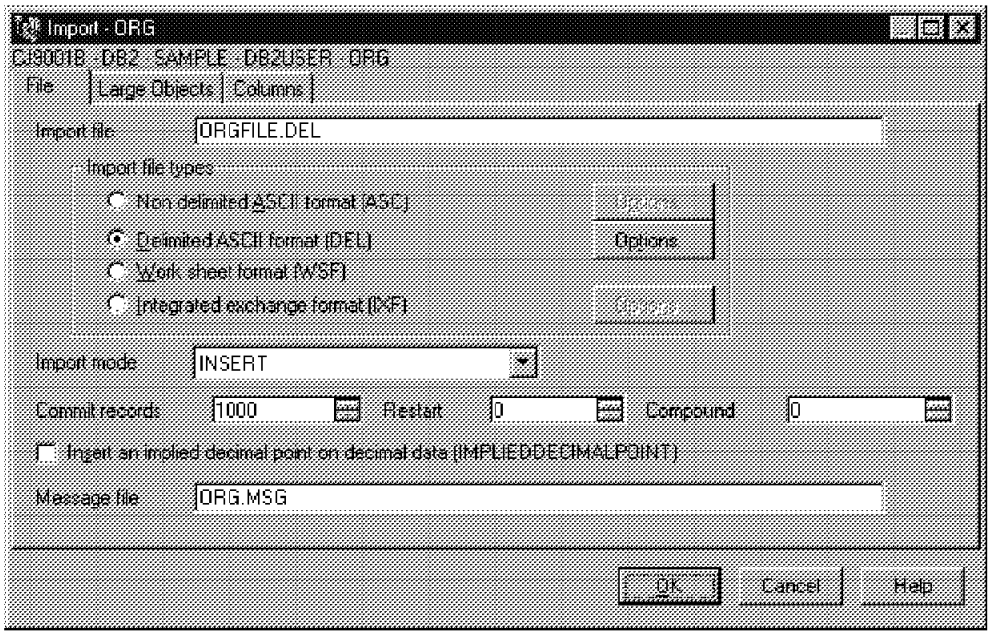


Figure 83. Import Notebook

When you have finished, select **OK** to begin the import process.

4.4.3.3 Export

You can use the Export utility to export data from a table or view into an output file. If the output file already exists, data in it will be overwritten. You need SYSADM or DBADM authority and CONTROL or SELECT privilege on each table or view.

From the Control Center, click the mouse button two on the table you want to export the data from and select **Export** from the pop-up menu.

The Export notebook as shown in Figure 84 on page 167 is displayed. There are three pages to complete:

File Specify the output and message file names, export file types such as DEL, WSF or IXF (each with their own options) and the select statement for the rows to be exported.

Large Objects Optional: Specify if LOBs from a table or view will be stored on separated files. If you want to store large objects (LOBs) in separate files (LOBSINFILE) specify LOB paths and LOB file names

Columns Optional: Specify new column names to be used in the output file instead of the table or view column names. This option can be used with IXF and WSF formats only.

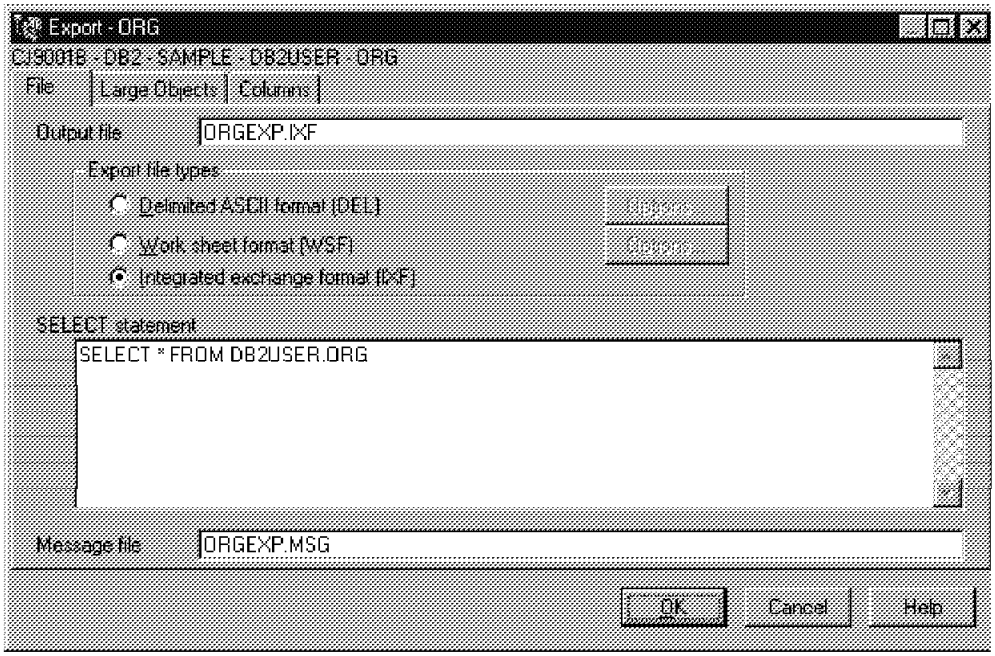


Figure 84. Export Notebook

When you have finished, select **OK** to begin the export process.

4.4.3.4 Load

You can use the Load utility to populate a table with data from an input file. You need SYSADM or DBADM authorization.

From the Control Center, click the mouse button two on the table you want to load data in and select **Load** from the pop-up menu.

The Load notebook shown in Figure 85 on page 168 is displayed. There are six pages to complete:

File — Specify the location of files, pipes or devices containing data and the type of data to be loaded, such as ASC, DEL or IXF (each with their own options).

Columns — Specify how to load the data columns into the table columns. Methods to include columns are by location (L), names (N) or position (P).

Counts — Specify the load mode and record counters, such as savecount (scope of records to set a point of consistency during the load phase), restart options, limit of rows to be loaded and warning counts.

Statistics — Optional: Specify how statistics are gathered for the table (and existing indexes) during load. This option cannot be used if Insert or Restart modes are selected.

Copy Options — Optional: Specify that this is a non-recoverable load operation and that a copy of changes made to the table during the load, will be saved and where this copy will be created.

Perform non-recoverable load operation option introduces the new NONRECOVERABLE parameter. If this parameter is used during a load, it will not be possible to recover the load transaction by a subsequent rollforward action. The rollforward utility will skip the transaction and will mark the table as *invalid*. It will also ignore any subsequent transactions. After rollforward is completed, this table can only be dropped.

Others Optional: Specify special details for the load process, such as where to write warning or error messages, temporary directories used during index creation, whether you want the table left in quiesced state after the load, exception table schema and name, and so on.

The degree of CPU parallelism option introduces the new PARALLELISM n parameter. It specifies the degree of parallelism to be used, that is, the number of processes or threads that the load utility will spawn to perform the operations that build table objects. This parameter is designed to exploit SMP parallelism.

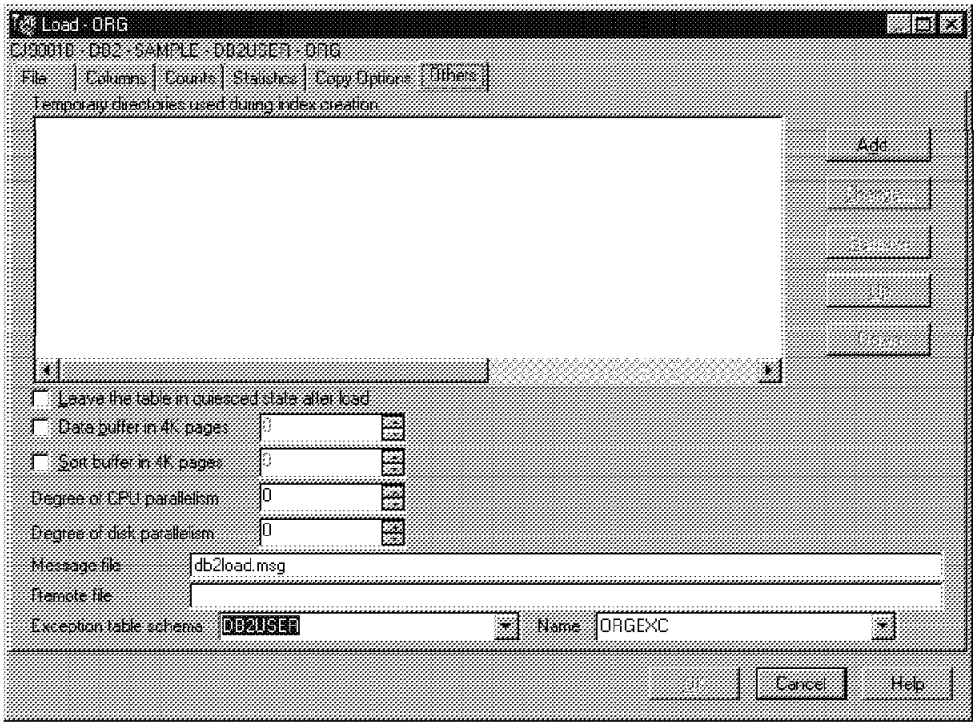


Figure 85. Load Notebook

When you have finished, select **OK** to begin the load utility.

After finishing the load process, if any check constraints are defined on the table, the table will be left in check pending state. To make the table available for use, you must activate these constraints. From the Control Center, click the mouse button two on the table you just finished to load data in and select **Set Constraints** from the pop-up menu.

The Set Constraints Notebook is displayed. Turn on the check constraints with deferred checking. With this option you must specify an exception table schema and table name. After running set constraints, the table is ready to be used.

4.4.4 Data Replication

Moving data via replication allows you to set up copy operations for relational data from one database to another. The actual copy is done from outside the administration tools, using two functions:

- **Capture**, which records changes made to data in source tables by reading the database log.
- **Apply**, which reads the changed data previously captured and stored in a change data table and applies it to the target tables.

When you perform your first replication action, such as defining a replication source, the Control Center will create the customized control tables.

4.4.4.1 Replication source tables

A replication source table is a table used as a source for copying data to a replication target table.

The replication source tables can be the following types of tables:

User Table	A table created outside of replication and used for an application. It is the table that gets the original update that triggers other replication.
Replication Target Table	Read-only target tables that have been redefined as sources for replication.
Consistent Change Data Table	A staging table, a target table type that has been redefined as a source table for staged replication.
External Table	An origin table created in a non-DB2 database and manually updated to match the CCD table structure. It allows non-DB2 table to be a source for replication.
Replica Table	An updatable target table that is used to replicate data back to the origin table or to other target tables.

When you define a table as a replication source, a Replication Source object is created. Additionally, an entry in the ASN.IBMSNAPREGISTER control table is added, identifying it as an available source table.

Finally, if you indicate that you want update copying, only updated rows are copied to the target table. The source table is checked for the Data Capture Changes option. If Data Capture Changes is not enabled, the original table is altered to enable this feature.

To define a replication source, from the Control Center, click the mouse button two on the table you want to define as source and select **Define as replication source**, then **Custom** from the pop-up menu.

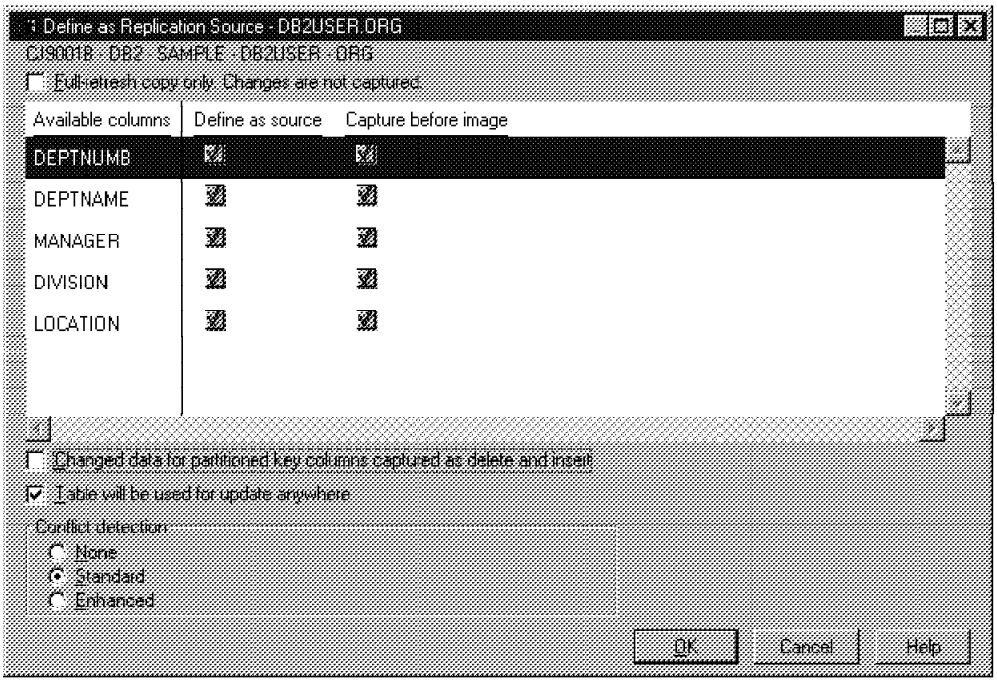


Figure 86. Defining a Replication Source

This allows you to define the table as a replication source. Figure 86 shows the panel where you define a customized source. After completing this panel you will be prompted to run the SQL or save a script to be run later (the same panel shown on Figure 87 is presented if you select the **Quick** option from the pop-up menu).

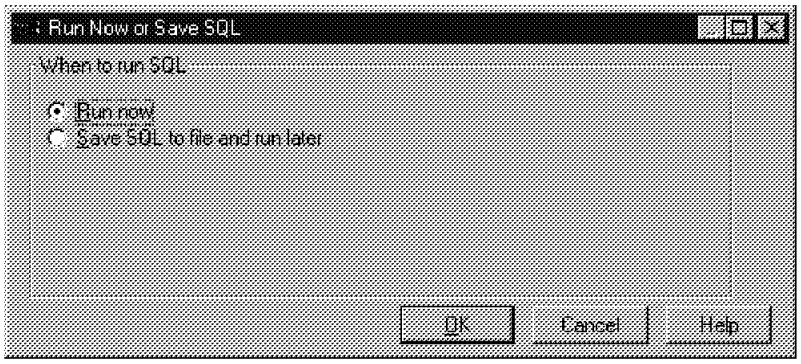


Figure 87. When to Run SQL

If you choose to run the file now, an object is created in the Replication Sources object folder. This object is now available as replication source and can be included in a subscription set.

4.4.4.2 Subscriptions

A subscription defines the structure of a target table, and how the Apply program replicates data to that target table: when, how often, which enhancements, all data or changed data, and so on. It also defines the type of target table to be created, such as a user copy table or a replica.

Subscriptions sets are the grouping of one or more subscriptions that are somehow related; for example, they have RI constraints. For these tables, the changed data needs to be copied to the target tables at the same time, or in the same copy interval.

To define a subscription set, from the Control Center, select the **Replication Sources** folder, then click the mouse button two on the replication source you want to define as source for the subscription set and select **Subscribe** from the pop-up menu. The Define Subscription window opens as shown in Figure 88.

In the Define Subscription window fill in the fields for subscription name, target server and apply qualifier. Modify the target table name and specify whether to create the target table or not.

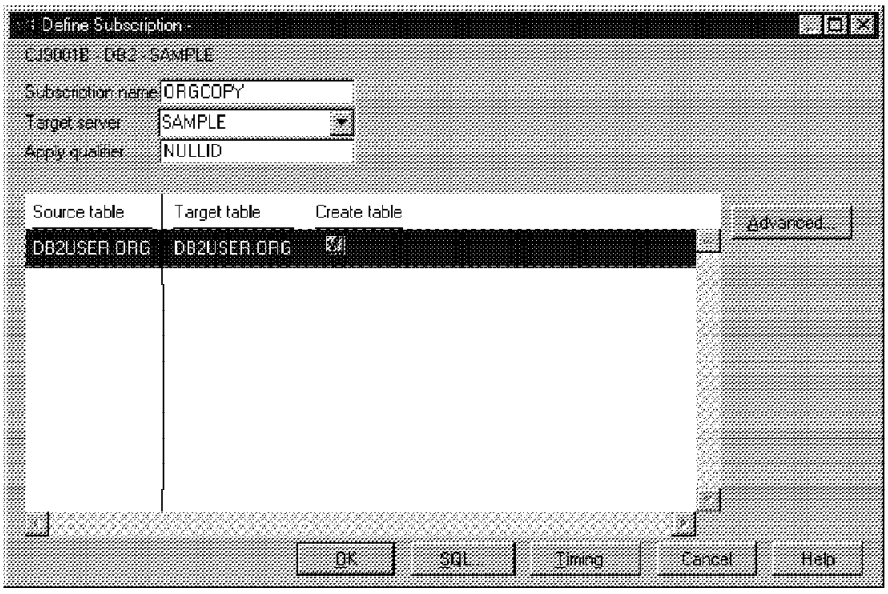


Figure 88. Defining a Subscription

Click on the **Timing** push button to specify the start date, time and frequency of replications based on time or events. You will be able to fill in the fields in the **Source to Target** tab if you are defining replication from source to read-only or replica target as shown in Figure 88, or in the **Replica to Source** tab if you are defining replication from replica to source. In the **Data Blocking** tab specify the number of minutes at a time Apply will copy committed data.

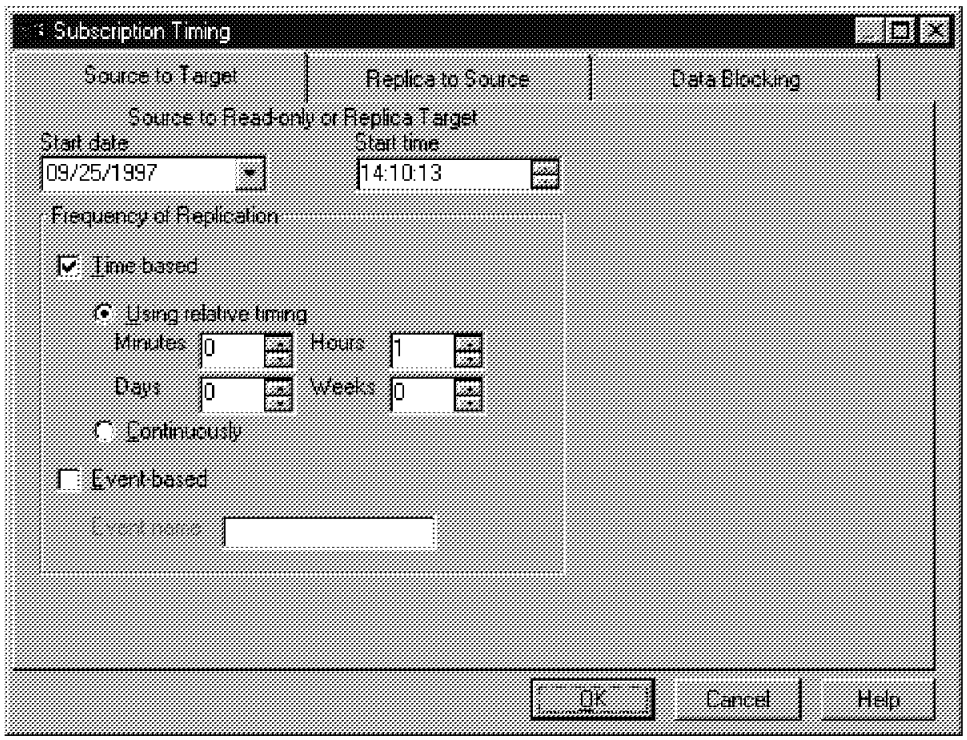


Figure 89. Defining Subscription Timing

Click on the **SQL** push button to add SQL statements or stored procedures to be executed before or after the subscription is processed. You can define acceptable SQLSTATEs for each statement or procedure.

Once defined, the subscription definition will appear in the contents pane for **Replication Subscription** object.

4.4.5 Recovering Your Database

4.4.5.1 Backup

The concept of a database backup is taking a copy of the data and storing it on a different media in case of failure or damage to the original. If the database needs to be restored to a point beyond the last full backup, the database logs are required to roll the data forward to the point of failure.

- **Backup Database Notebook**

The Backup Database notebook prompts you to fill in some information about the destination of the backup image and other backup options. From the Control Center, click the mouse button two on the database you want to backup and select **Backup**, then **Database** from the pop-up menu.

In the Backup Database notebook there are two pages to complete:

Backup Specify the backup destination, such as media type and list of directories or tapes. Also specify information about the estimated backup image size.

Options Specify the performance parameters to be used in the backup utility, and whether it is taken on-line or off-line.

You may run the backup immediately, schedule it to be run one or more times or save a script to be run later as shown in Figure 90 on page 173.

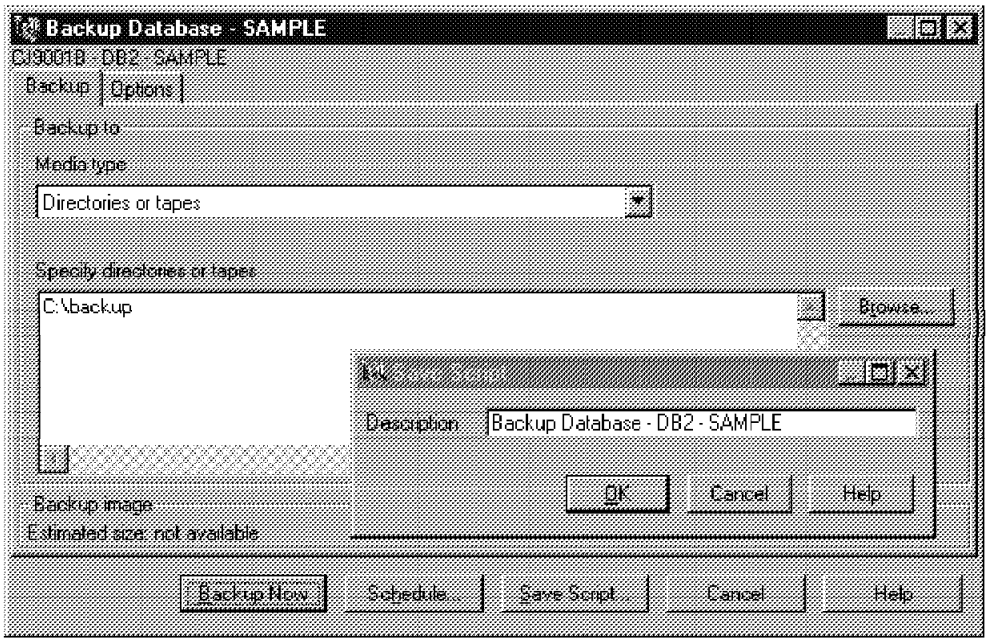


Figure 90. Backup Database Notebook

- **Backup Database SmartGuide**

The Backup Database SmartGuide helps you to design a backup strategy for the database based on your business requirements.

From the Control Center, click the mouse button two on the database you want to backup and select **Backup**, then **Database using SmartGuide** from the pop-up menu.

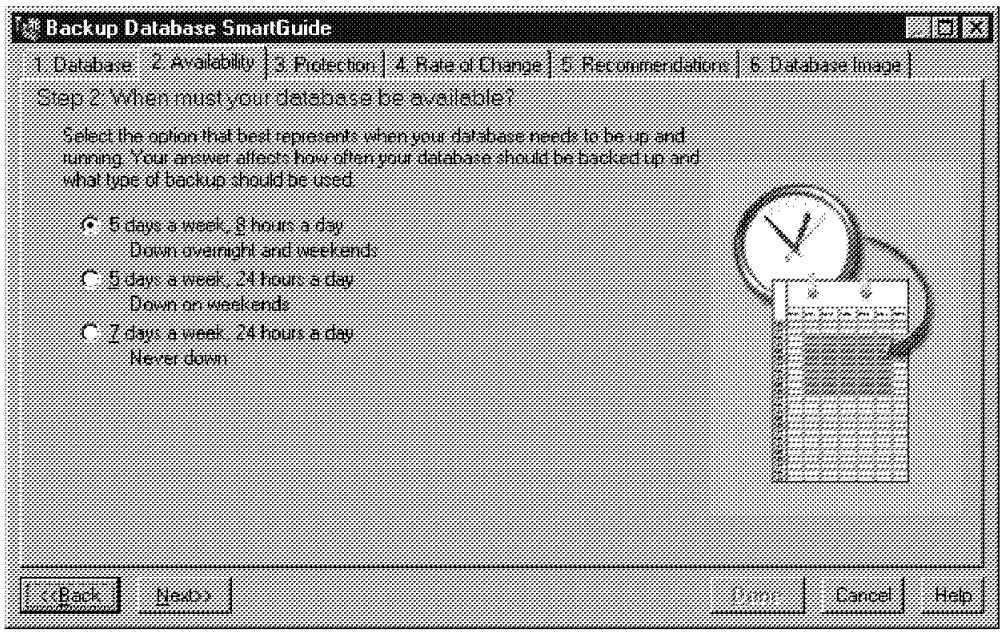


Figure 91. Backup Database SmartGuide

The Backup Database SmartGuide is shown in Figure 91. There are six pages to complete:

Database	Select the database to backup.
Availability	Determine how often the database should be backed up based on its availability requirements.
Protection	Select the appropriate level of protection for the data.
Rate of change	Estimate how much of your data is new or changes each day.
Recommendations	Some options are available based on the previous choices you made, such as type of logging, type of backup and schedule. You can change any of them if you consider it is necessary, for example, the schedule as shown in Figure 92 on page 175.
Database image	Specify where to store the database backup files.

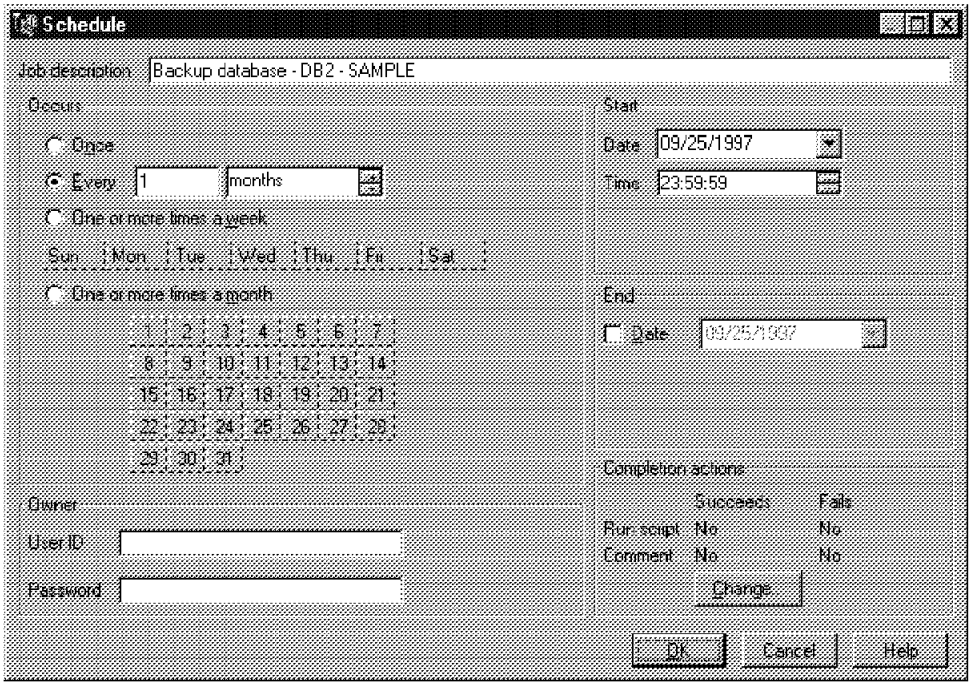


Figure 92. Scheduling a Database Backup

4.4.5.2 Restore

The restore process can be used only if the database has been previously backed up with the BACKUP command. To restore to an existing database from a full database backup, you need SYSADM. SYSCTRL or SYSMAINT authority. To restore to a new database, you must have SYSADM or SYSCTRL authority.

- **Restore Database Notebook**

The Restore Database notebook prompts you through the process of recovering a database from a disk failure. From the Control Center, click the mouse button two on the database you want to recover and select **Restore**, then **Database** from the pop-up menu.

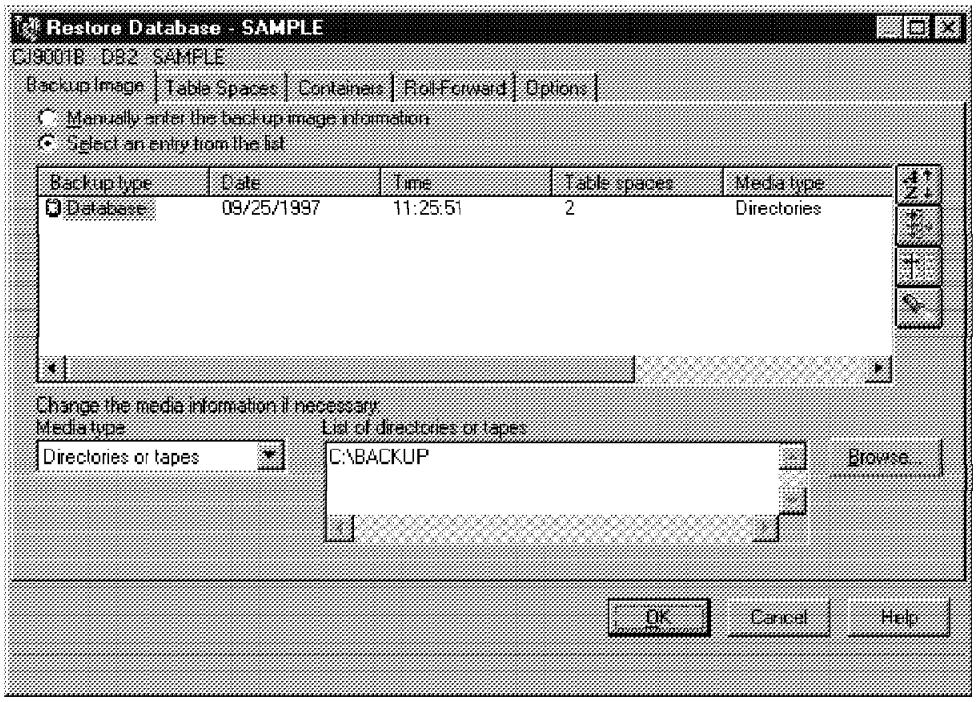


Figure 93. Restore Database Notebook

In the restore database notebook shown in Figure 93, fill in the fields to specify the backup image, whether to redefine the table spaces and containers or not, considerations for roll-forward and parameter options. Click **OK** to begin the restore process.

- **Restore Database SmartGuide**

The Restore Database SmartGuide guides you through the process of recovering a database. From the Control Center, click the mouse button two on the database you want to restore and select **Restore**, then **Database using SmartGuide** from the pop-up menu.

The Restore Database SmartGuide is shown in Figure 94 on page 177. There are 4 pages to complete:

Restore status Verify the status of your database before beginning the restore. The level of possible recovery depends on the backup plan you created before.

Backup history This will show you a list of backup images available to restore from.

Roll-forward Specify whether to reapply the changes to the database since the last backup (if your database is enabled to do so) or to leave the roll-forward process to be run later.

Log files Specify the current location of the log files.

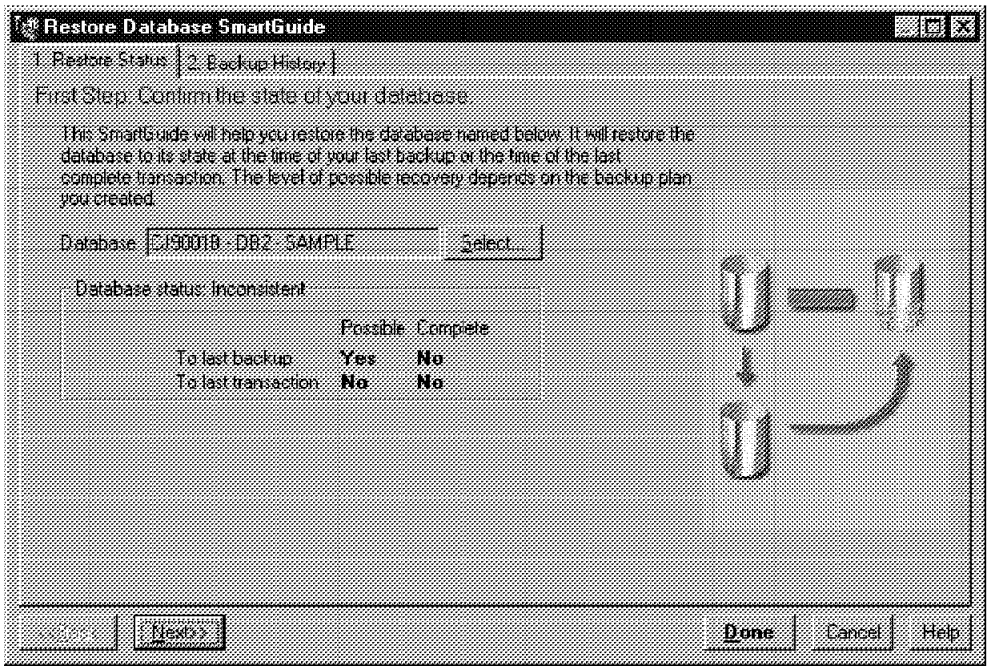


Figure 94. Restore Database SmartGuide

- **Restore Database to New Notebook**

The Restore Database to new notebook guides you through the process of recovering a database backup to a new database. From the Control Center, click the mouse button two on the database you want to recover and select **Restore to new** from the pop-up menu.

The Restore Database to New window prompts you for the same information that is required for restore database to an existing one, except that you must specify the new database name and destination drive where it will be created.

- **Roll-forward**

While restore and roll-forward are independent operations, restore is the first phase of a complete roll-forward recovery of a database. After a successful restore, a database that was configured for roll-forward recovery enters in a roll-forward pending state, and it is not usable until the ROLLFORWARD process has been run successfully.

After the restore database process finished successfully, from the Control Center click the mouse button two on the database you restored and select **Rollforward**. This will begin the application of logs on the database.

4.4.5.3 Table Space Level Backup, Restore and Roll-forward

A table space level backup contains one or more table spaces within a database. A table space level backup can be taken on-line or off-line.

From the Control Center click the mouse button two on the table space you want to backup and select **Backup** from the pop-up menu. In the **Backup Table Space** window fill in the information required to begin the table space level backup as shown in Figure 95 on page 178.

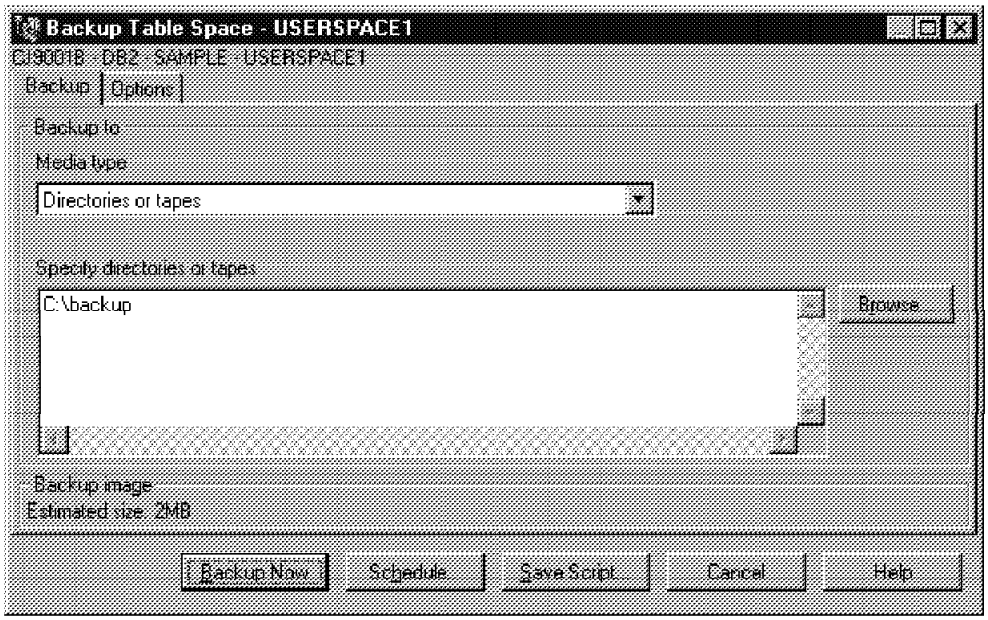


Figure 95. Backing Up a Table Space

The table space level backup can be used to recover from problems that only affect specific table spaces. While this recovery is taking place, all other table spaces are available for processing.

To restore a table space, from the Control Center click the mouse button two on the table space you want to restore and select **Restore** from the pop-up menu. In the **Restore Table Space** window select the backup image you will use, and specify other considerations for the restore process as shown in Figure 96 on page 179.

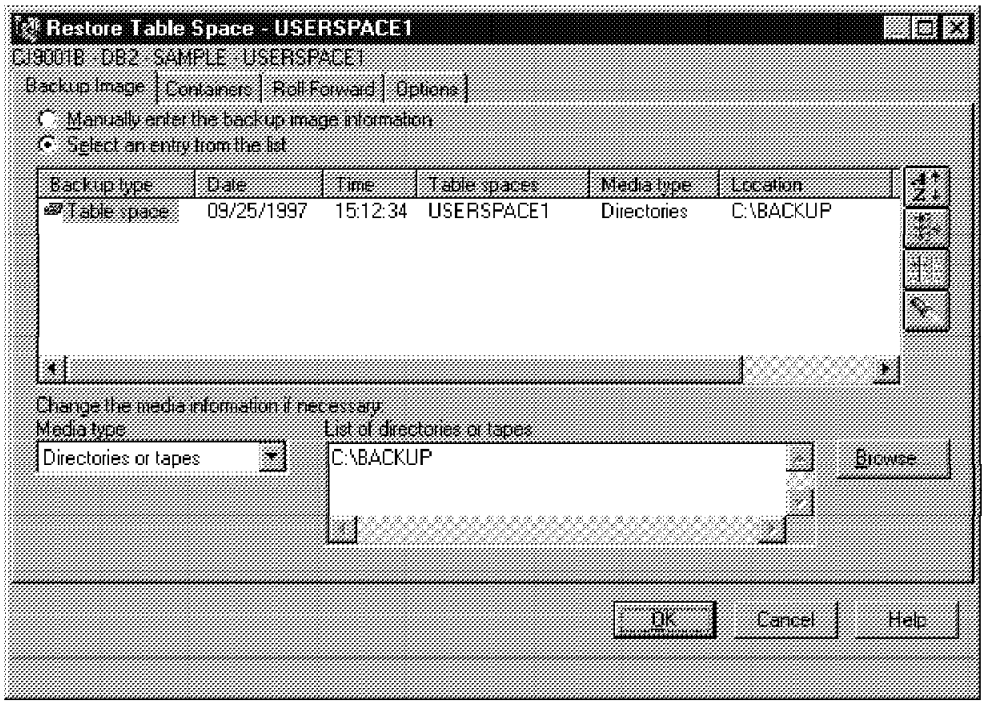


Figure 96. Restoring a Table Space

To ensure that the restored table spaces are synchronized with the rest of the database, the table spaces must be rolled forward to the end of logs. For this reason, table space level backup and restore can only be performed if roll-forward recovery is enabled.

To roll-forward a table space, from the Control Center click the mouse button two on the table space which is in Roll-forward pending state and select **Roll-forward**. Complete the information as shown in Figure 97 and click **OK** to roll-forward the table space.

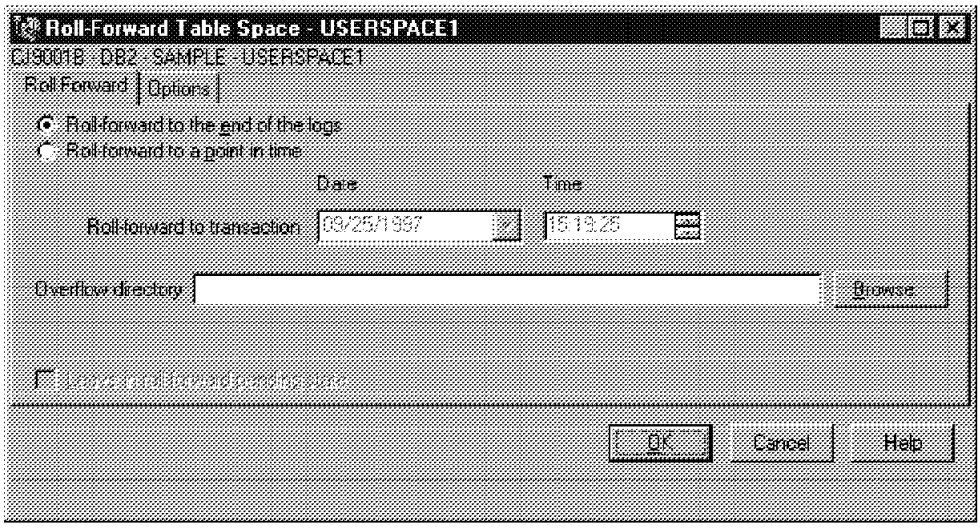


Figure 97. Rolling Forward a Table Space

Each component of a table (for example, data and index) may be backed up and restored with the table space in which it resides, independently of the other components of the table. Table space level backup and restore cannot be run concurrently.

4.4.6 Securing Your Database

DB2 manages access to its databases and database objects using administrative authorities and privileges:

- A **privilege** is the right to access a specific database object in a specific way, for example, being able to connect to a database.
- An **authority** is a higher level way of grouping privileges and other database operations, for example being able to perform system maintenance operations.

DB2 UDB also introduces the definition of a schema, a named collection of objects, that provides a logical classification of those objects. Each schema can be explicitly created and has an owner who can control who puts objects into this schema.

For compatibility with previous versions, an IMPLICIT_SCHEMA authority allows the user to automatically create a schema just by creating an object. If the new object schema does not exist, a new schema is created.

4.4.6.1 Database Authorities

- System Administration Authority - SYSADM - is the highest level of administrative authority. Users with SYSADM authority can run utilities, issue database and database manager commands, and access the data in any table in any database within the instance.
- System Control Authority - SYSCTRL - is the highest level of system control authority. Users with SYSCTRL authority can perform maintenance and utility operations against the instance and its databases. These operations can affect system resources but they do not allow direct access to data in the databases.
- System Maintenance Authority - SYSMAINT - is the second level of system control authority. Users with SYSMAINT authority can perform maintenance and utility operations against the instance and its databases. As SYSCTRL, users can affect system resources but not the data in the databases.
- Database Administration Authority - DBADM - is the second level of administrative authority and applies only to a specific database. Users with DBADM authority can run utilities, issue database commands and access data in any table in the database.
- Implicit Schema Authority - IMPLICIT_SCHEMA - is the default when a database is created or migrated from a previous release. Users with IMPLICIT_SCHEMA authority can create objects in any schema that does not already exist without being the owner of that schema object.

By default, SYSADM authority is granted to:

- Usernames that are members of the administrator's group in the Windows NT User Manager or OS/2 User Profile Manager.

- Any user that belongs to the group specified by the SYSADM_GROUP parameter in the database manager configuration file for a particular instance.

You can change the authorities for each database manager instance by changing the SYSADM_GROUP, SYSCTRL_GROUP and SYSMAINT_GROUP parameters. But before you do so, ensure that:

- The group exists. Use the Windows NT User Manager Administrative Tool, OS/2 User Profile Manager or AIX smit to create groups.
- The users you want to control this instance are also members of that group.

Defining these privileges has the benefit of allowing someone to be DB2 administrator without being a system administrator.

To update the database manager configuration file, open the Client Configuration Assistant and click on the **Client Settings** push button. On the Administration tab enter the name of an existing group that you want to assign each authority to as shown in Figure 98 and then click **OK**.

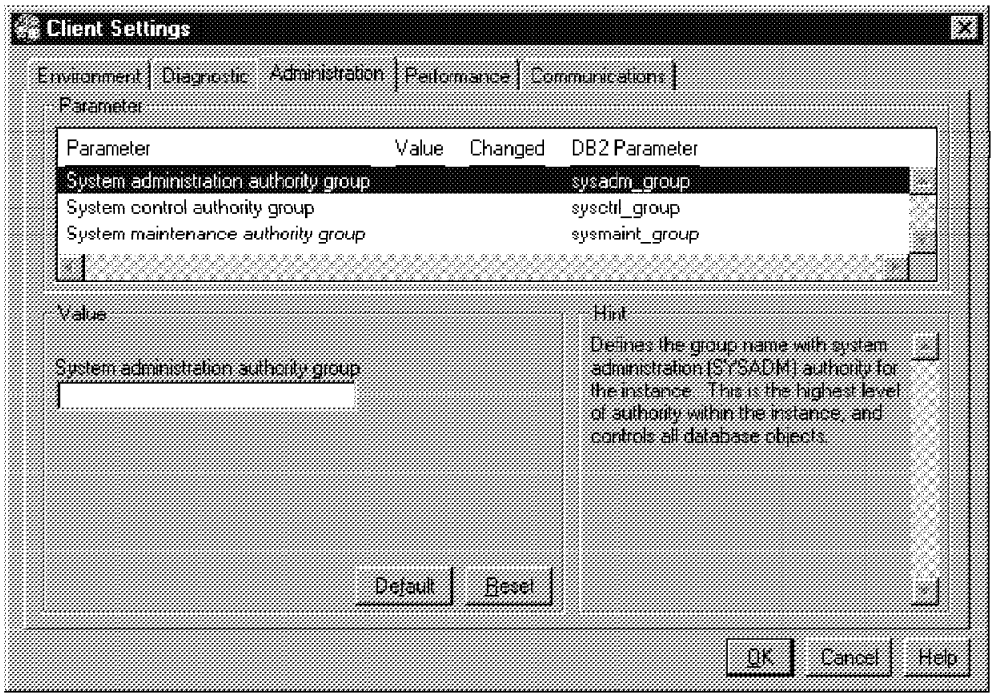


Figure 98. Updating Database Manager Configuration Parameters for Administration

4.4.6.2 Database Privileges

When a database is created, the following privileges are automatically granted to anyone who accesses the database:

Connect Allows a user to access the database.

Createtab Allows a user to create new tables in the database.

Bindadd Allows a user to create new packages in the database.

Createin Allows a user to create objects in a schema.

Alterin Allows a user to alter objects definition in a schema.

Dropin Allows a user to drop objects from within a schema.

Select Allows a user to use select on system catalog views.

Other privileges at table, view, package, schema and index level must be granted by the appropriate authorization ID.

The privileges that can be granted and revoked at table and view level are:

Control Provides all the privileges for a table or view (drop, grant and revoke).

Alter Allows a user to add columns to a table, to add or change comments in columns, create or drop a primary key or check constraints.

Delete Allows deletion of rows on a table.

Index Allows index creation on a table.

Insert Allows a user to insert an entry into a table or view, and to run the IMPORT utility.

References Allows a user to create a referential constraint on a table.

Select Allows a user to retrieve rows from a table or view, create a view on a table, and to run the EXPORT utility.

Update Allows a user to change an entry in a table or view.

The privileges that can be granted and revoked at package level are:

Control Provides user with the BIND and EXECUTE privileges.

Bind Allows a user to rebind an existing package.

Execute Allows a user to process an existing package.

The privileges that can be granted and revoked at schema level are:

Createin Allows a user to create objects within the schema.

Alterin Allows a user to alter objects within the schema.

Dropin Allows a user to drop objects from within the schema.

The privilege at index level is one:

Control The creator of an index receives this privilege on the index.

You can grant and revoke privileges for users and groups from the administration tools for databases, schemas, packages, tables, views and indexes. Figure 99 on page 183 shows the privileges granted to PUBLIC at database creation time.

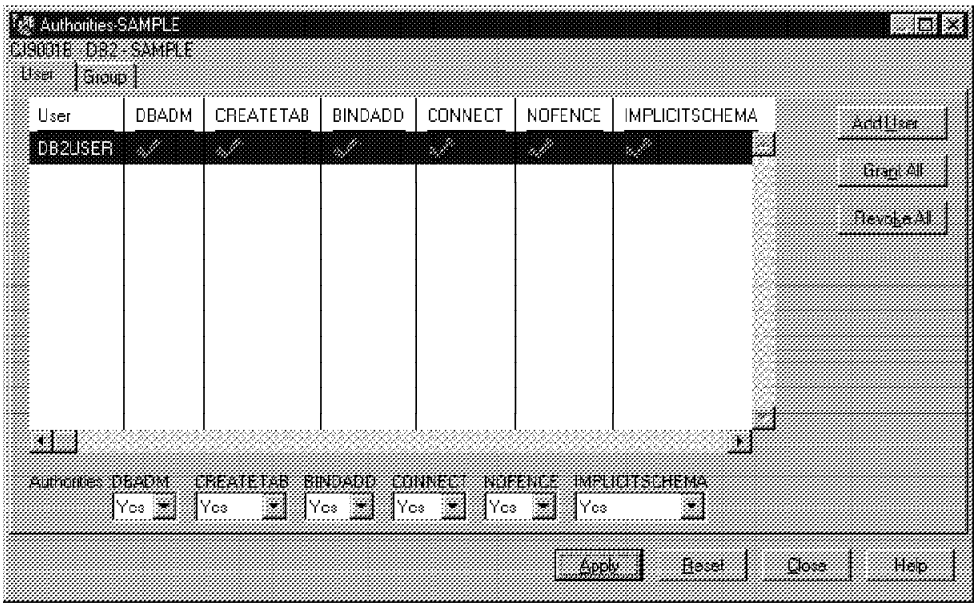


Figure 99. Granting and Revoking Authorities

From the Control Center click the mouse button two on the object you want to grant or revoke privileges, and select **Privileges** from the pop-up menu.

The privileges window opens and you can select the **User** page to work with user privileges or select the **Group** page to work with group privileges. Along the bottom of the window, select **Yes**, **No** or **Grant** for each individual privilege. When you have finished, click on **Apply** and close the window.

4.4.7 Tuning Your Database

To facilitate the tuning of database performance, DB2 UDB V5 introduces the **SmartGuide for Performance Configuration**.

By extracting static information from the system and by asking a small number of questions about the database workaday, this tool will run a series of calculations designed to determine an appropriate set of customized values for selected database and database manager configuration variables.

Some considerations on running this tool:

- It needs at least 16 MB of memory on the database server.
- It will do tuning only for DML statements.
- It cannot be used for tuning MPP systems.

From the Control Center, click the mouse button two on the database you want to run the performance tuning on and select **Configure Performance** from the pop-up menu.

When using this tool, it is recommended that each instance of the database manager has only one production database. If you have more than one active production database at the same time, move one to a new instance before running this tool.

The Performance Configuration SmartGuide prompts you to answer some questions related to:

- The amount of server memory your database can use.
- The type of workload of your database, choosing between queries (for example, decision support) or transactions (for example, order entry) or mixed.
- The typical database transaction, estimating the number of transactions per minute and also the number of SQL statements in a single unit of work.
- The database administration priority for recovering after disk drive failure.
- Whether the database is already populated. It is recommended to run this tool each time the size of the database increases or decreases significantly.
- The number of applications connected at the same time. This is used to allocate enough connections, by taking into consideration the number of users waiting for a connection and the available memory resources.
- The isolation level that best reflects your applications needs.

Finally, you will be provided with a list of recommended parameters values as seen in Figure 100. Applying these values will in most cases lead to better performance. This tool should be used as a starting point upon which further adjustments can be made to obtain optimized performance.

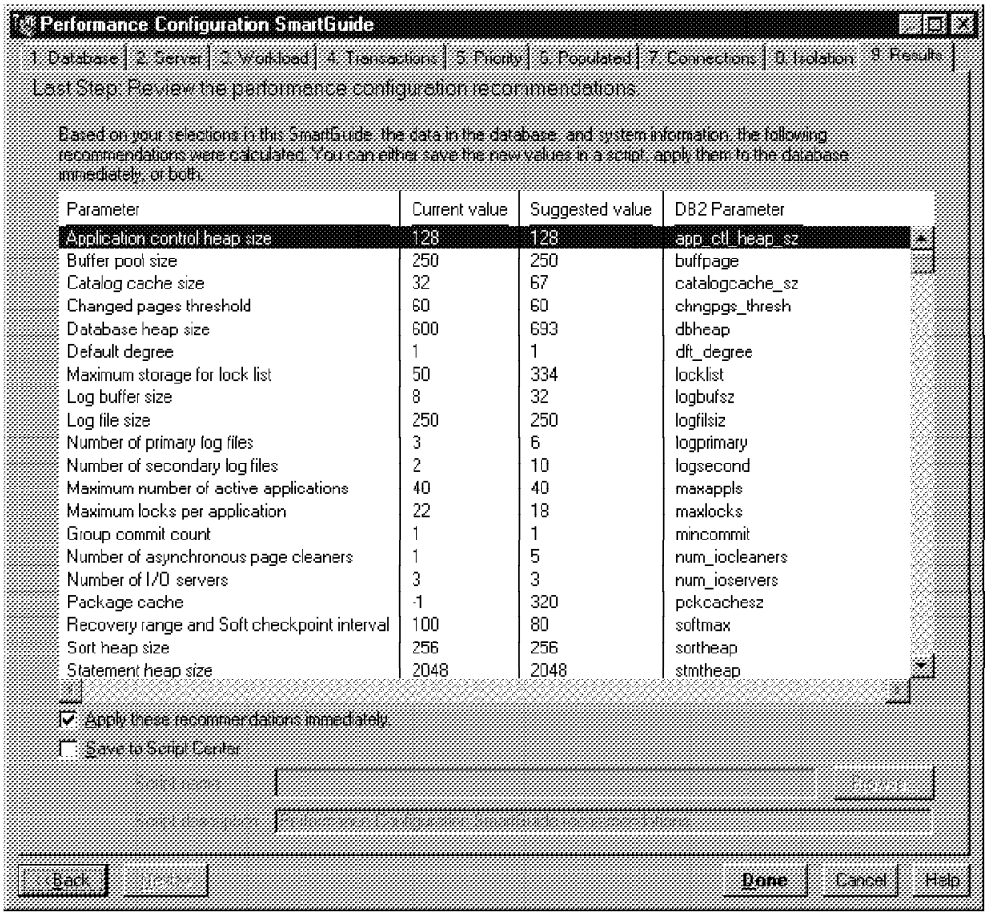


Figure 100. Performance Configuration SmartGuide - Results

The parameter value changes recommended can be applied immediately or saved on a script file to be run later. Bear in mind that the database administrator is responsible for stopping and starting the database to make these changes effective. It is also recommended to rebind all packages after changing database parameters.

This tool can be invoked at any time to adjust the configuration parameters, for example:

- Before a change in estimated workload;
- After adding some additional memory to the server;
- Each time database size changes significantly.

4.4.8 Miscellaneous Tasks

4.4.8.1 Creating a Schema

To create a schema, from the Control Center click the mouse button two on **Schemas** icon under the database you want to create the schema in and select **Create** from the pop-up menu.

In the Create Schema window complete the fields with the schema name, authorization name (owner of this new schema) and a comment. Click **OK** to create the schema and it will appear listed on the contents pane.

4.4.8.2 Creating User Defined Types

The User-defined Types (UDTs) let you define the distinct data types you need for the applications you want to build, extending the capabilities of the built-in data types.

To create a UDT, from the Control Center click on the plus sign (+) at the Application Objects icon, then click the mouse button two on the **User Defined** icon and select **Create** from the pop-up menu.

A panel prompts you to fill in some information to define the new type, such as the schema and type name, source data type, some characteristics of the data type, whether comparison operators will be generated or not and comments. After completing the panel, click **OK** and the data type will be created and listed on the contents pane.

4.4.8.3 Creating Triggers

A trigger lets you define a set of actions that are executed at, or triggered by a delete, insert or update operation on a specified table. You can use triggers in several ways:

- To check or modify values before they are actually updated or inserted in the database.
- To update data in other tables, for maintaining relationships between data or keeping audit trail information.
- To check against other data in the table or in other tables, for example, to ensure data integrity when referential integrity are not appropriate.
- To run other non-database operations coded in user-defined functions, such as issuing alerts or updating information outside the database.

From the Control Center, click the mouse button two on the **Triggers** icon and select **Create** from the pop-up menu.

The **Create Trigger** notebook will request information on two pages:

Trigger Specify the trigger schema and name, related table schema and name, time to trigger action (before or after), the operation that causes the trigger to be executed (insert, delete or update of listed columns) and an optional comment.

Triggered action Specify details of the actions to be taken at trigger execution time, such as correlation name for old and new rows, temporary tables for old and new rows, scope of execution (for each row or statement), and triggered action, as a block of SQL statements.

Then, you may **Apply** the definition to create the trigger, **Reset** the information to define a new trigger or **Close** the notebook discarding all the changes.

4.4.8.4 Reorganizing tables

The REORG utility is used to rearrange data of a table into a physical sequence according to a specified index and removing the free space inherent in fragmented data. This can provide faster access to the data and improve performance.

From the Control Center, click the mouse button two on the table you want to reorganize and select **Reorganize** from the pop-up menu.

In the reorganize window you can optionally specify to use a temporary table space and the index to be used to order in the reorganized table. Click on **Reorganize Now** to immediately reorganize the table.

4.4.8.5 Updating Statistics

The statistics describe the physical characteristics of a table and its indexes. The SQL optimizer uses these statistics to generate the access plan.

You should update statistics:

- When a table has been loaded with data and indexes have been created.
- When table data has been reorganized.
- When there have been extensive updates, deletes and inserts that affect the table and its indexes.
- Before binding application programs whose performance is critical.

From the Control Center, click the mouse button two on the table you want to update the statistics on and select **Run Statistics** from the pop-up menu.

In the Run Statistics window specify the level of statistics you want to gather for the table and its indexes, the type of access you want other users to have to the table while statistics are being gathered and click **OK** to begin the statistics collecting process.

You should rebind application programs which reference tables for which statistics have been updated, because the SQL optimizer may choose a different access plan with the new statistics.

4.4.8.6 Creating and scheduling script files

Many of the functions available from the Control Center can be performed using corresponding commands and APIs. This section will show how the Command Center, Script Center and Journal can be used together.

From the Control Center toolbar, click on the **Command Center** icon. In the script page enter DB2 commands, then from the **Script** menu select **Save as** and complete the fields required, ensuring to select **To Script Center**. The command script will be saved and then be accessible through the Script Center.

From the Control Center toolbar, click on the **Script Center** icon. A list of script defined will be available. Click with the mouse button two on the script you just created and saved in the Command Center and select **Schedule** from the pop-up menu.

In the Schedule window fill in the job description, select the frequency of the job and a completion action, such as a message or scripts to be run on success or failure of the job. Click **OK** to save.

From the Control Center toolbar, click on the **Journal** icon. Click on the **Pending Jobs** tab to see the job you created.

4.4.8.7 Forcing off applications

There may be occasions when you need to force local or remote users or applications off the system for maintenance on a server. If an operation that cannot be interrupted is forced, the operation must be successfully re-executed before the database becomes available.

To force users off a remote server, it is first necessary to attach to that server. From the Control Center click the mouse button two on the instance server you want to work with and select **Attach**. Enter a valid user id and password with administrative authorization.

Now, from the Control Center click the mouse button 2 on the instance server you are attached and select **Force applications**. In this window you can select to force all applications or specify to force an agent ID from a list of applications running on this instance.

4.5 Visual Explain and Performance Monitor Overview

This section will explain how to monitor database activity, using tools such as Visual Explain and Performance Monitor. Based on the information provided by these tools, you can decide which actions to take to tune the SQL statements used in your applications or the database environment.

4.5.1 Visual Explain

DB2 Visual Explain is an easy-to-use, graphical tool for analyzing and tuning SQL statements.

It helps Database Administrators and Application Developers to:

- View the access plan chosen by the database manager′s optimizer for a given SQL statement.
- Design application programs and databases.

- View all the details of the access plan, including the statistics in the system catalogs.
- Decide whether or not to add an index to a table.
- Identify the source of problems in SQL statements and database transactions.
- Tune SQL statements for better performance
- Use the portable snapshot function to view snapshots from a remote DB2 server.
- Display access plans for partitioned and SMP systems.
- Model the impact of environment changes on SQL statements

Using IBM industry-leading query optimization technology, DB2 Visual Explain provides a wealth of information about the access plan chosen by the database optimizer, and presents this information in an easy-to-use, intuitive manner using a graphical display as shown in Figure 102 on page 189. .*

4.5.1.1 Dynamic Explain

From the Control Center, click the mouse button 2 on the database you want to run the explain utility on and select **Explain SQL** from the pop-up menu.

In the Explain SQL Statement panel, fill in the SQL text you want to analyze and the parameters to be considered by the optimizer as shown in Figure 101 and click **OK**.

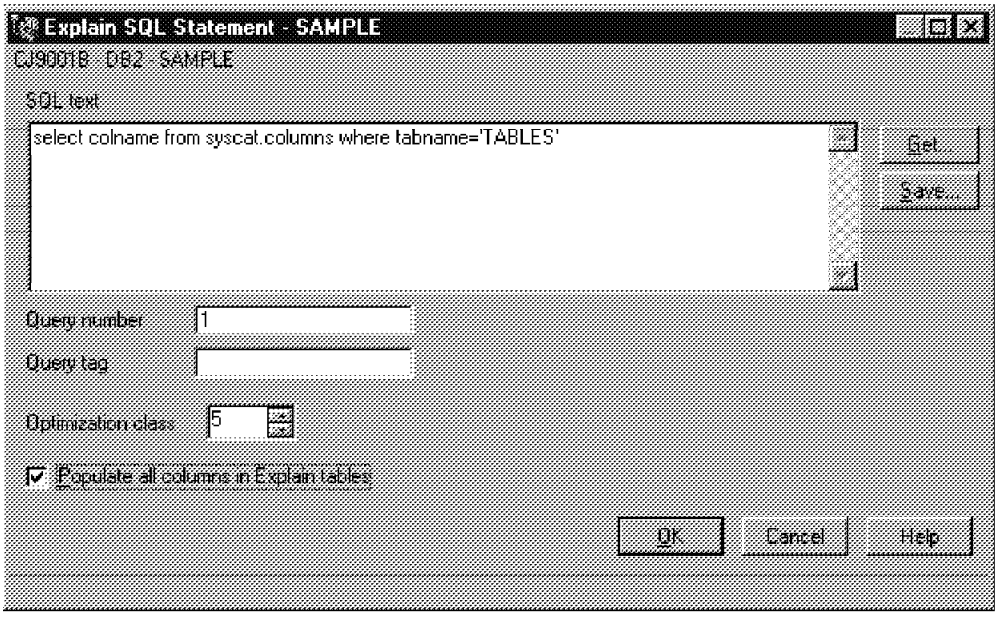


Figure 101. Explaining an SQL Statement

The Access Plan Graph window shown in Figure 102 on page 189 is displayed.

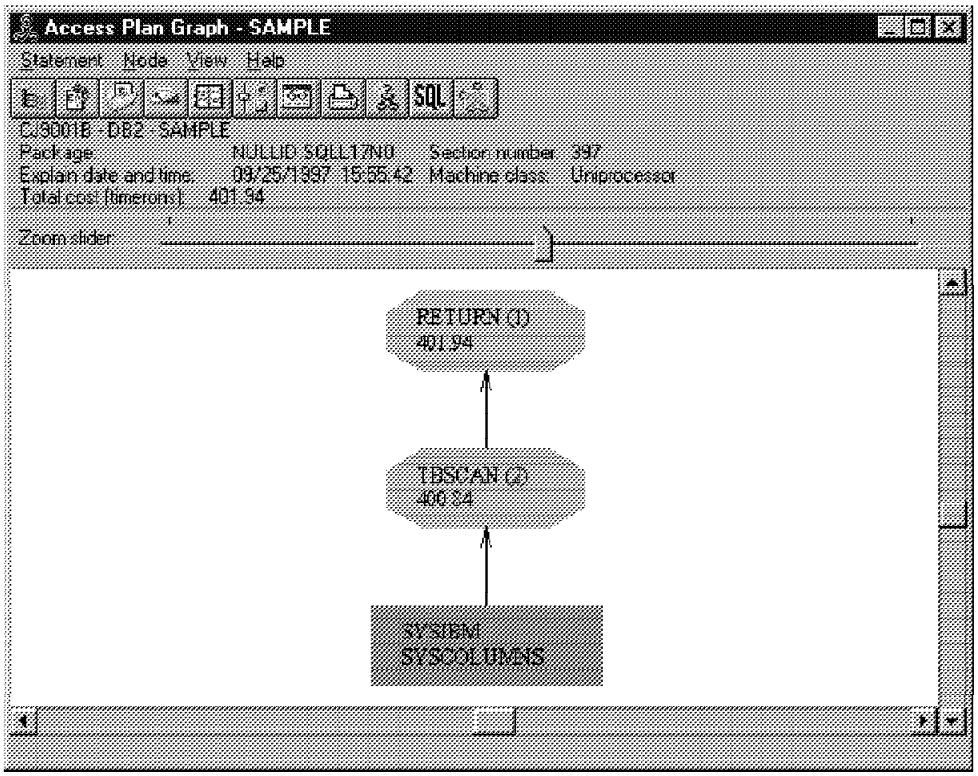


Figure 102. Graphical Representation of an SQL Statement Access Plan

4.5.1.2 Explaining a Package

From the Control Center, select **Packages** then click the mouse button two on the package you want to explain and select **Show explainable statements** from the pop-up menu.

A panel is displayed listing the SQL statements contained in this package. You can choose one statement, click the mouse button two and select **Explain SQL** from the pop-up menu as shown in Figure 103 on page 190. Fill in the parameters needed and click **OK** to see the Access Plan Graph.

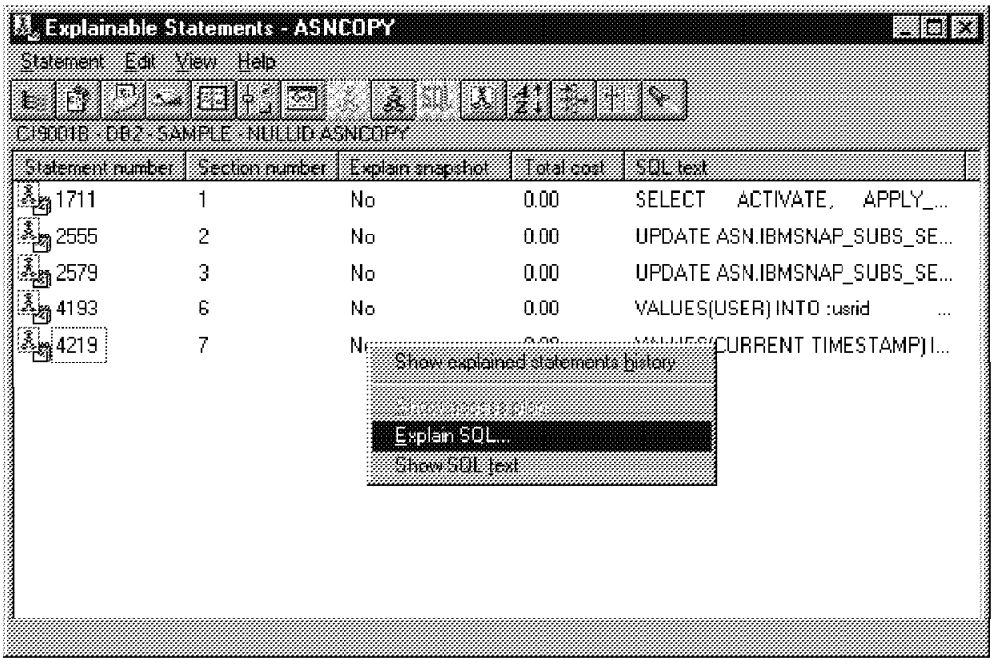


Figure 103. Explaining a Package

4.5.2 Monitoring Performance

The administration tools provide you with comprehensive performance monitoring of your DB2 system, including performance data collection, viewing, reporting, analysis and alerting. These tools can be used for tuning purposes and to provide an early warning of potential problems, or to automate actions to correct problems.

The information can be gathered by the database system monitor in one of two ways:

- Analyzing an event at a point in time (monitoring objects with the Snapshot Monitor).
- Analyzing an event for a period of time (creating an event monitor and analyzing events with the Event Analyzer).

You should use the Snapshot Monitor when you have a problem that is occurring at a single point in time, whereas you use the Event Analyzer when you need to know summary information, such as how long a transaction took or how much CPU a statement is using.

4.5.2.1 Using the Snapshot Monitor

The Snapshot Monitor is the tool that lets you:

- Graph performance information for a point in time.
- Define performance variables.
- Set the capture frequency of performance snapshots.
- View the results of performance calculations.
- Define threshold values and threshold actions.
- Generate and store alerts.

The performance monitor allows you to define exception conditions by specifying threshold values. When a threshold value is reached, you can predefine any combination of the following actions:

- Notification through a window or audible alarm.
- Logging a record in a database.
- Execution of a script or program.

Taking a snapshot at predefined intervals gives you the information for a specific point in time. This allows you to view snapshot data in real time either as graphs or textual views in both detail and summary form, for the current activity in the database manager and its database applications.

You can only have one Snapshot Monitor capturing snapshots from an instance of a database manager. This means that the API used to get the snapshot information is issued only once for all monitored database objects in the database manager.

From the Control Center, click the mouse button two on the database you want to monitor and select **Snapshot Monitoring**, then **Show monitor profile** from the pop-up menu.

The Monitor Profile notebook in Figure 104 opens. You may accept the default profile or specify new values for performance and thresholds. Note that a profile is associated with an instance only, not a specific database.

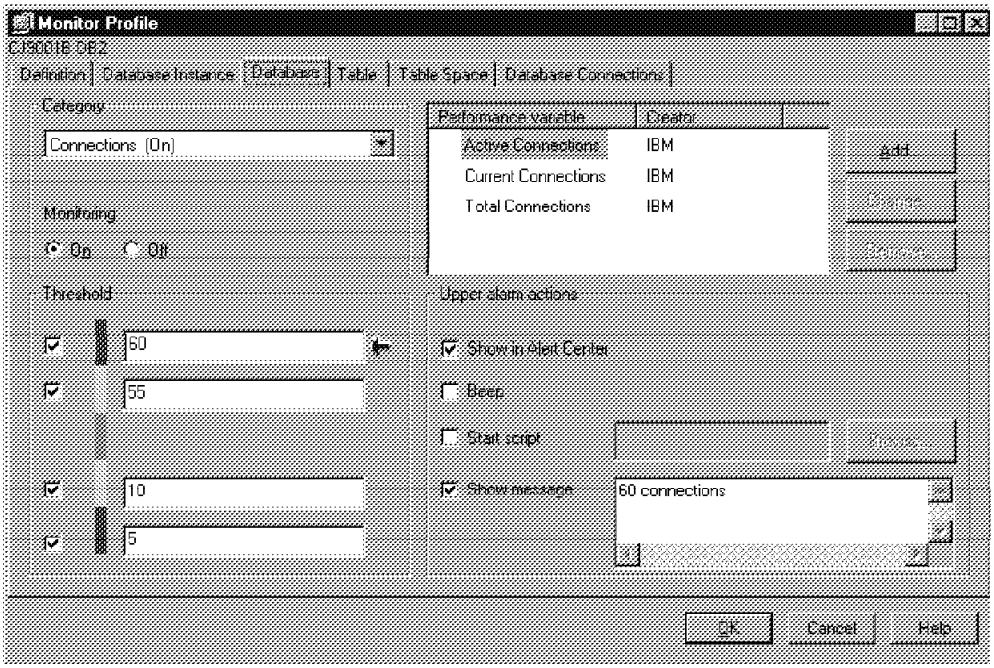


Figure 104. Performance Monitor Profile

The information collected by the snapshot monitor is grouped into the following categories:

- Status information is available at the database manager (instance), database, table, and table space levels. It contains counters, status indicators, and other data specific to each level.

- Application-level information includes data on unit of work, lock status, and numerous counters. You can also get information on the application′s current SQL statement.
- Locking details, such as lock waits, can help determine who is holding a lock when another user is waiting.
- Status information on Distributed Database Connection Services (DDCS) applications is available if you are using a DDCS gateway to access DRDA hosts from client applications.

To start snapshot monitoring, from the Control Center, click the mouse button 2 on the database you want to monitor and select **Snapshot Monitoring**, then **Start Monitoring** from the pop-up menu

To display the snapshot data in textual format, from the Control Center, click the mouse button two on the database you want to monitor and select **Snapshot Monitoring**, then **Show Monitor Details** from the pop-up menu. The Performance Details window is displayed as shown in Figure 105.

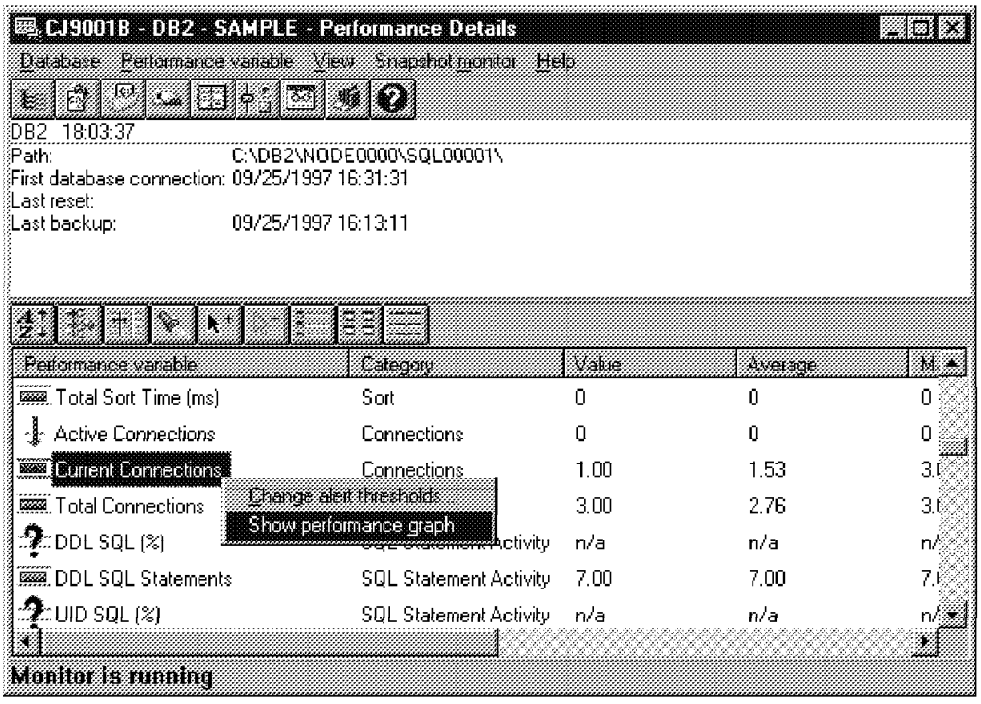


Figure 105. Performance Monitor - Performance Details

To display the snapshot data in graphical format for a given performance variable, from the Performance Details window, click the mouse button two on the performance variable and select **Show Performance Graph** from the pop-up menu. The Performance Graph window is displayed as shown in Figure 106 on page 193.

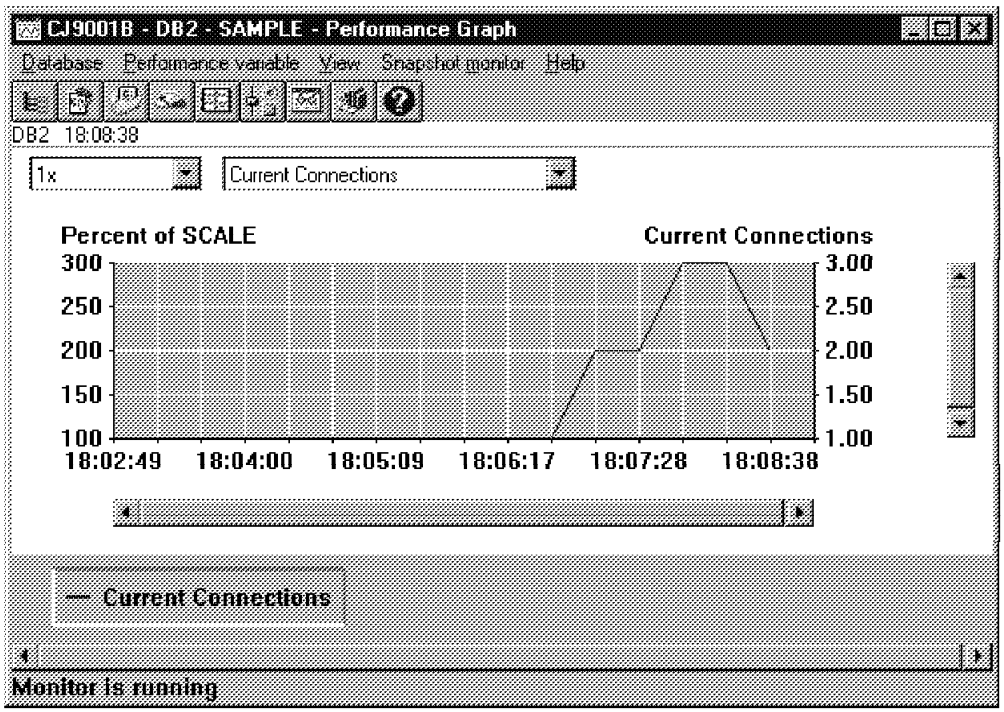


Figure 106. Performance Monitor - Performance Graph

Windows NT Performance Monitor: For monitoring the database and database manager on Windows NT you can also use the Windows NT Performance Monitor. From the Administrative Tools (Common) desktop folder, select **Performance Monitor** and the Performance Monitor window is displayed.

Click the **Add to Chart** option from the Edit menu or the plus sign (+) in the toolbar. Then, from the Object list you can select DB2 NT Applications, DB2 NT Database Manager or DB2 NT Databases counters to be monitored.

If these options are not available, run the `db2perfi -i` command from the SQLLIBBIN directory to install these counters.

Once you add each counter, the monitor graph will reflect the changes as shown in Figure 107 on page 194.

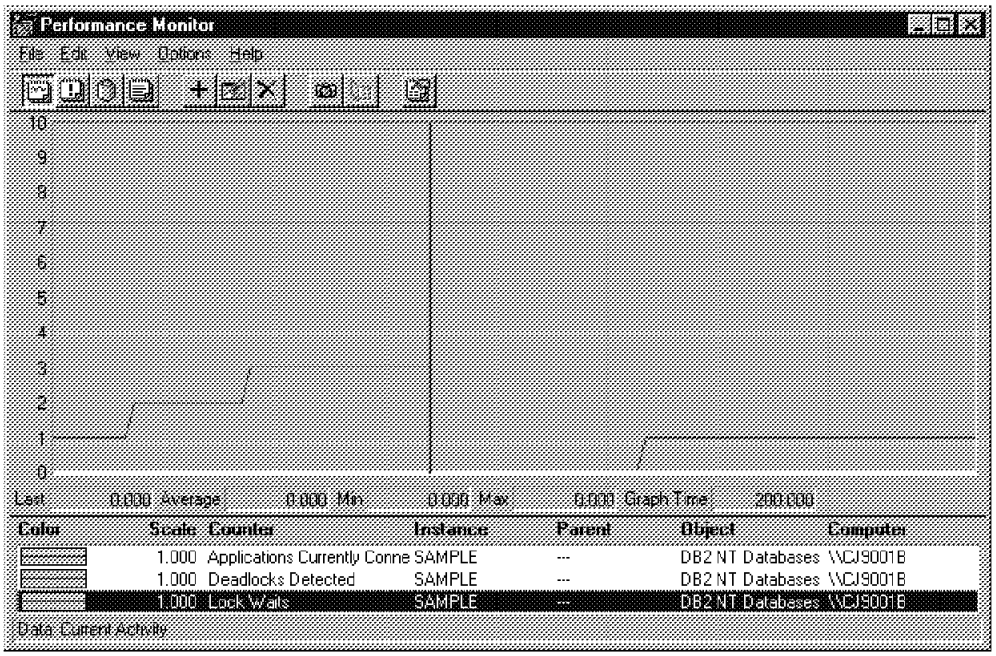


Figure 107. Using NT Performance Monitor to Monitor Database Activity

4.5.2.2 Event Analyzer

The Event Analyzer lets you perform the following tasks:

- Create event monitors to monitor the types of database events.
- Activate the Event Monitor to start collecting event data. Data gathered by the monitor is stored in a file.
- De-activate the Event Monitor to stop collecting event data.
- View the trace-type summary information that is produced by an event monitor.
- Remove the event monitor when you no longer have a need for it. You can also clean up the trace files.
- Display definitions of event monitors associated with the database.
- View the definition of an event monitor.

To create an event monitor, from the Control Center click the mouse button two on the database you want to monitor and select **Monitor Events** from the pop-up menu. The Event Monitors window opens, then select **Event Monitors**, **Create** from the Menu Bar.

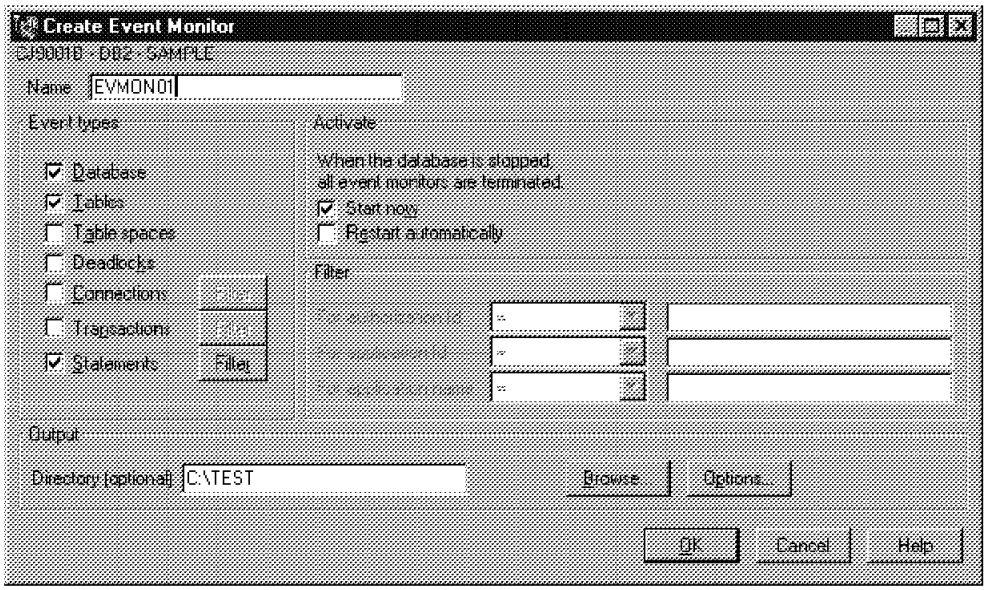


Figure 108. Creating an Event Monitor

In the Create Event Monitor window shown in Figure 108 you are prompted to specify the event monitor name, event types to be monitored, when to activate the monitor and the output directory for the trace files. Click **OK** to create the event monitor.

The number of defined event monitors is unlimited but only 32 can be active at the same time. Remember that the directory specified for the output must exist at event monitor activation and do not specify the same directory for the output of more than one event monitor.

Some event information can be captured based on authorization ID, application name or event condition, these filters can be set by the WHERE clause. Filtering the event monitor output can be used as a form of database auditing.

Connect to the database and generate activity against it, by issuing queries or application transactions Information will be gathered by the event monitor.

Turn off the event monitoring by clicking the mouse button two on the event monitor you want to use and select **Stop Event Monitoring** from the pop-up menu as shown in Figure 109 on page 196.

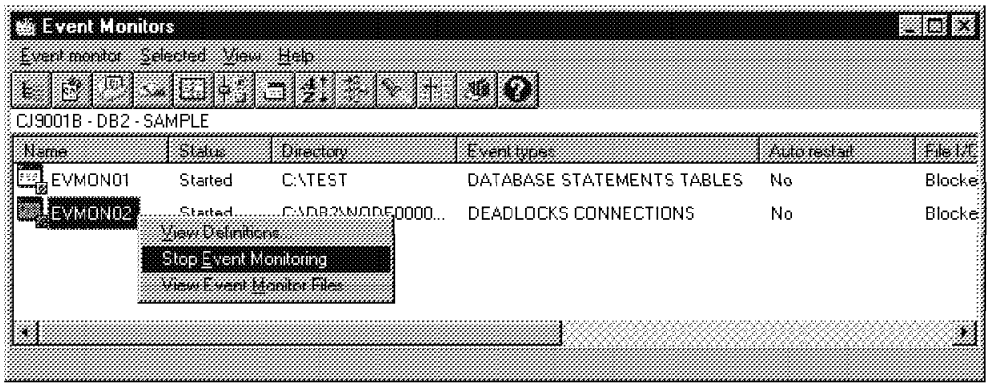


Figure 109. Working with Event Monitors

To review the files generated, you can either:

- Click the mouse button two on the event monitor just stopped and select **View Event Monitor Files** from the pop-up menu as shown in Figure 109, or
- Later, from the Administration Tools folder, double-click on **Event Analyzer** icon. Complete the information requested, such as Event file path, database name and whether you want to view static SQL text or not and click **OK**.

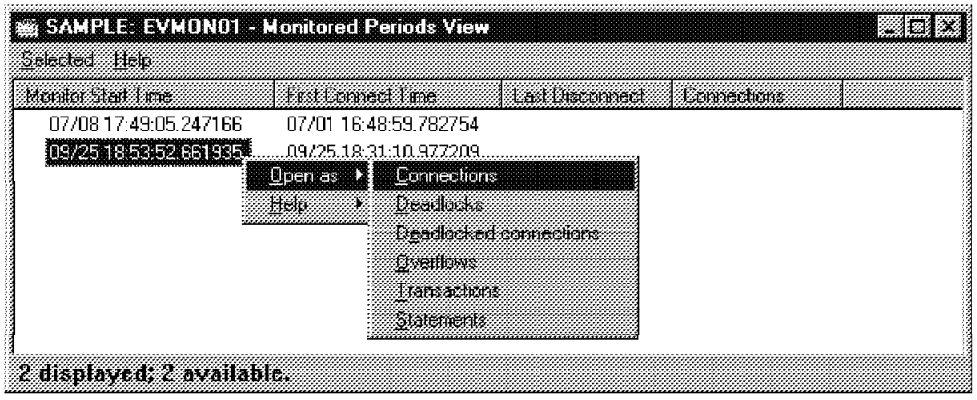


Figure 110. Analyzing Event Monitor Files

Subsequent options will allow you to view and analyze the information gathered for connections, deadlocks, transactions, statements and so on, as shown in Figure 111 on page 197.

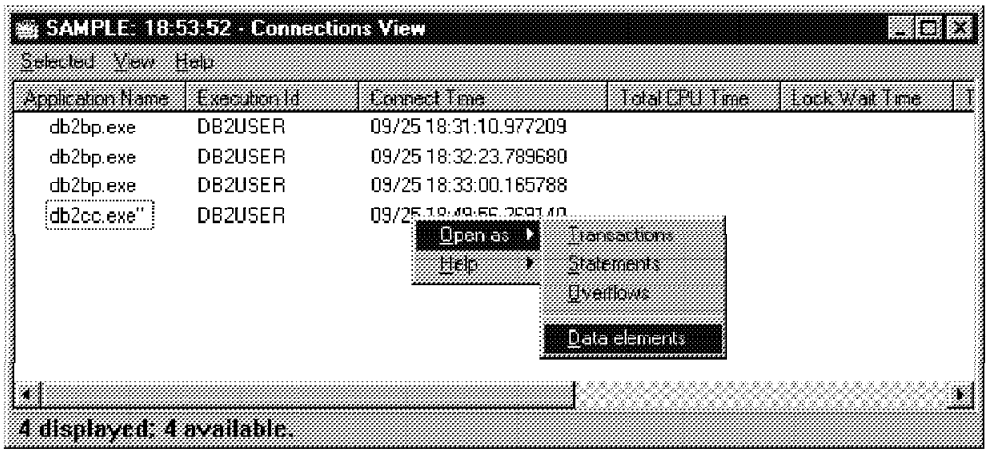


Figure 111. Analyzing Event Monitor Files - at Connections Level

A detailed view of the data elements is shown in Figure 112:

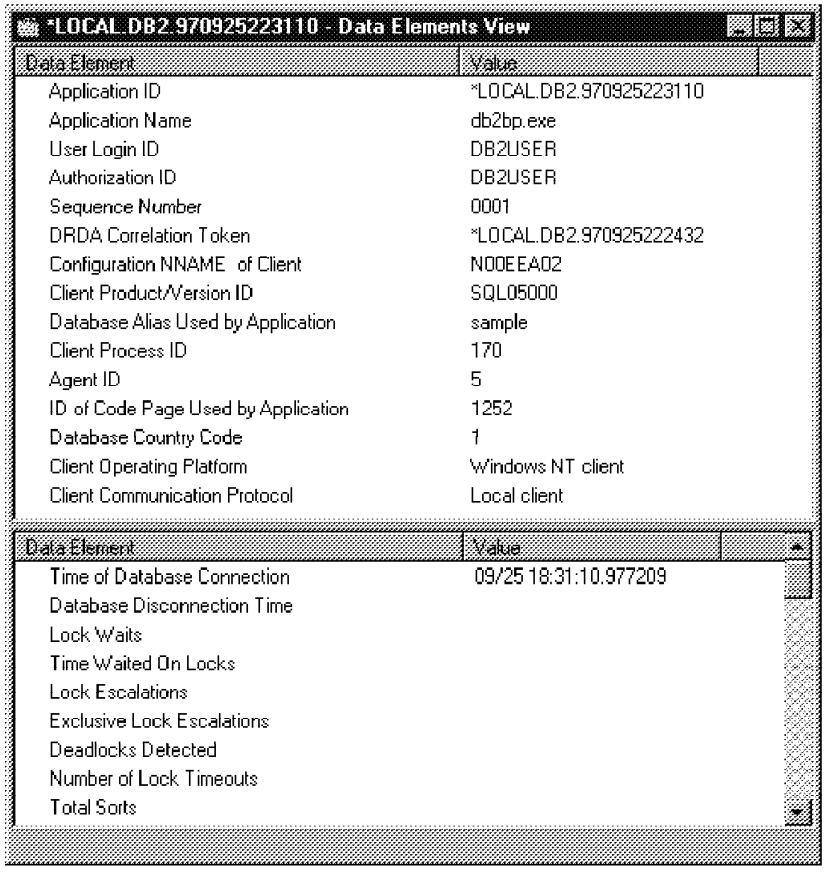


Figure 112. Analyzing Event Monitor Files - Data Elements View

Chapter 5. DB2 UDB V5 Migration on OS/2 and Windows NT

This chapter gives an in-depth view of the migration process from DB2 V1 and DB2 V2 to DB2 UDB V5, on the OS/2 and Windows NT platforms (where appropriate). We will cover the pre-migration, migration and post-migration steps in detail, with emphasis on operating system and DB2 version differences. We will deal with the separate issues of DB2 Server and DB2 Client migration as follows:

- The DB2 Server migration to DB2 UDB V5 is quite complex. When planning the migration, there are many things to take into consideration. The actual migration process is different for each operating system and DB2 version, while the post-migration process includes many optional activities which take advantage of the enhancements introduced in DB2 UDB V5. Hence it is suggested that you read the following sections in this order:
  1. 5.1, “DB2 Server Migration: Planning and Considerations”
  2. Choose one of the following sections, based on your operating system and current DB2 version:
     - 5.2, “Windows NT DB2 V2 Server Migration” on page 204
     - 5.3, “OS/2 DB2 V2 Server Migration” on page 223
     - 5.4, “OS/2 DB2 V1 Server Migration” on page 238
  3. 5.5, “DB2 Server Post-Migration” on page 244
- The DB2 Client migration to DB2 UDB V5 is quite straightforward, by contrast, and we will deal with the migration aspects for all the above operating systems and DB2 versions in the same section. Refer to 5.6, “DB2 Client Migration” on page 249. Note that if you have a DB2 Client system with local databases (for example, a DB2 Single User Server), refer to the above DB2 Server migration process.

Note: The naming convention used throughout this chapter to identify different operating systems, different DB2 versions and whether DB2 is installed as a server or client system is:

- <O/S> DB2 <Version> <Server/Client>

For example:

- Windows NT DB2 V2 Server

5.1 DB2 Server Migration: Planning and Considerations

This section applies to migration scenarios involving:

- Windows NT DB2 V2 Servers
- OS/2 DB2 V2 Servers
- OS/2 DB2 V1 Servers

When you migrate your database, the following events occur:

- The following database entities are migrated:
  - Database configuration file

  - Database system catalog tables
  - Database directories
  - Database log file header
  - Database index files
  - Database data files
  - UDFs and stored procedures
  - user exit (db2uexit)
- The database is relocated to a new database path.
- System catalog tables are changed as follows:
  - New columns are added.
  - New tables are created.
  - A set of catalog views is migrated, and a set of new catalog views is created, in the SYSCAT schema.
  - A set of updateable catalog views is created in the SYSSTAT schema.
  - A set of general purpose scalar functions is kept, and a set of new general purpose scalar functions is created, in the SYSFUN schema. Only SYSFUN.DIFFERENCE scalar function is dropped and re-created during database migration.
- A new directory called db2event is created in the database directory.
- A buffer pool file is created in the database directory.
- A database history file and its shadow are created in the database directory.This file contains a summary of backup information that can be used if a database must be restored, and it is updated whenever a backup, restore, or table load operation is performed on the database. A summary of backup information is also kept for backup and restore operations on a table space.

Hence, as part of the pre-migration process, you must consider the following:

- Migration restrictions
- Security and Authorization
- Storage Requirements
- Release-to-Release incompatibilities

5.1.1 Migration Restrictions

There are certain pre-conditions or restrictions that you should be aware of before attempting to migrate your database to V5:

- Migration is only supported from V1.x or V2.x. Earlier versions of DB2 (Database Manager) must be migrated to V1.x or V2.x before attempting to migrate to V5.
- Issuing the migration command from a V5 client to migrate a database on a V5 Server is supported. However, issuing a migration command from earlier versions of DB2 clients to a V5 Server is not supported.
- Migration between platforms is not supported.

- User objects within your database cannot have V5 reserved schema names as object qualifiers. These reserved schema names include: SYSCAT, SYSSTAT, and SYSFUN.
- User-defined distinct types using the names DATALINK or REFERENCE must be renamed before migrating the database.
- Database objects with a dependency on the SYSFUN.DIFFERENCE function must be dropped before migrating the database. Objects that might have a dependency on this function include: views, constraints, functions and triggers.
- Your database cannot be in one of the following states:
  - Backup pending
  - Roll-forward pending
  - One or more table spaces not in a normal state
  - Transaction inconsistent
- Restore of down-level (V1.x or V2.x) database backups is supported but rolling-forward of down-level logs is not supported.
- Version 5 clients are currently not supported by Version 1 DB2 Database servers with the notable exception of DB2 Parallel Edition V1.2.

5.1.2 Security and Authorization

You need SYSADM authority to migrate your database.

If migrating from DB2 Version 1, you should know that a database cannot be cataloged with a mix of authentication types. The authentication type of the instance, in Version 5, defined in the database manager configuration file. If mixed types are detected during migration from Version 1, you can either stop the migration and change the directories or continue with the migration. If migration continues all the authentication types are changed to blank, and the database uses the authentication type specified in the instance.

To use two databases with different authentication types, a new instance must be created for one of the databases. Under OS/2, this means you must *dual boot* (using the boot manager) to a different partition, and install a new instance of DB2 at the current Version 1 level so it will not interfere with the Version 1 instance on the previous partition. The database should be backed up and restored to a new database under the new instance. It can then be dropped under the old instance and migration can then be run.

Note that there is no DB2 Version 1 for Windows NT.

Beginning with DB2 V2, users and groups are differentiated in SQL statements and the system catalog. As a result, if a user and a group have the same name in the previous version of DB2, the authority and privileges granted to the group must be explicitly re-granted after migration.

During migration, the authorization catalog tables, SYSCAT.DBAUTH, SYSCAT.INDEXAUTH, SYSCAT.PLANAUTH, and SYSCAT.TABAUTH, are checked to determine if existing privileges are for users or groups, and the GRANTEETYPE is defined as follows:

- If the name in the GRANTEE column is a user or is not defined; the GRANTEETYPE is defined as U.

- If the name in the GRANTEE column is a group; the GRANTEETYPE is defined as G.
- If the name in the GRANTEE column is both a user and a group; the GRANTEETYPE is defined as U. Privileges must then be explicitly granted to the group.

5.1.3 Storage Requirements

Space is required for both the old and new catalogs during the migration, and the amount of disk space required will vary depending on the size and complexity of the database as well as the number and size of the database objects. These objects include all tables and views. You should make available at least two times the amount of disk space that the database catalog currently occupies. If there is not enough disk space, migration fails, all changes are rolled back, and an SQLCODE of -1704 with reason code 4 is returned.

You should also consider increasing the database configuration parameters associated with the log files. You should increase *logfilsz*, *logprimary* and *logsecond* to prevent space for these files from running out. If log space is completely used, you will receive a SQLCODE of SQL1704N with a reason code of 3. If this happens, increase the log space parameters and re-issue the database migration command.

5.1.4 Release-to-Release Incompatibilities

To successfully migrate a database, you should consider the impact of the incompatibilities between the two versions of the product. The following incompatibilities deserve special attention before you begin your migration:

- **View Definitions**

  If an existing view from before Version 2 involves SELECT * the view may be unusable after migration. If the view is unusable, attempts to use it, directly or indirectly, will result in SQLCODE -158.

  The view must be dropped and recreated in order to avoid this error. If fewer than the current number of columns in the SELECT * table is desired, the recreated view must specify the needed columns.

- **Configuration Parameters**

  Configuration parameter values are preserved during the migration of the database, with the exception of the parameters shown in Table 16 on page 203:

INSTANCE LEVEL	DATABASE LEVEL
sheapthres	app_ctl_heap_sz
backbufsz	locklist
restbufsz	dbheap
numdb	logfilsiz
tm_database	applheapsz
query_heap_sz	sortheap
max_idleagents	stmtheap
svcename	softmax
	pckcachesz

Table 16. Parameters Changed during Migration

Refer to the 5.5, “DB2 Server Post-Migration” on page 244, for a more detailed explanation of the parameter changes, under the difference DB2 versions.

These are the most significant parameters changed:

- Application Heap Size (*applheapsz*)
- Package Cache Size (*pckcachesz*) (applies to all platforms except DB2 for OS/2 V1.x.)
- Maximum Storage for Lock List (*locklist*)
- Recovery Range and Soft Checkpoint Interval (*softmax*)

For these parameters, the use of the associated heap has changed significantly in Version 5.

- *applheapsz* is reset to the Version 5 default value if the current value is less than the Version 5 default value.
- *pckcachesz* is always reset to the Version 5 default value.

For *locklist*, the DB2 Version 1 value is multiplied by a factor of 32/25. This computed value will be used as the Version 5 parameter value, if this value is greater than the Version 5 default. Otherwise, the Version 5 default will be used.

Note for OS/2 Users: if you are migrating from Version 1, parameters previously allocated in units of 64 KB segments are multiplied by 16 to allow for allocation in units of 4 KB pages. In addition, the *softmax* configuration parameter will be set to the default value, since this parameter is now measured as a percentage of the log records written rather than the number of log records written.

In order to take advantage of Version 5 enhancements, you should re-tune your database manager and database configuration after migrating your databases. To assist in this tuning, you may wish to record and compare configuration parameter values from before and after your migration. You might also want to consider resetting all configuration parameters to their default values after you complete your migration.

- **User exit:**

  DB2 UDB Version 5 has changed the interface it uses to invoke the user exit program to archive and retrieve log files. These new interfaces are documented in the Administration Guide. The name for the user exit program has changed to db2uext2 in Version 5; in previous versions, it was called db2uexit.

  The following should be considered before installing Version 5. The installation of Version 5 automatically migrates Version 2 instances.

  If the Version 2 user exit program, db2uexit, is found in sqllibbin before installation, it will remain in this directory after installation. The db2uext2.exe program will also be installed in this directory. Its function is to invoke db2uexit.cmd (for OS/2) or db2uexit.exe using the version 2 interface. This allows the old user exit program to be used on version 5.

  If db2uexit.exe is in a directory other than sqllibbin, it will remain there after installation, but db2uext2.exe will not be installed in sqllibbin. Following the installation, if you want to use the old user exit, you will have to copy it to sqllibbin, then copy db2uext2.v2 from sqllibmisc, and rename it to db2uext2.exe.

  At a convenient time, you should modify your user exit program to use the new version 5 interfaces. The new user exit program should replace db2uext2 in the sqllibbin, used to support the pre-version 5 exit program, db2uexit, which should be removed.

5.2 Windows NT DB2 V2 Server Migration

This section will guide you step-by-step through a migration of a Windows NT DB2 V2 Server system to DB2 UDB V5.

5.2.1 Windows NT DB2 V2 Server: Pre-Migration

After checking through the Planning and Consideration section, you are ready to prepare your current database server and databases for migration. Remember that this is done using your previous release of DB2, before installation of DB2 UDB V5.

This procedure helps to ensure that all databases on your system are ready for migration to Version 5 of DB2. Log in with administrator access, and perform the following steps:

1. Ensure that all databases are cataloged.

   All databases, local and remote, that are to be migrated, must be cataloged. In a DB2 Command Window, check the database directory with the following command:

   ```
   db2 list database directory
   ```

   Note: Any databases which are not cataloged will not be migrated. Also, because of the new directory structures used by DB2 V5, these databases will not be accessible by the new version of DB2 after migration.

2. Make a backup copy of all databases.

   Migration is not a recoverable process. This problem is further compounded as follows:

- If you back up your database before the Version 5 restricted schema names are changed, you will not be able to restore the database from backup using DB2 V5. Instead, you will have to use the version of the database manager from which you are migrating your databases.
- Regardless of the migration outcome, you will no longer have access to the previous version of the database manager, after DB2 V5 is installed, unless you have it installed in another partition which is hidden.
- You should also be aware that any database transactions done during the period between the time the backup was completed, and the time the upgrade to V5 is complete are not recoverable. That is, if at sometime following the completion of the installation and migration to V5, the database needs to be restored (to a V5 level), the logs from before V5 installation cannot be used in roll-forward recovery.

It is important to make sure you have the most recent backup copy of the database before you continue the migration process. If migration fails, you can reinstall the DB2 code for the previous version, restore from the most recent backup, and try again. Alternatively, you can restore from backups to DB2 UDB V5 after a successful installation.

As an example, in a DB2 Command Window, you can use the following command to backup a database called <database> to an accessible directory called <dir>:

```
db2 backup database <database> to <dir>
```

3. Stop the database manager.

   In a DB2 Command Window, issue the following command:

   ```
   db2stop
   ```

4. Complete all database transactions.

- In a DB2 Command Window, you can check if there are any applications outstanding, with the following command:

  ```
  db2 list applications
  ```

- Ensure all applications are disconnected from the database.

  In a DB2 Command Window, you can force all applications to disconnect, with the following command:

  ```
  db2 force application all
  ```

1. Use the db2ckmig pre-migration utility to verify that your databases can be migrated. Run this utility repeatedly until there are no more errors in the log file which is produced. Note that this is different from the UNIX migration process, where the DB2 V5 product must be installed before db2ckmig can be used.

   DB2 V5 provides the db2ckmig pre-migration utility on the product CD-ROM. The program is located in the <drive>:db2common directory on the Windows NT CD-ROM, and is invoked as follows:

   ```
   db2ckmig <database> -l <logfile> [<user/password>]
   ```

   where:

   - <drive> is the drive letter of the CD-ROM drive containing the DB2 V5 for Windows NT CD-ROM.
   - <database> can be a database alias as listed in the database catalog, or specify all cataloged databases to be checked, with the *-e* flag.

- `-l <logfile>` is a mandatory flag and filename, to specify a file where *db2ckmig* will place all error log output. This file is overwritten each time *db2ckmig* is invoked.
- `<user/password>` is optional, specified as `-u <userid> -p <password>`, and can be used to specify a different user ID and password to access a database.

Note:

- Normally, db2ckmig is run against local databases. However, db2ckmig can be run against remote databases, in which case the database parameter must specify the alias name of the remote database. The log file will be written on your local system.
- The database migration verification tool, db2ckmig does not verify uncataloged databases.

For example, to check a database called sample, issue the following command in a Windows NT Command Window:

```
db2ckmig sample -l c:templog
```

This will run db2ckmig against a database called sample (which may be local or remote), writing any messages to the log file called log, in the directory c:temp.

Note: The database migration verification tool, db2ckmig, when used with the `-e` flag, will attempt to verify all cataloged databases, whether local or remote. If you have a mixture of local and remote databases, with different username and password combinations, be aware that running db2ckmig with the `-e` flag may log numerous error messages when it attempts to verify the remote databases. If this is the case, you may choose to uncatalog these remote databases, run the db2ckmig program, and then recatalog these remote databases again so that during installation, the database directories are migrated. You may also choose to run db2ckmig on each database separately, in which case you should check the log file after each iteration.

If there are any errors in the log file, refer to the following for suggested corrective actions.

- A database is in Backup pending state:

  Perform a backup of the database.

- A database is in Roll-forward pending state:

  Recover the database as required; perform or resume a Roll-forward Database

- **Table space not in normal state:**

  Recover the database and table spaces as required; perform or resume a Roll-forward Database.

- A database is in Database transaction inconsistent state:

  Restart the database to return it to a consistent state.

- The database contains database objects that have a schema name of SYSCAT, SYSSTAT, or SYSFUN:

  These schema names are reserved for the Version 5 database manager. To correct this error, do the following:

  a. Back up the database.

b. Export the data from the database object (catalogs or tables).
c. Drop the object.
d. Recreate the object with the corrected schema name.
e. Import / Load the data into the object.
f. Run db2ckmig against the database again, ensuring that the database passes the db2ckmig check.
g. Make a backup copy of the database.

- The database contains database objects that have a dependency on the SYSFUN.DIFFERENCE function. Possible violated database objects are: Constraint, Function, Trigger, View.

  The SYSFUN.DIFFERENCE function must be dropped and recreated during database migration. However, if there is a database object that is dependent on this function, migration will fail. To correct this error, do the following:

  – **Constraint**

    Issue the ALTER TABLE command to drop the constraint.

  – **Function**

    Issue the DROP FUNCTION command to drop the function dependent on SYSFUN.DIFFERENCE.

  – **Trigger**

    Issue the DROP TRIGGER command to drop the trigger.

  – **View**

    Issue the DROP VIEW command to drop the view.

  Note: Any package dependent on SYSFUN.DIFFERENCE will be marked inoperative after migration. Therefore, the db2ckmig program will not report any package that is dependent on the SYSFUN.DIFFERENCE function.

- The database contains user-defined distinct types that use the type name of DATALINK or REFERENCE.

  The data type names, DATALINK and REFERENCE, are reserved for the Version 5 database manager. To correct this error, you must rename these objects by doing the following:

  a. Back up the database.
  b. Export the data from any tables that are dependent on the data types.
  c. Drop the tables dependent on the data types, and then drop the data types. These drops may drop other objects such as views, indexes, triggers, or functions.
  d. Create data types with different type names and recreate the tables using the new data type names. Recreate any dropped views, indexes, triggers, or functions.
  e. Import/Load the data into the tables.
  f. Run db2ckmig against the database again, ensuring that you make a backup copy of the database.

5.2.2 Windows NT DB2 V2 Server: Installation of DB2 UDB V5

After the pre-migration checks, you are ready to perform the migration process. This involves the installation of DB2 UDB V5, which also automatically migrates all database and node catalogs, and then the migration of local databases. This section will cover the installation of DB2 UDB V5. Migration of local databases will be covered in 5.2.3, “Windows NT DB2 V2 Server: Migrating Databases” on page 220.

Note: If you need to restart the installation for any reason, you may wish to clean up the TCP/IP services file located in winntsystem32driversetc. DB2 UDB V5 adds db2cDB2 and db2iDB2 services for the DB2 instance TCP/IP communications protocol. These usually are at port 50000 and 50001, respectively. However, if you restart the installation, and these entries in the services file already exist, the DB2 UDB V5 installation will append db2cDB20 and db2iDB20 at ports 50002 and 50003, respectively (if these already exist, db2cDB21 at 50004, and db2iDB21 at 50005, and so on). Delete these entries if you see them in the services file then restart the DB2 UDB V5 installation.

5.2.2.1 Terminate DB2 Sessions and Services

Terminate all running DB2 sessions and services by doing the following:

- Make sure you have issued a `db2stop` command in all Windows NT Command Windows, then exit each Command Window.
- Make sure you check which services are active in Windows NT. There are two ways to check this:

  1. From the Start Bar, select **Settings**, then **Control Panel**, and finally **Services**, which will display the *Services* panel as in 48. All related DB2 Services should be stopped by selecting on the DB2 Service to be stopped, and clicking on the **Stop** button.

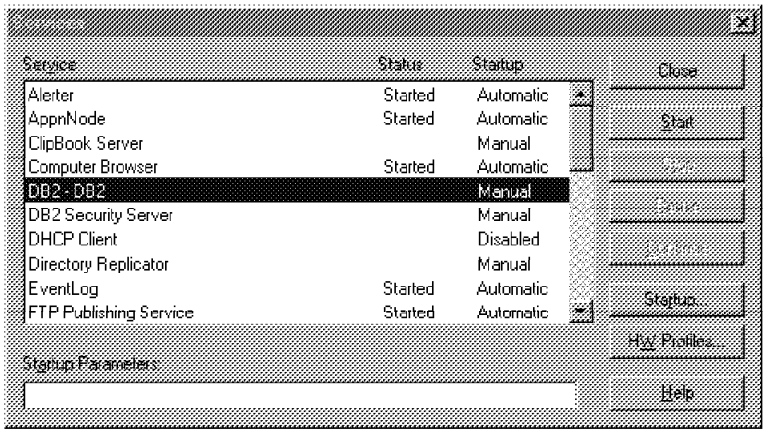


Figure 113. Windows NT Services

  2. From an NT Command Window, execute the `net start` command, which will list the services that have been started, as in Figure 114 on page 209. For each service, issue the command `net stop <service>` (where `<service>` is the Windows NT service to be stopped). For example, to stop the DB2 Security Server, issue the following command (The double quotes are necessary for the net stop command to recognize db2 security server as a single parameter):

```
net stop "db2 security server"
```

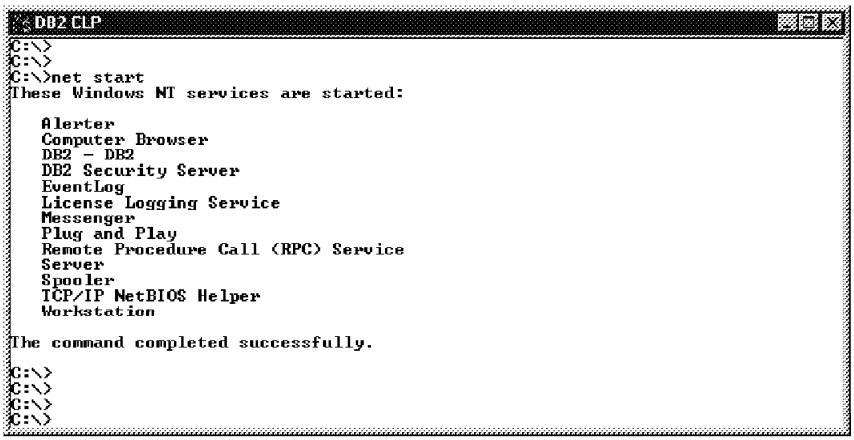


Figure 114. Windows NT Net Start

- Bring up the Windows NT Task Manager, and explicitly end any DB2-related tasks on the local system. To display the NT Task Manager screen, you can do either of the following:
  - Right-click with the mouse cursor on the Task Bar, and select the **Task Manager** choice when the menu choices appear (Figure 115).

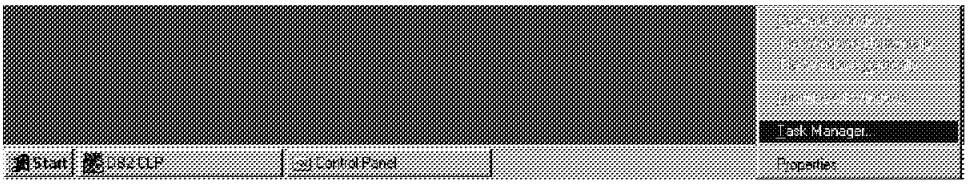


Figure 115. Windows NT Task Bar

  - Press the **Ctrl-Alt-Del** key combination and select **Task Manager** from the resulting pop-up screen (Figure 116)

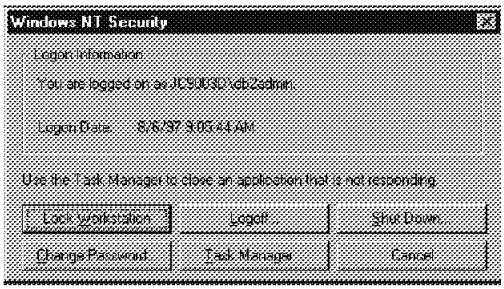


Figure 116. Windows NT Ctrl-Alt-Del

  - Press the **Ctrl-Shift-Esc** key combination, which displays the Task Manager screen as in Figure 117 on page 210.

At the Task Manager screen, you can check for any DB2 processes and end any that are still running. Choose the Applications tab, then select the task in question, and click on the **End Task** button to end the task.

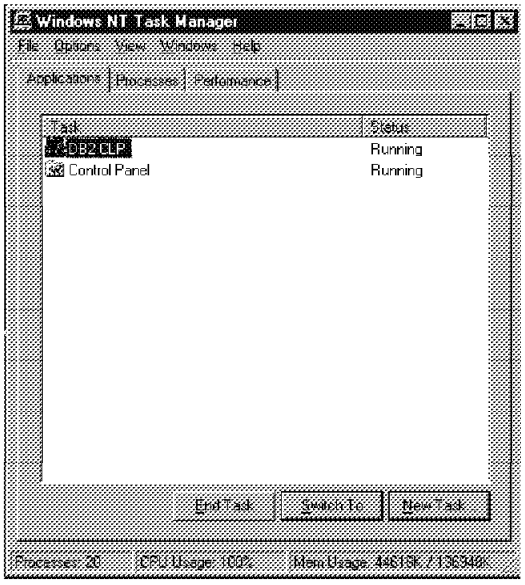


Figure 117. Windows NT Task Manager

Once all DB2 related tasks are terminated, you should rerun the pre-migration tool db2ckmig to ensure migration success, before restarting the DB2 UDB V5 installation.

5.2.2.2 Start the DB2 UDB V5 Installation

To start the installation of DB2 UDB V5, execute the command <drive>setup.exe, where <drive> is the drive letter of the CD-ROM drive containing the DB2 UDB V5 for Windows NT CD-ROM.

If the DB2 UDB V5 installation program detects that there are any DB2 applications still active, the following screen will pop-up (see Figure 118 on page 211). DB2 UDB V5 installation requires that all DB2 applications be terminated before the installation can continue.

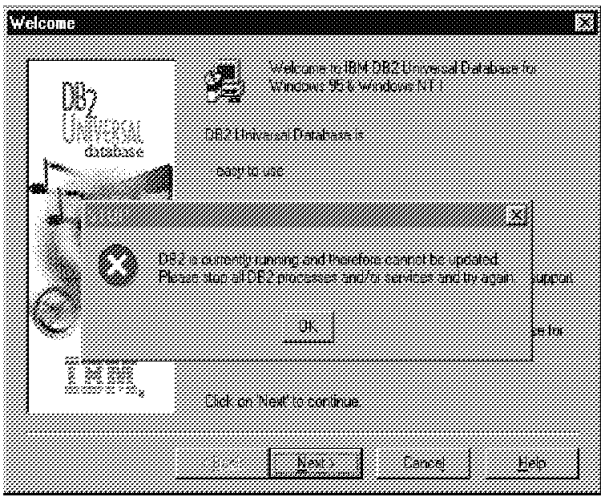


Figure 118. Windows NT DB2 UDB V5: DB2 is Currently Running

Click on the **OK** button to remove this screen. The installation of DB2 UDB V5 will terminate, allowing you to check for any running DB2 applications and services, including any CLP sessions. Terminate all the DB2 sessions and services, as outlined in the previous section, and restart the DB2 UDB V5 installation.

5.2.2.3 Select Products

After space requirements are calculated, you are presented with the *Select Products* screen, listing a choice of DB2 products to install (Figure 119).

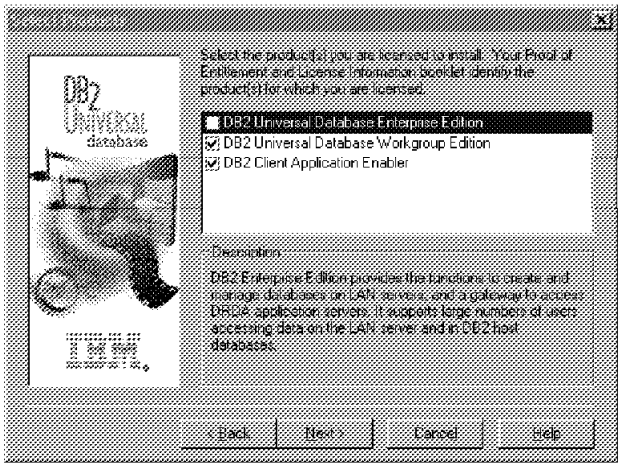


Figure 119. Windows NT DB2 UDB V5 Select Products Screen

Choose one or more products, and click on the **Next** button to continue with the installation. Since there is a previous version of DB2 installed, you will receive a screen prompting you to run db2ckmig to ensure the DB2 migration will run successfully (Figure 120 on page 212). If you have already done so, you can click on the **Next** button to continue with the installation. Otherwise click on the **Cancel** button to exit the installation of DB2 UDB V5. After running db2ckmig, restart the installation.

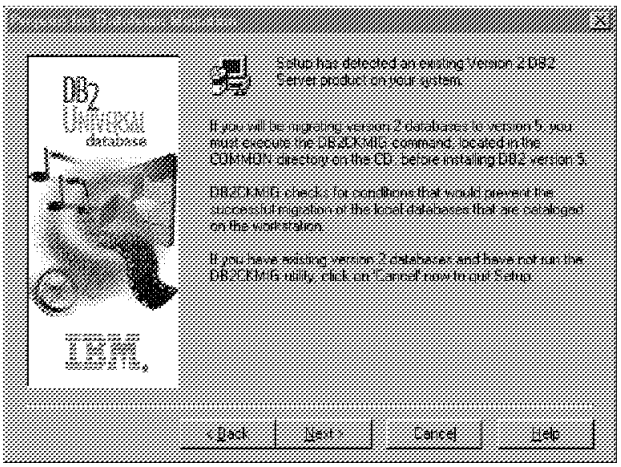


Figure 120. Windows NT DB2 UDB V5 Previous DB2 Version Detected

5.2.2.4 Select Installation Type

Once you choose to continue, you are presented with a screen to *Select Installation Type* (Figure 121). For this example, we will choose *Custom*.

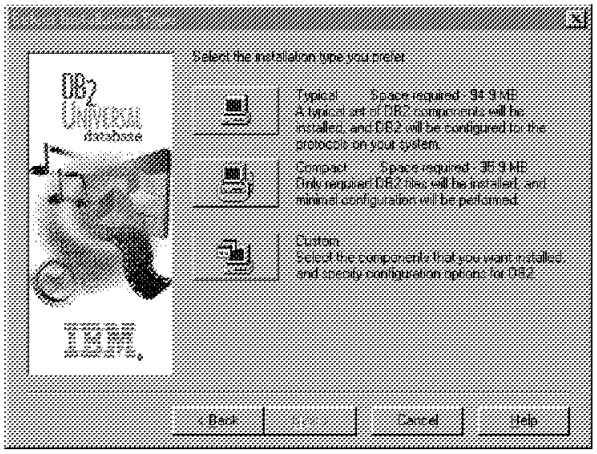


Figure 121. Windows NT DB2 UDB V5 Select Installation Type

5.2.2.5 Select DB2 Components

The *Select DB2 Components* (Figure 122 on page 213) screen allows you to customize your choice of installation for the various DB2 components. You can also change the drive and base path information for the default sqllib directory. The *Space Required* portion of this screen will display the amount of disk space required for the components you have chosen to install.

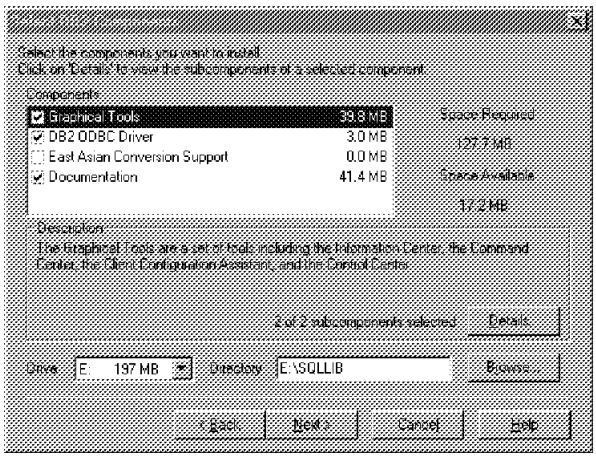


Figure 122. Windows NT DB2 UDB V5 Select DB2 Components

Ensure you have enough space as recommended by the installation program. If the amount of space available is less than the space required, you will receive a warning that there may not be enough space, as in Figure 123.

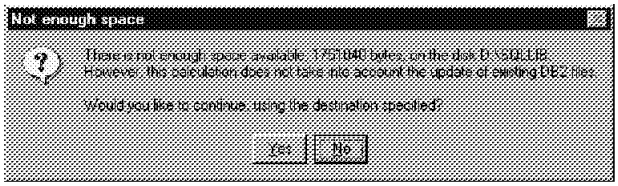


Figure 123. Windows NT DB2 UDB V5 Not Enough Space

Due to the fact that migration is a non-recoverable process, it is recommended that you ensure there is more than enough disk space to complete the migration (see 5.2.1, “Windows NT DB2 V2 Server: Pre-Migration” on page 204). Note that you may encounter a problem with the installation program if you run out of disk space during installation. This will be covered later in this section when the actual installation is started.

5.2.2.6 DB2 Start Options

As part of the Start Options for Windows NT DB2 UDB V5 (Figure 124 on page 214), you can choose to have the DB2 instance automatically started at boot time, and the Control Center automatically started at boot time.

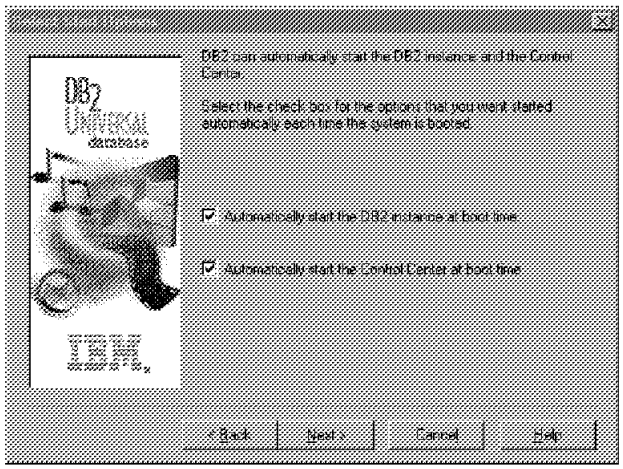


Figure 124. Windows NT DB2 UDB V5 Select Start Options

5.2.2.7 Customize Communications Protocols

Once you have made your choices, click on the **Next** button to continue on to the *Customize Communications Protocols* screen. You can then choose to customize the communications protocols (Figure 125) used by the DB2 instance and the DB2 Administration Server (DAS).

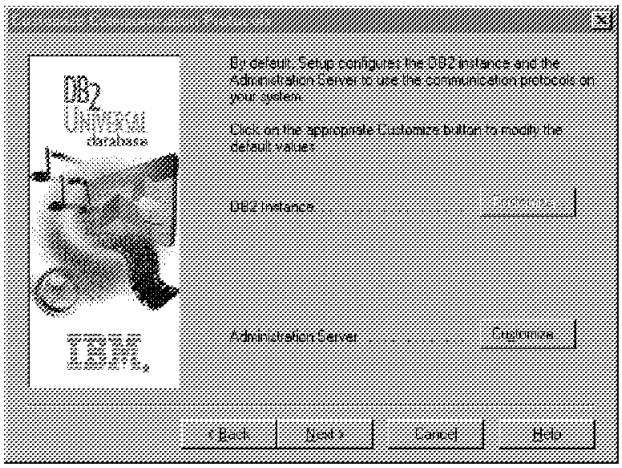


Figure 125. Windows NT DB2 UDB V5 Customize Communication Protocols

Click on the appropriate **Customize** button to customize the respective communications protocol. When you have finished, click on the **Next** button and use the *Enter Username and Password* screen to assign a username and password that the Administration Server will use to log on when it is started as a Windows NT service (Figure 126 on page 215).

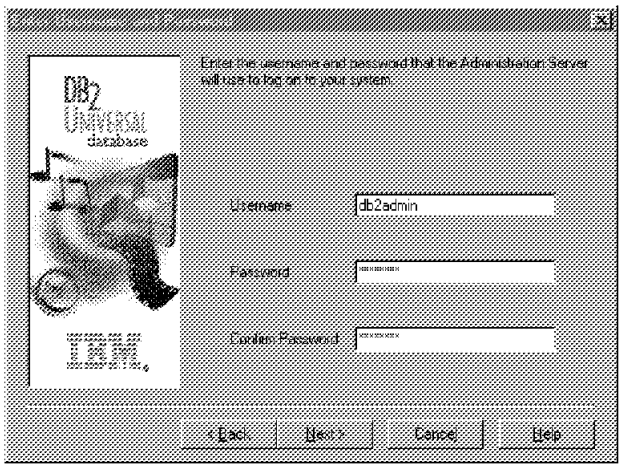


Figure 126. Windows NT DB2 UDB V5 Username and Password for DAS

By default, the setup program will fill the Username, Password, and Confirm Password fields with db2admin. You can accept these default values, or provide your own. If you choose to accept the default values, it is strongly recommended that you change the default password. You should read the following section which covers Windows NT User Manager to make sure that you understand the full implications of this change.

5.2.2.8 Windows NT User Manager

If you provide your own username, you must use the Windows NT User Manager. To access the Windows NT User Manager, click on the **Start Bar**, select **Programs**, **Administration Tools (Common)**, and then finally **User Manager for Domains**. This will invoke the User Manager screen, as in Figure 127.

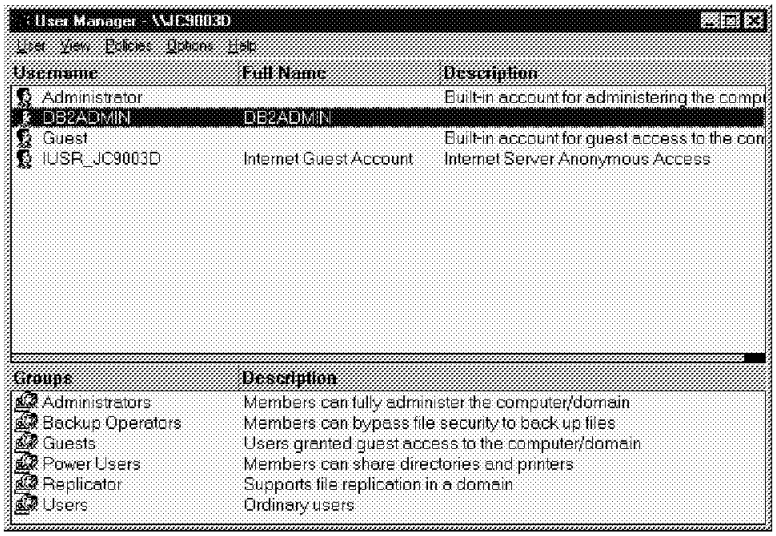


Figure 127. Windows NT User Manager

The DB2 UDB V5 setup program will check to see if the username specified for the Administration Server exists. If it does not, it will be created. If it does exist, the setup program will verify that the username is a member of the Administrators group.

To check group information for a user, from the User Manager, select the **user ID**, the **User** menu item, then select **properties**, or double click on the **user ID**. This will display the User Properties screen as in Figure 128 on page 216.

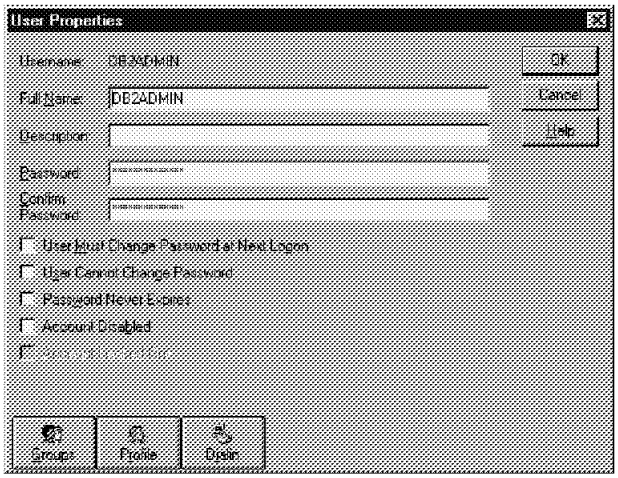


Figure 128. Windows NT User Properties

From the *User Properties* screen, click on the **Groups** button, which will bring up the *Groups Membership* screen as in Figure 129.

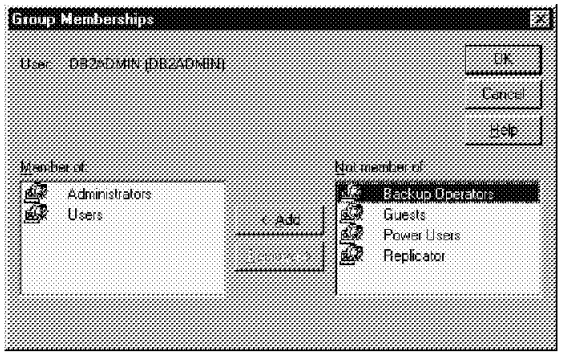


Figure 129. Windows NT Group Memberships

On the left side of this screen, you can see that, in this case, user db2admin is a member of the Administrators group.

You should check that:

- The password is valid.
- The username used to install DB2 has the *Act as part of the operating system* advanced user right.

To check the user rights, from the *User Manager* screen, select the **user ID**, then choose the **Policies** menu item, and then **User Rights**, to show the *User Rights Policy* screen as in Figure 130 on page 217.

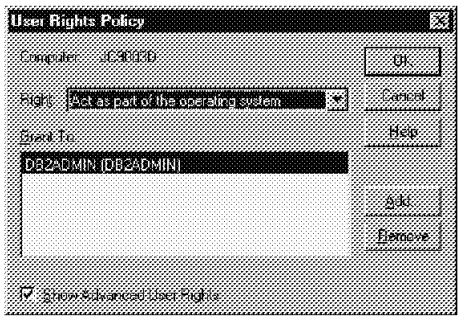


Figure 130. Windows NT User Rights Policy

In order to list the advanced user rights, you must tick the box for **Show Advanced User Rights**, then from the *Right* list box, choose the User Right you wish to check (in this case, *Act as part of the operating system*). This will display a list of user IDs which have been granted the right.

When the setup program creates the db2admin username, it also makes it a member of the Administrators group. Since its password is well known, you should do the following:

1. Change the password for db2admin via the User Properties screen as in Figure 128 on page 216.
2. Change the password for the DB2-DB2DAS00 service to match the new password that you specified for the db2admin username. You can do this in one of the following ways:
   - After DB2 UDB V5 installation and restart, in the Windows NT Services screen, you will see new entries called DB2 and DB2-DB2DAS00, as in Figure 131:

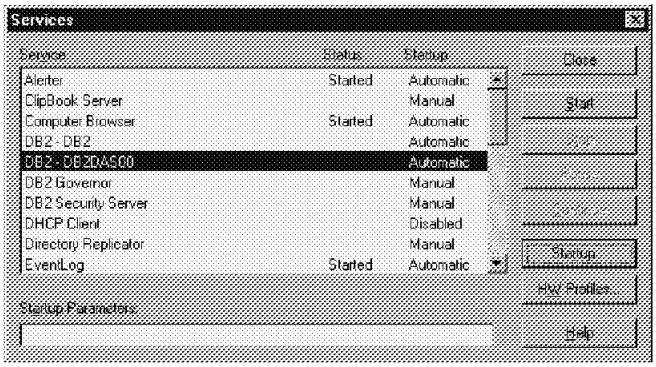


Figure 131. Windows NT Services DB2-DB2DAS00

To change the password, select the **DB2-DB2DAS00** service, then select **Startup**. This will allow you to change the password (Figure 132 on page 218):

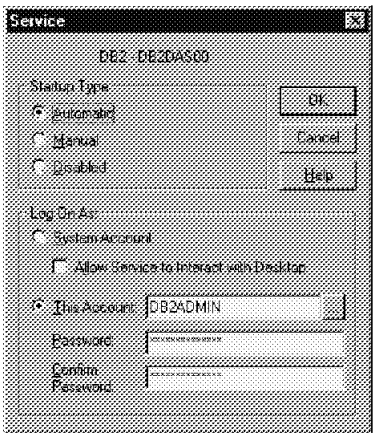


Figure 132. Windows NT Services DB2-DB2DAS00 Startup

- You can also change the password using the db2admin command.

  ```
  db2admin setid <username> <password>
  ```

  Note: This will fail unless the username and password conform to the username and password rules as required for the DB2 Administration Server detailed below in *"DB2 Naming Rules for Usernames and Passwords"* .

In order for this username and password to be used by the Administration Server to log on as a service, the setup program will assign the following user rights to this account:

- Log on as a service.
- Act as part of the operating system.
- Create a token object.
- Increase quotas.
- Replace a process level token.

DB2 Naming Rules for Usernames and Passwords: If you do not use the default username db2admin, the username you specify:

- Must be an existing member of the Administrators group on the local machine.
- Cannot be the same name as your workstation or domain name.
- Must be a valid DB2 username. Valid DB2 usernames:
  - Can contain 1 to 8 characters.
  - Can include letters, numbers, @, #, or $.
  - Cannot begin with IBM, SYS, SQL, or a number
  - Cannot be a DB2 reserved word (USERS, ADMINS, GUESTS, PUBLIC, or LOCAL), or an SQL reserved word.
  - Cannot end with a $.
  - Cannot include accented characters.

Note: Do not use the default *Administrator* username supplied with Windows NT. This is not a valid DB2 username because it has more than eight characters.

The password you specify:

- Should not have expiry, usage restrictions, or require changing.
- Can contain 1 to 8 characters.
- Must begin with:
  - A through Z.
  - @, #, or $.
- Can include:
  - A through Z.
  - 0 though 9.
  - @, #, $, or & .

Note: Windows NT passwords are case-sensitive.

In this example, we already have a database administrator username called db2admin. If the default password provided by the DB2 UDB V5 installation program (also db2admin) is not valid you will encounter a setup error as in Figure 133.

Click on **OK**, and make sure you enter the correct password in both the *Password* and *Confirm Password* fields.

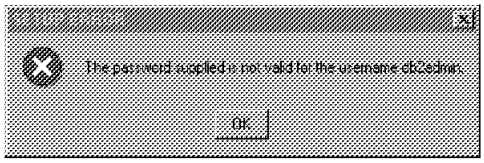


Figure 133. Windows NT DB2 UDB V5 Setup Error

This error is distinct from a mismatch between the Password and Confirm Password field, which will produce the screen in Figure 134.

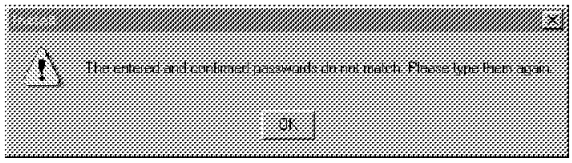


Figure 134. Windows NT DB2 UDB V5 Password Mismatch

5.2.2.9 Start Copying Files

Finally, the last screen labelled *Start Copying Files* allows you to check the current settings (Figure 135 on page 220) for the DB2 UDB V5 installation, before clicking on the **Install** button will start installing the UDB files onto disk. This is the last point at which you can click on the **Back** button to change any settings, or the **Cancel** button to cancel the installation.

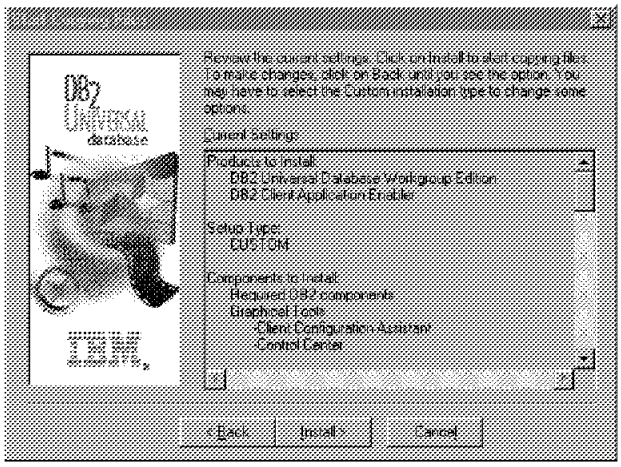


Figure 135. Windows NT DB2 UDB V5 Start Copying Files

Note: If you run out of space during the installation, you may see a pop-up panel as in Figure 136.

Figure 136. Windows NT DB2 UDB V5 Not Enough Disk Space

If you get this message, you may not be able to cancel the installation by clicking on the **Cancel** button. If you free up more disk space and then click on the **OK** button the installation may hang, and in this case the subsequent **Cancel** button will not work. You will have to terminate the application using the Task Manager, as outlined before, and kill the DB2 UDB for NT setup task.

It is recommended that you free up more space, then restart the installation. Note that the previous version of DB2 will be in an inconsistent and uncertain state, since the DB2 V5 installation will have overwritten certain parts of the code and support libraries.

Once the installation completes with no problems, restart the Windows NT system for all changes to take effect.

5.2.3 Windows NT DB2 V2 Server: Migrating Databases

This procedure describes how to migrate cataloged databases after the installation of DB2 UDB V5. You need to perform the following steps:

1. Log on with a user ID that has SYSADM authority.
2. Ensure that the databases you wish to migrate are cataloged.
3. Migrate the database using one of the following:

- The SQLEMGDB migrate database API.

  Refer to the *DB2 UDB V5 API Reference* for details on this API.

- The MIGRATE DATABASE command. From a DB2 Command Window, issue the following command:

  ```
  db2 migrate database <cataloged database name>
  ```

- The RESTORE DATABASE command, when restoring a full backup of the database. For example to restore a database called <database name> from the backup in <path>, from a DB2 Command Window, issue the following command:

  ```
  db2 restore database <database name> from <path>
  ```

5.2.4 Differences Between Windows NT DB2 V2 and DB2 UDB V5

Apart from the DB2 configuration parameter changes for the database and database manager as outlined in the 5.1, “DB2 Server Migration: Planning and Considerations” on page 199 and explained in detail in 5.5, “DB2 Server Post-Migration” on page 244, there are other important changes that are performed as part of the migration process. These changes affect the directory structure, the TCP/IP services file, Windows NT services and Windows NT registry entries.

- **Directory Structure**

  The subdirectory structures underneath the DB2 path have changed in DB2 UDB V5. The directory structure:

  ```
  db2sql00001
  ```

  is now moved into the node0000 sub-directory:

  ```
  db2node0000sql00001
  ```

- **TCP/IP Services**

  In the TCP/IP services file, winntsystem32driversetcservices, there are new TCP/IP services added for DB2 communications, as follows:

  ```
  db2cDB2         50000/tcp   # Connection port for DB2 instance DB2
  db2iDB2         50001/tcp   # Interrupt  port for DB2 instance DB2
  ```

- **NT Services**

  New Windows NT Services are added including DB2DAS00 for the DB2 Administration Server and DB2 Governor for the DB2 Governor. To see which services have been added, you can do one of the following:

  – In an NT Command Window, issue the following command:

    ```
    net start
    ```

    This will list all the Windows NT services that have been started, including the new DB2 UDB V5 services DB2 - DB2 and DB2 - DB2DAS00 as shown in Figure 137 on page 222.

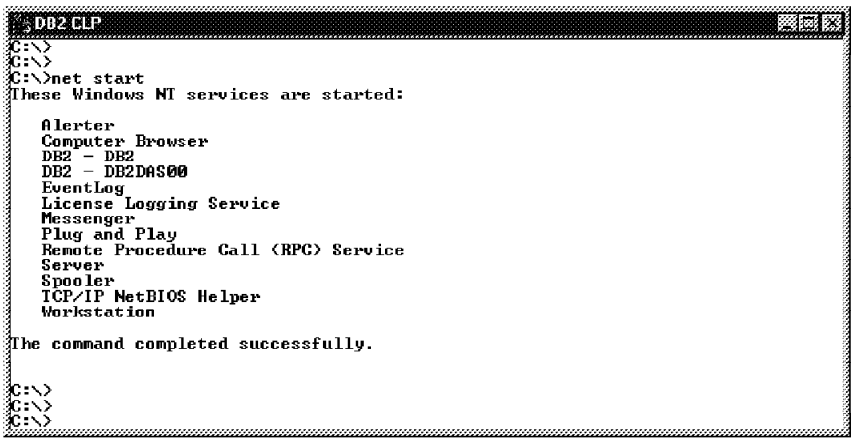


Figure 137. Windows NT Net Start after DB2 UDB V5 Installation

– You can get a complete list of all services, by invoking the Windows NT Services panel (via Start, Settings, Control Panel, Services). This will display the Services screen as in Figure 138:

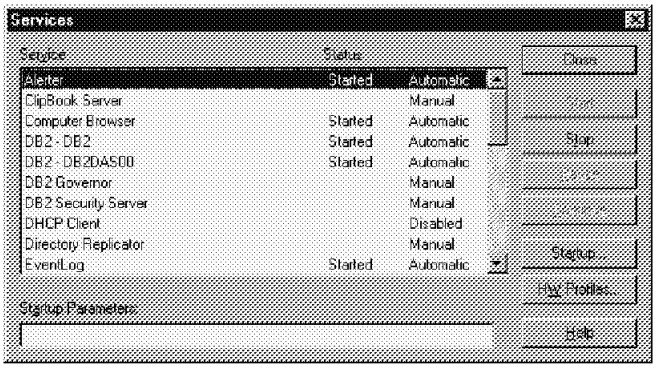


Figure 138. Windows NT Services after DB2 UDB V5 Installation

- **Windows NT Registry**

  Invoke the Windows Registry Editor by executing the command regedit. You will notice that there is a new set of entries for DB2 in the hive:

  [HKEY_LOCAL_MACHINESOFTWAREIBMDB2]

  Among others, you will see entries for the DB2 Global Profile Registry, as well as Instance Profile Registry entries for the DB2 instance and the DB2 Database Administration Server instance.

  – The DB2 Global Profile Registry is located at:

    [HKEY_LOCAL_MACHINESOFTWAREIBMDB2GLOBAL_PROFILE]

    Expected default contents include:

    "DB2PATH"="D:SQLLIB"
    "DB2INSTDEF"="DB2"
    "DB2SYSTEM"="JC9003D"
    "DB2ADMINSERVER"="DB2DAS00"

  – The DB2 Instance Profile Registry for the DB2 instance is located at:

    [HKEY_LOCAL_MACHINESOFTWAREIBMDB2PROFILESDB2]

Example default contents include:

```
"DB2NBADAPTERS"="1"
"DB2COMM"="netbios,TCP/IP,npipe"
"DB2_BDINFO"="132 268623040 2012296232"
```

– The DB2 Instance Profile Registry for the Database Administration Server instance is located at:

```
[HKEY_LOCAL_MACHINESOFTWAREIBMDB2PROFILESDB2DAS00]
```

Example default contents include:

```
"DB2NBADAPTERS"="1"
"DB2COMM"="netbios,TCP/IP,npipe"
"DB2_BDINFO"="138 268623040 2012296232"
```

5.3 OS/2 DB2 V2 Server Migration

This section will guide you step-by-step through a migration of an OS/2 DB2 V2 Server system to DB2 UDB V5.

5.3.1 OS/2 DB2 V2 Server: Pre-Migration

After checking through 5.1, "DB2 Server Migration: Planning and Considerations" on page 199, you are ready to prepare your current database server and databases for migration. Remember that this is done on your previous release of DB2 before installation of DB2 UDB V5.

This procedure helps to ensure that all databases on your system are ready for migration to Version 5 of DB2. Log in with administrator access, and perform the following steps:

1. Complete all database transactions.

   In an OS/2 Command Window, you can check if there are any applications outstanding, with the following command:

   ```
   db2 list applications
   ```

2. Ensure all applications are disconnected from the database.

   In an OS/2 Command Window, you can force all applications to disconnect with the following command:

   ```
   db2 force application all
   ```

3. Ensure all databases are cataloged.

   All databases, local and remote that are to be migrated must be cataloged. In an OS/2 Command Window, display the database directory with the following command:

   ```
   db2 list database directory
   ```

   Note: Any databases which are not cataloged will not be migrated. Also, because of the new directory structure used by DB2 V5, these databases will not be accessible by the new version of DB2 after migration.

4. Make a backup copy of all databases.

   Migration is not a recoverable process. This problem is further compounded as follows:

   - If you back up your database before the Version 5 restricted schema names are changed, you will not be able to restore the database from

backup using DB2 V5. Instead, you will have to use the version of the database manager from which you are migrating your databases.

- Regardless of the migration outcome, you will no longer have access to the previous version of the database manager, after DB2 V5 is installed, unless you have it installed in another partition, which is hidden.
- You should also be aware that any database transactions done during the period between the time the backup was completed and the time the upgrade to V5 is complete are not recoverable. That is, if sometime following the completion of the installation and migration to V5, the database needs to be restored to a V5 level, the logs from before V5 installation cannot be used in roll-forward recovery.

Hence, make sure you have the most recent backup copy of the database before you continue the migration process. This way, if migration fails, you can reinstall the DB2 code for the previous version, restore from the most recent backup, and try again. Alternatively, you can restore from backups to DB2 UDB V5 after a successful installation.

As an example, in an OS/2 Command Window, you can use the following command to backup a database called <database> to an accessible directory called <dir>:

```
db2 backup database <database> to <dir>
```

5. Stop the database manager.

   In an OS/2 Command Window, issue the following command:

   ```
   db2stop
   ```

6. Use the db2ckmig pre-migration utility to check to see if your databases can be migrated. Run this utility repeatedly until there are no more errors in the log file which is produced. Note that this is different to the UNIX migration process, where the DB2 V5 product must be installed first, before db2ckmig can be used.

   DB2 V5 provides the db2ckmig pre-migration utility with the product CD-ROM. The program is located in the <drive>:db2<language>install directory on the OS/2 CD-ROM, and is invoked as follows:

   ```
   db2ckmig <database> -l <logfile> [<user/password>] [<authentication>]
   ```

   where:

   - <drive> is the drive letter of the CD-ROM drive which has the DB2 V5 for OS/2 CD-ROM.
   - <language> is the two-character file name that represents your language (for example, en for English).
   - <database> can be a database alias as listed in the database catalog, or specify all cataloged databases to be checked, with the *-e* flag.
   - -l <logfile> is a mandatory flag and filename, to specify a file where db2ckmig will place all error log output. This file is overwritten each time db2ckmig is invoked.
   - <user/password> is optional, specified as -u userid -p password, and can be used to specify a different user ID and password to access a database.
   - <authentication> is optional, specified as -a <authentication type>, where <authentication type> can be one of CLIENT, SERVER or DCS.

This allows a database to be connected to using an authentication type that is different to the instance authentication type.

Note:

- You can also use slash ″/″ instead ″-″ preceding the flags.
- db2ckmig can be run on remote systems; the database parameter must specify the alias name of the remote database. The log file will be written on your local system.
- The database migration verification tool db2ckmig does not verify uncataloged databases.

For example, to check a database called sample, issue the following command in an OS/2 Command Window:

```
db2ckmig sample -l c:templog
```

This will run db2ckmig against a database called sample (which may be local or remote), writing any messages to the log file called log, in the directory c:temp.

Note: The database migration verification tool, db2ckmig, when used with the -e flag, will attempt to verify all cataloged databases, whether local or remote. If you have a mixture of local and remote databases, with different username and password combinations, be aware that running db2ckmig with the -e flag may log numerous error messages when it attempts to verify the remote databases. If this is the case, you may choose to uncatalog these remote databases, run the db2ckmig program, and then recatalog these remote databases again so that during installation, the database directories are migrated. You may also choose to run db2ckmig on each database separately, in which case, remember to check the log file after each iteration.

If there are any errors in the log file, refer to the following for suggested corrective actions.

- A database is in Backup pending state.

  Perform a backup of the database.

- A database is in Roll-forward pending state.

  Recover the database as required; perform or resume a Roll-forward Database.

- **Table space ID not in normal state.**

  Recover the database and table spaces as required; perform or resume a Roll-forward Database.

- A database is in Database transaction inconsistent state.

  Restart the database to return it to a consistent state.

- The database contains database objects that have a schema name of SYSCAT, SYSSTAT, or SYSFUN.

  These schema names are reserved for the Version 5 database manager. To correct this error, do the following:

  a. Back up the database.

  b. Export the data from the database object (catalogs or tables).

  c. Drop the object.

d. Recreate the object with the corrected schema name.

e. Import / Load the data into the object.

f. Run db2ckmig against the database again, ensuring that the database passes the db2ckmig check.

g. Make a backup copy of the database.

- The database contains database objects that have a dependency on the SYSFUN.DIFFERENCE function. Possible violated database objects are: Constraint, Function, Trigger, View.

  The SYSFUN.DIFFERENCE function must be dropped and recreated during database migration. However, if there is a database object that is dependent on this function, migration will fail. To ensure that database migration is successfully completed and to correct this error, do the following:

  – **Constraint**

    Issue the ALTER TABLE command to drop the constraint.

  – **Function**

    Issue the DROP FUNCTION command to drop the function dependent on SYSFUN.DIFFERENCE.

  – **Trigger**

    Issue the DROP TRIGGER command to drop the trigger.

  – **View**

    Issue the DROP VIEW command to drop the view.

  Note: Any package dependent on SYSFUN.DIFFERENCE will be marked inoperative after migration. Therefore, the db2ckmig program will not report any package that is dependent on the SYSFUN.DIFFERENCE function.

- The database contains user-defined distinct types that use the type name of DATALINK or REFERENCE.

  The data type names, DATALINK and REFERENCE, are reserved for the Version 5 database manager. To correct this error, you must rename these objects by doing the following:

  a. Back up the database.
  b. Export the data from any tables that are dependent on the data types.
  c. Drop the tables dependent on the data types, and then drop the data types. These drops may drop other objects such as views, indexes, triggers, or functions.
  d. Create data types with different type names and recreate the tables using the new data type names. Recreate any dropped views, indexes, triggers, or functions.
  e. Import/Load the data into the tables.
  f. Run db2ckmig against the database again, ensuring that you make a backup copy of the database.

5.3.2 OS/2 DB2 V2 Server: Installation of DB2 UDB V5

After the pre-migration checks, you are ready to perform the migration process. This involves the installation of DB2 UDB V5, which also automatically migrates all database and node catalogs, and then the migration of local databases, with the option to migrate any database catalogs which were not migrated as part of the installation. This section will cover the installation of DB2 UDB V5. Migration of local databases will be covered in the next section.

Note: If you need to restart installation for any reason, you may wish to clean up the TCP/IP services file located in mptnetc. DB2 UDB V5 adds services called db2cDB2 and db2iDB2 services for the DB2 instance TCP/IP communications. These usually are at ports 50000 and 50001, respectively. However, if you restart the installation, and these entries in the services file already exist, the DB2 UDB V5 installation will append db2cDB20 and db2iDB20 at ports 50002 and 50003, respectively (if these already exist, db2cDB21 at 50004, and db2iDB21 at 50005 and so on). Delete these entries if you see them in the services file, then restart the DB2 UDB V5 installation.

5.3.2.1 Start the DB2 UDB V5 Installation

To start the DB2 V5 installation, execute the command `<drive>:<language>installinstall.exe`, where:

- `<drive>` is the drive letter of your CD-ROM drive containing the DB2 V5 for OS/2 CD-ROM
- `<language>` is the two-character file name that represents your language (for example, `en` for English)

From the screen shown in Figure 139, choose one or more products (for example, IBM DB2 Universal Workgroup Edition), the operation (in this case, *Install*, since we are installing DB2 UDB V5), and click on the **Continue** button.

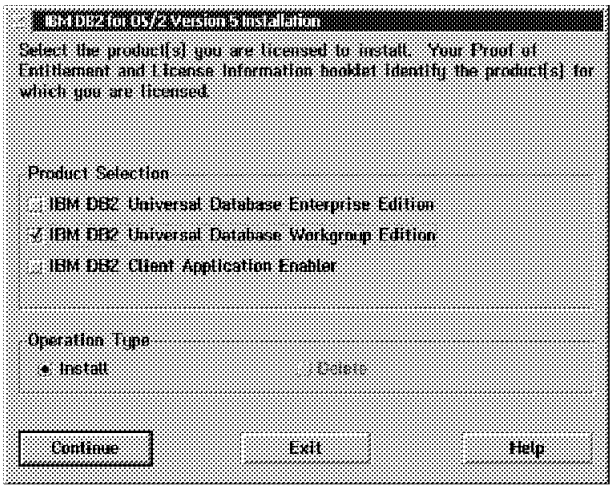


Figure 139. OS/2 DB2 UDB V5 Product Selection

The DB2 UDB V5 installation must update your CONFIG.SYS file (Figure 140 on page 228). A backup is saved, called CONFIG.000 (if this already exists, CONFIG.001 and so on). Leave this option ticked and click on the **OK** button to continue, or the **Cancel** button to cancel the installation.

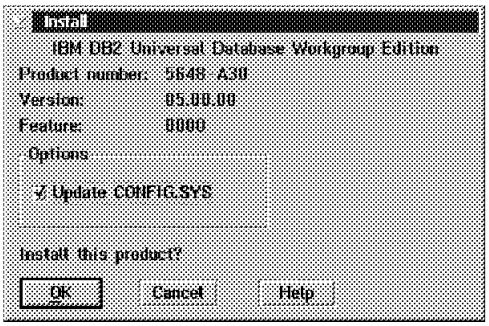


Figure 140. OS/2 DB2 UDB V5 Update CONFIG.SYS

Figure 141 shows that the DB2 UDB V5 installation program has detected a previous version of DB2 and recommends running db2ckmig. At this point, if you have not yet run db2ckmig, you should do this is. Click on the **Cancel** button to cancel the installation, run db2ckmig to check that migration will be successful, and then restart the installation.

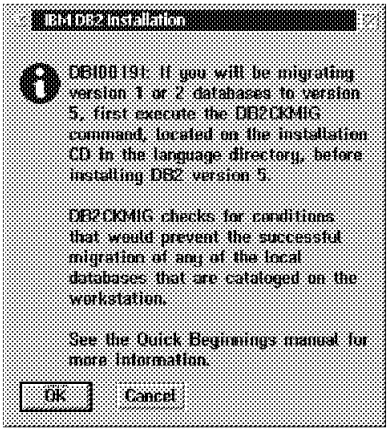


Figure 141. OS/2 DB2 UDB V5 DB2CKMIG Warning

5.3.2.2 Install Directories

Figure 142 on page 229 shows the Install - directories screen, which allows you to customize which components to install, and also change the drive and base path information for the default sqllib directory. Choose your components, and click on the **Disk space...** button to confirm you have enough disk space to perform the installation.

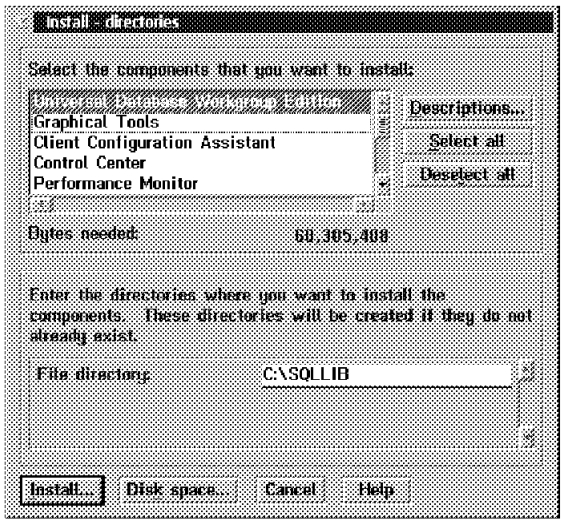


Figure 142. OS/2 DB2 UDB V5 Component Installation

The Disk space screen (Figure 143) displays the amount of free space on each drive, and also shows you the amount required by the DB2 components you have chosen. Note that this does not show the amount of space that would be freed up by any deleted or replaced files belonging to the previous version of DB2.

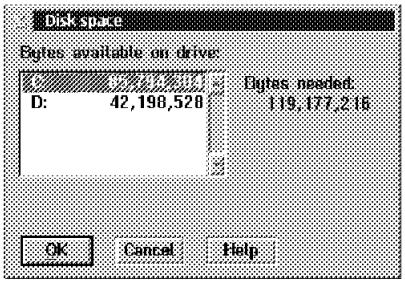


Figure 143. OS/2 DB2 UDB V5 Disk Space

Click on the **OK** button to return to the Install - directories screen. When you clicking on the **Install**... button, you are warned that the installation program will migrate the DB2 configuration and remove the previous version as part of the migration process (Figure 144 on page 230).

Figure 144. OS/2 DB2 UDB V5 IBM DB2 Installation

To continue the installation, click on the **Yes** button, which also confirms that the previous version of DB2 will be removed. Click on **No** to cancel the installation.

The installation program also prompts you to choose whether or not to autostart the Control Center (Figure 145) at boot time.

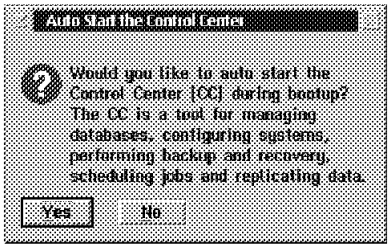


Figure 145. OS/2 DB2 UDB V5 Autostart Control Center

Click on **Yes** or **No** button click to indicate your choice.

5.3.2.3 DB2 System Name

DB2 UDB V5 needs a DB2 System name to uniquely identify your system within your network. In the Specify System Name screen, it will prompt you for a DB2 System name (Figure 146 on page 231).

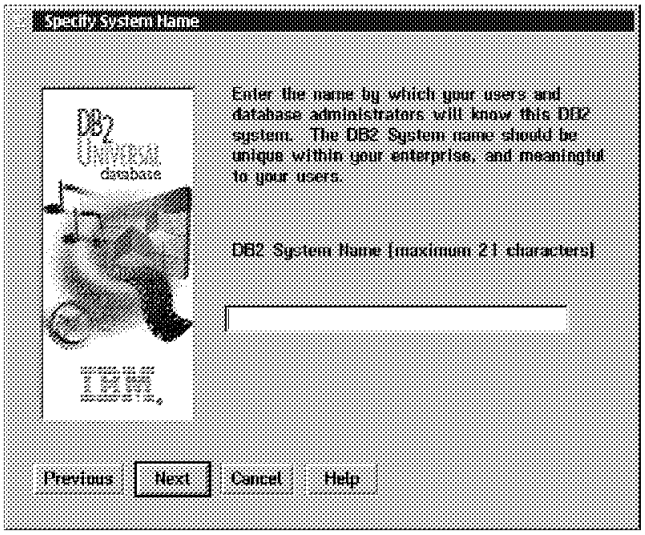


Figure 146. OS/2 DB2 UDB V5 Specify System Name

Note:

- The DB2 System name must conform to the following rules:
  - Contain 1 to 21 characters
  - Include all letters, numbers, and symbols
  - Cannot have spaces embedded
- The DB2 System name is not case sensitive.

5.3.2.4 Customize Communications Protocols

If you click on the **Next** button, you can choose to customize the communications protocols used by the DB2 instance and the DB2 Administration Server in the Customize Communications Protocols screen (Figure 147).

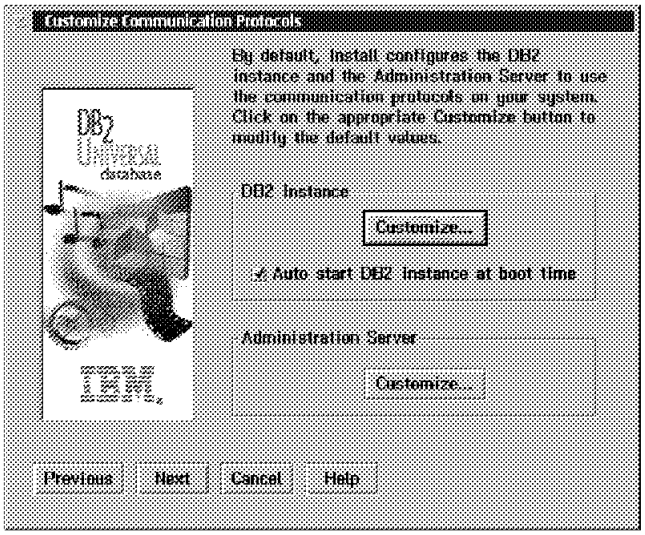


Figure 147. OS/2 DB2 UDB V5 Customize Communications Protocols

5.3.2.5 User ID and Password for DB2 Administration Server

After customizing the communications protocols, click on the **Next** button. You will then need to enter an existing user ID and password for the DB2 Administration Server, as in Figure 148, via the Enter User ID and Password screen. Note that the DB2 UDB V5 installation program will not create a new user ID and password for you.

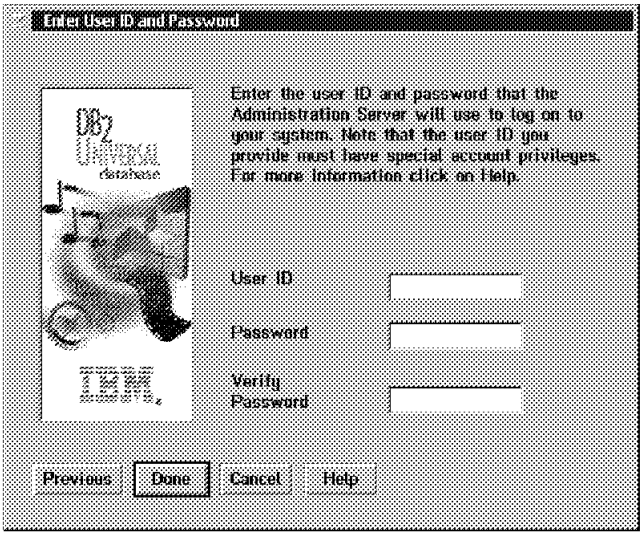


Figure 148. OS/2 DB2 UDB V5 DAS User ID and Password

5.3.2.6 OS/2 User Account Management

This user ID and password must be a valid DB2 logon. To create an account, or check an existing one, issue the upmaccts command in an OS/2 window, or double-click on the **User Account Management** icon (located in the *UPM Services* folder). This will invoke the User Profile Management - User Profile screen as shown in Figure 149.

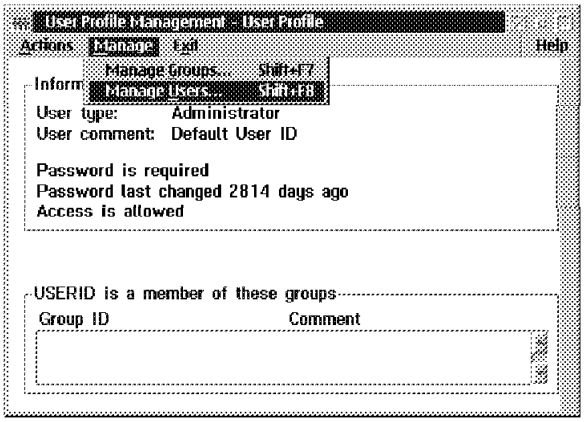


Figure 149. OS/2 UPMACCTS

From the *Manage* menu, choose *Manage Users*. You can also press the **Shift-F8** key combination to invoke the next screen, labelled *User Profile Management - User Management*, as shown in Figure 150 on page 233.

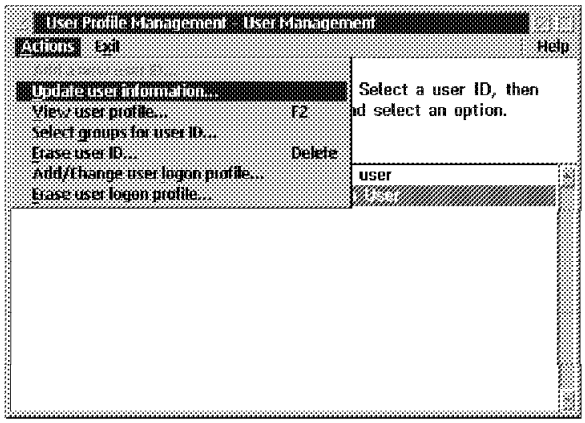


Figure 150. OS/2 UPMACCTS User Management

You can choose to add a new user, or update the information for an existing user. In this case, choosing DB2ADMIN and clicking on **Update User Information** will display Figure 151, the *Update User Information* screen.

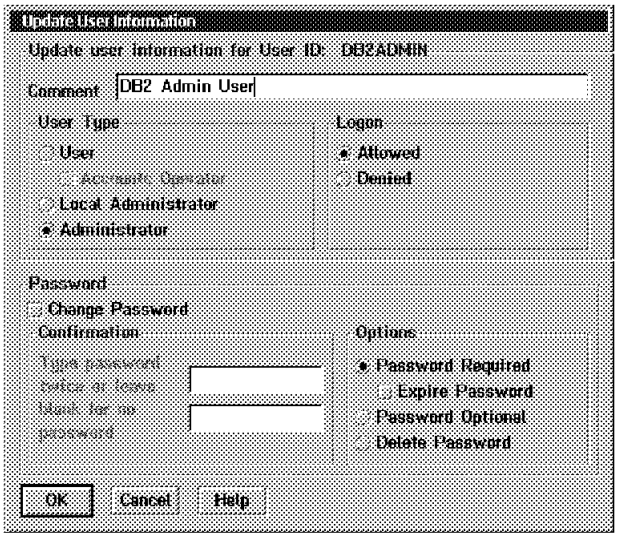


Figure 151. OS/2 UPMACCTS Update User Information

The information presented in this screen allows us to check for the following conditions for the user ID and password:

The user ID you specify:

- must have the user type *Administrator*, or *Local Administrator* assigned to the account.
- must be a valid DB2 username. Valid DB2 usernames:
  - contain 1 to 8 characters
  - can include letters, numbers, @, # or $
  - cannot begin with IBM, SYS, SQL or a number
  - cannot be a DB2 reserved word (USERS, ADMINS, GUESTS, PUBLIC, or LOCAL) or an SQL reserved word

  - cannot end with $
  - cannot include accented characters

The password you specify:

- should not have expiry or require changing
- can contain 1 to 8 characters
- must begin with:
  - A through Z
  - @, #, or $
- can include:
  - A through Z
  - 0 through 9
  - @, #, $, or &

While entering the user ID and password on the *Enter User ID and Password* screen (Figure 148 on page 232) there are two types of errors you may encounter. One is a *User ID and password are invalid* error, as in Figure 152. This error occurs if the user ID is not valid.

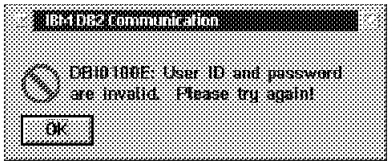


Figure 152. OS/2 DB2 UDB V5 User ID and Password Invalid

The other error occurs if the passwords entered in the *Password* and *Confirm Password* fields do not match, as in Figure 153. (The error message is somewhat misleading).

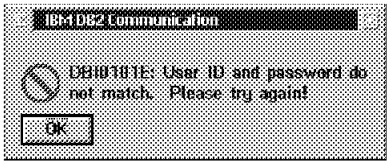


Figure 153. OS/2 DB2 UDB V5 User ID and Password Mismatch

Note: If you have never logged in before, in this working session, even if you type in a valid user ID and password at the *Enter User ID and Password* screen, the system may still display the error screen as in Figure 153, stating the user ID and password do not match.

If this happens, you should remove the error screen by clicking on the **OK** button, then perform a local login (`logon /l` in an OS/2 command window), using any existing user ID and password. As soon as the login is successful, click on the **Done** button on the *Enter User ID and Password* screen (Figure 148 on page 232) and the DB2 UDB V5 installation will continue.

At this point, if the DB2 UDB V5 installation program determines you have insufficient disk space to perform the installation, you will receive a warning screen, as in Figure 154 on page 235.

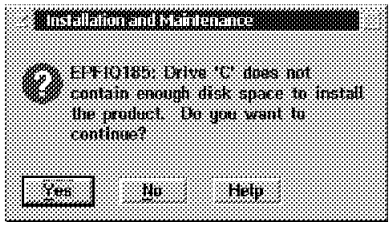


Figure 154. OS/2 DB2 UDB V5 Not Enough Space

Although it is advisable to make sure you have more than enough disk space to complete the DB2 UDB V5 installation, the installation program does not take into account space that would be freed by the removal of files belonging to the previous DB2 version. In this case, you can choose to continue the installation by clicking on the **Yes** button.

Once the installation has started copying files (as indicated by the installation screen showing a progress meter), if you do run out of disk space and cannot free up more space, you will have to terminate the installation and start all over again. The DB2 UDB V5 installation program will delete the files it has copied and you will have to restart the OS/2 system to initiate clean up. Note that the previous DB2 version will no longer be available, as it will have been deleted by the DB2 UDB V5 installation program.

If you do run out of disk space, you should receive a pop-up screen like Figure 155, which indicates the installation program has run out of disk space to copy the required DB2 UDB V5 files. In this case, free up more disk space, and try again with the Retry button.

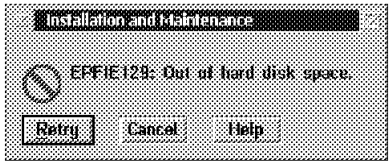


Figure 155. OS/2 DB2 UDB V5 Out of Disk Space

However, you may receive a pop-up screen like Figure 156 on page 236. In this case, it is also advisable to check that you have enough free space in the drive to which you are installing DB2 UDB V5. If not, free up space and try again with the Retry button.

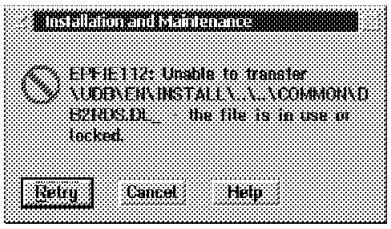


Figure 156. OS/2 DB2 UDB V5 Installation Problem

Once the installation completes with no problems you must restart the OS/2 system.

5.3.3 OS/2 DB2 V2 Server: Migrating Databases

This procedure describes how to migrate cataloged databases during the installation of DB2 products.

Note: The DB2CIDMG migration program, which works in a Configuration/Installation/Distribution (CID) architecture environment, is only available on DB2 for OS/2. It allows for remote unattended installation and configuration on LAN-based workstations. You must have NetView DM/2 on your LAN to use CID migration.

To migrate databases owned by an instance, you need to perform the following steps:

1. Log on with a user ID that has SYSADM authority.
2. Ensure that the databases you wish to migrate are cataloged.
3. Migrate the database using one of the following:
   - The SQLEMGDB migrate database API.

     Refer to the *DB2 UDB V5 API Reference* for details on this API.
   - The MIGRATE DATABASE command line processor command. From an OS/2 command Window, issue the following command:

     `db2 migrate database <cataloged database name>`
   - The RESTORE DATABASE command, when restoring a full backup of the database. For example, to restore a database called <database name> from the backup in <path>, in a DB2 Command Window, issue the following command:

     `db2 restore database <database name> from <path>`

Note: During installation of V5, all of the found local database directories are migrated. It may be that you require keeping one of your current local database directories after the installation of Version 5. For example, you may have chosen to install two instances of DB2 on separate partitions to keep Version 2 still intact on one partition. If you keep your current system directories, then you may need a way to migrate that database directory to the Version 5 format at some time later. The DB2MIGDR utility allows you to complete this migration.

In an OS/2 Command Window, issue the following command:

`db2migdr <drive>`

where <drive> is the drive letter where your system directories are located.

5.3.4 Differences Between OS/2 DB2 V2 Server and DB2 UDB V5

Apart from the DB2 configuration parameter changes for the database and database manager as outlined in the 5.1, “DB2 Server Migration: Planning and Considerations” on page 199 and explained in detail in the 5.5, “DB2 Server Post-Migration” on page 244, there are other changes of note that are performed as part of the migration process. These affect the directory structure, TCP/IP services file and DB2 registry files for OS/2.

- **Directory Structure**

  Regardless of the previous DB2 version, you have the SQLLIB directory in the base path of your installation. However, in DB2 UDB V5, the SQL00001 and SQLDBDIR directories are moved to a new path:

  `sql00001`

  is now located in the db2node0000 sub-directory:

  `db2node0000sql00001`

  The local database directory:

  `sqldbdir`

  is now located in the db2node0000 sub-directory:

  `db2node0000sqldbdir`

  After the installation and migration, you will still see a sqldbdir directory in the path sqldbdir, but empty. (UDB V5 uses db2node0000sqldbdir).

- **TCP/IP Services**

  In the TCP/IP services file, located in mptnetcservices, there are new TCP/IP services added for use by the default DB2 Database Administration Server (DB2DAS00), and for DB2 communications, as follows:

```
db2cDB2admin    523/tcp     # Connection port for DB2 instance DB2DAS00
db2iDB2admin    524/tcp     # Interrupt  port for DB2 instance DB2DAS00
db2cDB2         50000/tcp   # Connection port for DB2 instance DB2
db2iDB2         50001/tcp   # Interrupt  port for DB2 instance DB2
```

- **DB2 Registry**

  DB2 UDB V5 writes DB2 registry information into several flat text files, rather than using any OS/2 .ini files. These profile registry files exist for DB2, the DB2 Global Profile Registry, and Instance Profile Registries for the default DB2 instance and the default DB2 Database Administration Server instance.

  – The DB2 Profiles Registry is located in:

    `sqllib\profiles.reg`

    This file contains the names of all instances for which DB2 has profile registry information. Expected default contents include:

```
DB2
DB2DAS00
```

  – The DB2 Global Profile Registry is located in:

    `sqllib\profiles.reg`

    Note that DB2SYSTEM contains the system name you typed in Figure 146 on page 231. Expected default contents include:

```
DB2SYSTEM=OS/2DB2
DB2SERVICESTPINSTANCE=DB2
DB2INSTDEF=DB2
DB2COMM=NPIPE,TCP/IP
DB2CHKPTR=ON
DB2ADMINSERVER=DB2DAS00
```

- The DB2 Instance Profile Registry for the DB2 instance is located in:

  `sqllibdb2profile.env`

  Expected default contents include:

  `DB2COMM='TCP/IP'`

- The DB2 Instance Profile Registry for the Database Administration Server instance is located in:

  `sqllibdb2das00profile.env`

  Expected default contents include:

  `DB2COMM='TCP/IP'`

5.4 OS/2 DB2 V1 Server Migration

This section will guide you step-by-step through a migration of an OS/2 DB2 V1 Server to DB2 UDB V5.

5.4.1 OS/2 DB2 V1 Server: Pre-Migration

After checking through the Planning and Consideration section, you are ready to prepare your current database server and databases for migration. Remember that this is done on your previous release of DB2 before installation of DB2 UDB V5.

This procedure helps to ensure that all databases on your system are ready for migration to Version 5 of DB2. Log in with administrator access, and perform the following steps:

1. Complete all database transactions.

   For DB2 V1 for OS/2, from an OS/2 Command Window, the nearest equivalent to the V2 command db2 list applications is:

   `dbm get database status`

   This shows the number of current connections to the local databases.

2. Ensure all applications are disconnected from the database.

   In DB2 V1 for OS/2, there is no equivalent to the V2 command db2 force application all. You will have to ensure there are no connections, by issuing a `stopdbm` in an OS/2 Command Window. Issue a `startdbm`, in an OS/2 Command Window in order to complete the following steps.

3. Ensure all databases are cataloged.

   All databases, local and remote, that are to be migrated, must be cataloged. In an OS/2 Command Window, check the database listing with the following command:

   `dbm list database directory`

Note that any databases which are not cataloged will not be migrated. Also, because of the new directory structure used by DB2 V5, these databases will not be accessible by the new version of DB2 after migration.

4. Make a backup copy of all databases.

   Migration is not a recoverable process. This problem is further compounded as follows:

   - If you back up your database before the V5 restricted schema names are changed, you will not be able to restore the database from backup using DB2 V5. Instead, you will have to use the version of the database manager from which you are migrating your databases.
   - Regardless of the migration outcome, you will no longer have access to the previous version of the database manager after DB2 V5 is installed unless you have it installed in another partition, which is hidden.
   - You should also be aware that any database transactions done during the period between the time the backup was completed, and the time the upgrade to V5 is complete are not recoverable. That is, if sometime following the completion of the installation and migration to V5, the database needs to be restored (to a V5 level), the logs from before V5 installation cannot be used in roll-forward recovery.

   Hence, make sure you have the most recent backup copy of the database before you continue the migration process. This way, if migration fails, you can either reinstall the DB2 code for the previous version, restore from the most recent backup, and try again, or restore from backups to DB2 UDB version 5 when that has successfully been installed.

   As an example, in an OS/2 Command Window, you can use the following command to backup a database called <database> to an accessible drive called <drive>:

   ```
   dbm backup database <database> ALL to <drive>
   ```

   This will create two directories, <drive>:SQLUIF and <drive>:BACKUP, which contains control and backup information for the database backup, respectively.

5. Stop the database manager.

   In an OS/2 Command Window, issue the following command:

   ```
   stopdbm
   ```

6. Use the db2ckmig pre-migration utility to check to see if your databases can be migrated. Re-run the utility until there are no more errors in the log file produced. Note that this is different to the UNIX migration process, where the DB2 V5 product must be installed first, before db2ckmig can be used.

   DB2 V5 provides the db2ckmig pre-migration utility with the product CD-ROM. The program is located in the <drive>:db2<language>install directory on the OS/2 CD-ROM, and is invoked as follows:

   ```
   db2ckmig <database> -l <logfile> [<user/password>] [<authentication>]
   ```

   where:

   - <drive> is the drive letter of the CD-ROM drive which has the DB2 V5 for OS/2 CD-ROM.
   - <language> is the two-character file name that represents your language (for example, en for English).

- <database> can be a database alias as listed in the database catalog, or specify all cataloged databases to be checked, with the -E flag.
- -l <logfile> is a mandatory flag and filename, to specify a file where db2ckmig will place all error log output. This file is overwritten each time db2ckmig is invoked.
- <user/password> is optional, specified as -u userid -p password, and can be used to specify a different user ID and password to access a database.
- <authentication> is optional, specified as -a <authentication type>, where <authentication type> can be one of client, server or DCS. This allows a database to be connected to using an authentication type that is different to the instance authentication type.

Note:

- You can also use slash "/" instead "-" preceding the flags.
- db2ckmig can be run on remote systems; the database parameter must specify the alias name of the remote database. The log file will be written on your local system.
- The database migration verification tool db2ckmig does not verify uncataloged databases.

For example, to check a database called sample, issue the following command in an OS/2 Command Window:

```
db2ckmig sample -l c:templog
```

This will run db2ckmig against a database called sample (which may be local or remote), writing any messages to the log file called log, in the directory c:temp.

Note: The database migration verification tool, db2ckmig,when run with the -e flag, will attempt to verify all cataloged databases, whether local or remote. If you have a mixture of local and remote databases, with different username and password combinations, be aware that running db2ckmig with the -e flag may log numerous error messages when it attempts to verify the remote databases. If this is the case, you may choose to uncatalog these remote databases, run the db2ckmig program, and then recatalog these remote databases again so that during installation, the database directories are migrated. You may also choose to run db2ckmig on each database separately, in which case remember to check the log file after each iteration.

If there are any errors in the log file, refer to the following for suggested corrective actions.

- A database is in Backup pending state.

  Perform a backup of the database.

- A database is in Roll-forward pending state.

  Recover the database as required; perform or resume a Roll-forward Database.

- Table space ID not in normal state.

  Recover the database and table spaces as required; perform or resume a Roll-forward Database.

- A database is in Database transaction inconsistent state.

Restart the database to return it to a consistent state.

- The database contains database objects that have a schema name of SYSCAT, SYSSTAT, or SYSFUN.

  These schema names are reserved for the Version 5 database manager. To correct this error, do the following:

  a. Back up the database.
  b. Export the data from the database object (catalogs or tables).
  c. Drop the object.
  d. Recreate the object with the corrected schema name.
  e. Import / Load the data into the object.
  f. Run db2ckmig against the database again, ensuring that the database passes the db2ckmig check.
  g. Make a backup copy of the database.

- The database contains database objects that have a dependency on the SYSFUN.DIFFERENCE function. Possible violated database objects are: Constraint, Function, Trigger, View.

  The SYSFUN.DIFFERENCE function must be dropped and recreated during database migration. However, if there is a database object that is dependent on this function, migration will fail. To ensure that database migration is successfully completed and correct this error, do the following:

  – Constraint

    Issue the ALTER TABLE command to drop the constraint.

  – Function

    Issue the DROP FUNCTION command to drop the function dependent on SYSFUN.DIFFERENCE.

  – Trigger

    Issue the DROP TRIGGER command to drop the trigger.

  – View

    Issue the DROP VIEW command to drop the view.

  Note: Any package dependent on SYSFUN.DIFFERENCE will be marked inoperative after migration. Therefore, the db2ckmig program will not report any package that is dependent on the SYSFUN.DIFFERENCE function.

- The database contains user-defined distinct types that use the type name of DATALINK or REFERENCE.

  The data type names, DATALINK and REFERENCE, are reserved for the Version 5 database manager. To correct this error, you must rename these objects by doing the following:

  a. Back up the database.
  b. Export the data from any tables that are dependent on the data types.
  c. Drop the tables dependent on the data types, and then drop the data types. These drops may drop other objects such as views, indexes, triggers, or functions.

d. Create data types with different type names and recreate the tables using the new data type names. Recreate any dropped views, indexes, triggers, or functions.
e. Import/Load the data into the tables.
f. Run db2ckmig against the database again, ensuring that you make a backup copy of the database.

5.4.2 OS/2 DB2 V1 Server: Installation of DB2 UDB V5

After the pre-migration checks, you are ready to perform the migration process. This involves the installation of DB2 UDB V5, which also automatically migrates all database and node catalogs, and then the migration of local databases (with the option to migrate any database catalogs which were not migrated as part of the installation).

As the installation of DB2 UDB V5 onto a DB2 Server V1 system is identical to installing DB2 UDB V5 onto a DB2 Server V2 system, refer to 5.3.2, “OS/2 DB2 V2 Server: Installation of DB2 UDB V5” on page 227 for a step-by-step guide to this task.

5.4.3 OS/2 DB2 V1 Server: Migrating Databases

This procedure describes how to migrate cataloged databases during the installation of DB2 products.

Note: The DB2CIDMG migration program, which works in a Configuration/Installation/Distribution (CID) architecture environment, is only available on DB2 for OS/2. It allows for remote unattended installation and configuration on LAN-based workstations. You must have NetView DM/2 on your LAN to use CID migration.

To migrate databases owned by an instance, you need to perform the following steps:

1. Log on with a user ID that has SYSADM authority.
2. Ensure that the databases you wish to migrate are cataloged.
3. Migrate the database using one of the following:
   - The SQLEMGDB migrate database API.

     Refer to the *DB2 UDB V5 API Reference* about how to use this API.
   - The MIGRATE DATABASE command line processor command. From an OS/2 Command Window, issue the following command:

     `db2 migrate database <cataloged database name>`
   - The RESTORE DATABASE command, when restoring a full backup of the database. For example, to restore a database called `<database name>` from the backup in `<path>`, in a DB2 Command Window, issue the following command:

     `db2 restore database <database name> from <path>`

Note: During installation of V5, all of the found local database directories are migrated. It may be that you require keeping one of your current local database directories after the installation of Version 5. For example, you may have chosen to install two instances of DB2, on separate partitions, to keep Version 2 still intact on one partition. If you keep your current system directories, then you

may need a way to migrate that database directory to the Version 5 format at some time later. The DB2MIGDR utility allows you to complete this migration.

In an OS/2 Command Window, issue the following command:

```
db2migdr <drive>
```

where <drive> is the drive letter where your system directories are located.

5.4.4 Differences between OS/2 DB2 V1 Server and DB2 UDB V5

Apart from the DB2 configuration parameter changes for the database and database manager, as outlined in the 5.1, “DB2 Server Migration: Planning and Considerations” on page 199, and explained in detail in the 5.5, “DB2 Server Post-Migration” on page 244, there are other changes of note, that are performed as part of the migration process. These affect the directory structure, TCP/IP services file and DB2 registry files for OS/2.

- Directory Structure

  Regardless of the DB2 instance version, you have the SQLLIB directory in the base path of your installation. However, in UDB V5, the SQL00001 and SQLDBDIR directories are moved to a new path:

  ```
  sql00001
  ```

  is now located in the db2node0000 sub-directory:

  ```
  db2node0000sql00001
  ```

  The local database directory:

  ```
  sqldbdir
  ```

  is now located in the db2node0000 sub-directory:

  ```
  db2node0000sqldbdir
  ```

  After the installation and migration, you will still see a sqldbdir directory in the path sqldbdir, but empty. (UDB V5 uses db2node0000sqldbdir).

- TCP/IP Services file

  In the TCP/IP services file, located in mptnetcservices, there are new TCP/IP services added for use by the default DB2 Database Administration Server (DB2DAS00) and for DB2 communications, as follows:

  ```
  db2cDB2admin    523/tcp     # Connection port for DB2 instance DB2DAS00
  db2iDB2admin    524/tcp     # Interrupt  port for DB2 instance DB2DAS00
  db2cDB2         50000/tcp   # Connection port for DB2 instance DB2
  db2iDB2         50001/tcp   # Interrupt  port for DB2 instance DB2
  ```

  Since OS/2 DB2 V1 Servers do not support TCP/IP, there may not be any change to the TCP/IP services file, unless you checked in the *Customize Communication Protocols* screen that the DB2 and Database Administration Server instances were set up for TCP/IP.

- **DB2 Registry**

  DB2 UDB V5 writes DB2 registry information into several flat text files, rather than using any OS/2 *.ini* files. These profile registry files exist for DB2, the DB2 Global Profile Registry, and Instance Profile Registries for the default DB2 instance and the default DB2 Database Administration Server instance.

  – The DB2 Profiles Registry is located in:

    ```
    sqllibprofiles.reg
    ```

This file contains the names of all instances for which DB2 has profile registry information. Expected default contents include:

```
DB2
DB2DAS00
```

– The DB2 Global Profile Registry is located in:

```
sqllib\profiles.reg
```

Note that DB2SYSTEM contains the system name you typed in Figure 146 on page 231. Expected default contents include:

```
DB2SYSTEM=OS/2DB2
DB2SERVICESTPINSTANCE=DB2
DB2INSTDEF=DB2
DB2COMM=NPIPE,TCP/IP
DB2CHKPTR=ON
DB2ADMINSERVER=DB2DAS00
```

– The DB2 Instance Profile Registry for the DB2 instance is located in:

```
sqllib\db2profile.env
```

Expected default contents include:

```
DB2COMM='TCP/IP'
```

– The DB2 Instance Profile Registry for the Database Administration Server instance is located in:

```
sqllib\db2das00profile.env
```

Expected default contents include:

```
DB2COMM='TCP/IP'
```

5.5 DB2 Server Post-Migration

This section applies to migration scenarios involving:

- Windows NT DB2 V2 Servers
- OS/2 DB2 V2 Servers
- OS/2 DB2 V1 Servers

There are several optional activities you may wish to undertake following database migration:

- **Unique index conversion to DB2 Universal Database Version 5 semantics**

  Version 5 of DB2 supports deferred unique constraint checking until end of statement. This can result in correct processing of multiple row updates, that in previous releases of DB2, returned an error because the updates temporarily created duplicate values in the transient state. Deferred unique constraint checking will guarantee that updates that result in a table with only unique keys (for example, key = key + 1) will succeed regardless of the order of the data.

  Note: This change only applies for unique indexes that are created in DB2 UDB V5.

  All unique indexes in a migrated database do not automatically migrate to Version 5 semantics during database migration because of the following reasons:

– Converting unique indexes is a very time-consuming operation. You may have applications that depend on the previous version's unique index semantics. You may wish to manage the staged conversion of unique indexes on your own schedule, when needed.

– All existing applications will continue to work even if the unique indexes are not converted to Version 5 semantics. You have to convert unique indexes to Version 5 semantics only if support for deferred uniqueness checking is required.

To convert unique indexes, you need to perform the following steps:

1. Log on with a user ID that has SYSADM authority.
2. Issue the db2start command.
3. Run the db2uiddl command against your migrated database. The syntax of this command is as follows:

   ```
   db2uiddl -d <database-name> [<-u table-schema>] [<-t table-name>]
   [<-f filename>] [<-h help>]
   ```

   where:

– <database-name> is the name of the database to be queried.

– <-u table-schema> is optional, and can be used to specify the schema (creator user ID) of the tables that are to be processed. The default action is to process tables created by all user IDs.

– <-t table-name> is optional, and can be used to specify the name of a table that is to be processed. The default action is to process all tables.

– <-f filename> is optional, and can be used to specify the name of a file to which output is to be written. The default action is to write output to standard output.

– <-h help> can be used to display help information. When this option is specified, all other options are ignored, and only the help information is displayed.

Note: This tool was not designed to handle certain types of names. If a specific table name or table schema is a delimited identifier containing lower case characters, special characters, or blanks, it is preferable to request processing of all tables or schemas. The resulting output can be edited.

The db2uiddl command searches the database catalog tables and generates all the CREATE UNIQUE INDEX statements for user tables in an output file.

1. Review the output generated from the db2uiddl command, and make changes, if needed. Comments in the output will flag any situations that require your attention.
2. Execute the file as a DB2 Command Line Processor command file, using a command similar to the following:

   ```
   db2 -tvf filename
   ```

   where filename is output file from db2uiddl.

DB2 interprets the recreation of an already-existing unique index to signal that the index is ready to be converted to Version 5 semantics.

• Update Statistics

When database migration is completed, the old statistics, used to optimize query performance, are retained in the catalogs. However, Version 5 of DB2 has statistics that are modified or do not exist in the previous version. To take advantage of these, you may wish to issue the runstats command on tables, particularly those tables that are critical to the performance of your SQL queries.

Refer to the *DB2 UDB V5 Command Reference* for the syntax of the runstats command. For details on the statistics, refer to the *DB2 UDB V5 Administration Guide*.

- Rebind Packages

  During database migration, all existing packages are invalidated during catalog migration. After the migration, each package is rebuilt when it is used for the first time by the Version 5 database manager.

  However, for better performance, it is recommended that you run the db2rbind command to rebuild all packages stored in the database, after database migration is complete. The following gives an example using sample as our database:

  ```
  db2rbind sample -l /tmp/db2rbind.log
  ```

  Refer to the *DB2 UDB V5 Command Reference* for more information about this command.

- Update database and database manager configuration

  Some of the database configuration parameters are changed to Version 5 defaults or to other values during database migration. The same is true for database manager configuration parameters which may have changed, during instance migration, to Version 5 defaults or to other values. For better performance, you may wish to tune your database and database manager configuration parameters to take advantage of Version 5 enhancements. Refer to the *DB2 UDB V5 Command Reference* for the syntax of updating database and database manager configuration.

  The following database manager configuration parameters are changed to Version 5 defaults:

  - Database configuration release level

    This is changed to 0x0800.

  - Sort Heap threshold (SHEAPTHRES)

    If the migrating database configuration file has this parameter at a value which is less than the Version 5 Default, the parameter is reset to its Version 5 default value of 10000 x 4 K pages.

  - Backup buffer (BACKBUFSZ)

    If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to its Version 5 default value of 1024 x 4 K pages.

  - Restore Buffer (RESTBUFSZ)

    If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to the default value of 1024 x 4 K pages.

  - Number of concurrent databases running against the database manager (NUMDB)

If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to its Version 5 default value, as follows:

Type of Database	Number
Database server with local and remote clients	8
Database server with local clients	3

- Transaction Manager database name (TM_DATABASE)

  If the migrating database configuration file has this parameter set to NULL, the parameter is reset to 1ST_CONNECT (1ST_CONN).

- Query heap size (QUERY_HEAP_SZ)

  If the migrating database configuration file has this parameter at a value less than the Application Support Layer heap size (ASLEHAPSZ) of the same file, the parameter is reset to ASLHEAPSZ + 1. The default value is 1000 x 4 K pages.

- Maximum number of idle agents (MAX_IDLEAGENTS)

  If the migrating database configuration file has this parameter greater than the Maximum Simultaneous agents (MAXAGENTS) of the same file, the parameter is reset to MAXAGENTS -1. See the NUM_INITAGENTS and MAXAGENTS parameters for further information.

- TCP/IP Service Name (SVCENAME)

  If the service name in the TCP/IP services file is changed, the SVCENAME parameter is correspondingly changed. For example, from db2c to db2cDB2.

The following database configuration parameters are changed to Version 5 defaults:

- Database configuration release level

  This is changed to 0x0800.

- Database release level

  This is changed to 0x0800.

- Application control heap size (APP_CTL_HEAP_SZ)

  If the migrating database configuration file has this parameter at a value which is less than the Version 5 Default, the parameter is reset to its Version 5 default value, as follows:

Type of Database	Number (4 K pages)
Database server with local and remote clients	128
Database server with local clients	64
Partitioned Database server with local and remote clients	256

– Lock list (LOCKLIST)

  For Version 1 DB2 databases, the LOCKLIST parameter will be first adjusted to LOCKLIST * 32/ 25. If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to its Version 5 default value, as follows:

Type of Database	Number (4 K pages)
Database server with local and remote clients	50
Database server with local clients	25

– Database heap (DBHEAP)

  For Version 1 DB2 databases, the database heap size will first be adjusted to DBHEAP * 16. If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to its Version 5 default value, as follows:

Type of Database	Number (4 K pages)
Database server with local and remote clients	600
Database server with local clients	300

– Default log space (LOGFILSIZ)

  The database migration process will attempt to increase the LOGFILSIZ value, if the log file related parameters have a total LOGFILSIZ that is less than the default LOGFILSIZ value of 250 x 4 K pages.

– Application Heap size (APPLHEAPSZ)

  For Version 1 DB2 databases, application heap size will be first adjusted to APPLHEAPSZ * 16. If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to its Version 5 default value, as follows:

Type of Database	Number (4 K pages)
Database server with local and remote clients	128
Database server with local clients	64

– Sort Heap (SORTHEAP)

  For Version 1 DB2 databases, the value of the SORTHEAP parameter will be adjusted to SORTHEAP * 16.

– Statement Heap Size (STMHEAP)

If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to the default value of 2048 x 4 K pages.

- Recovery Range and Soft Checkpoint Interval (SOFTMAX)

  If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to the default value of 100.

- Package Cache Size (PCKCACHESZ)

  This is always reset to the Version 5 default, of -1, which means the value is 8 * MAXAPPLS, unless 8 * maxappls is less than 32, in which case the default of -1 will set PCKCACHESZ to 32.

- Migrate Explain Tables.

  If you are not using explain tables in DB2 Version 2, skip this task.

  Version 5 of DB2 has added several new columns to the explain tables. These columns provide for the capture of:

  - Data for new SQL features added in Version 5.
  - More detailed access plan information.

  While the explain function in Version 5 will continue to work with explain tables created for Version 2, the new Version 5 data will not be captured in them.

  For better performance of SQL statements, we recommend that the Version 2 explain tables be dropped and new explain tables be created; see the SQL Reference and the Administration Guide for details on creating new explain tables. If, however, there are Version 2 explain tables that you need for ongoing comparison, you can use the EXPLMIG.DLL script to migrate them.

  To migrate the explain tables in a database that has been migrated to Version 5, connect to the database and run the following command from the sqllibmisc directory:

  ```
  db2 -tf sqllibmiscEXPLMIG.DLL
  ```

  The explain tables belonging to the user ID that is used to connect to the database will be migrated. To migrate explain tables belonging to another user, connect to the database with that user ID and run the command.

5.6 DB2 Client Migration

The DB2 Client migration process is a relatively straightforward task, compared with the DB2 Server migration process. Read the following sections in the list order below:

1. 5.6.1, “DB2 Client Pre-Migration” on page 250.
2. Choose one of the following sections, based on your operating system:
   - 5.6.2, “Windows NT DB2 Client Migration” on page 250.
   - 5.6.3, “OS/2 DB2 Client Migration” on page 256.

Note: If your DB2 Client system also has local databases configured, it is recommended you follow the DB2 Server migration process. Refer to the introduction at Chapter 5, “DB2 UDB V5 Migration on OS/2 and Windows NT” on page 199, and follow the steps described for DB2 Server migration.

5.6.1 DB2 Client Pre-Migration

Since this is a Client Application Enabler (CAE) only installation, there are a few pre-migration checks to be aware of. These include the following:

- Migration Restrictions: You should be aware that Version 5 clients are currently not supported by Version 1 servers, with the notable exception of DB2 Parallel Edition V1.2.
- Storage Requirements: As part of the DB2 UDB V5 installation, database and node catalogs are automatically migrated. Space is required for both the old and new catalogs. Hence, you should make available at least two times the amount of disk space as the database catalog currently occupies.
- DB2 sessions: Ensure that no DB2 sessions are active.

5.6.2 Windows NT DB2 Client Migration

After the pre-migration checks, you are ready to perform the migration process. This involves the installation of DB2 UDB V5, which also automatically migrates all database and node catalogs.

Note: If you need to restart the installation for any reason, you may wish to clean up the TCP/IP services file located in winntsystem32driversetc. DB2 UDB V5 adds db2cDB2 and db2iDB2 services for the DB2 instance TCP/IP communications protocol. These usually are at port 50000 and 50001, respectively. However, if you restart the installation, and these entries in the services file already exist, the DB2 UDB V5 installation will append db2cDB20 and db2iDB20 at ports 50002 and 50003, respectively (if these already exist, db2cDB21 at 50004, and db2iDB21 at 50005 and so on). Delete these entries if you see them in the services file, before restarting the DB2 UDB V5 installation.

5.6.2.1 Terminate DB2 Sessions

Terminate all running DB2 sessions. If necessary, bring up the Windows NT Task Manager, and explicitly end any DB2-related tasks on the local system. To obtain the NT Task Manager screen, you can do either of the following:

- Right-click with the mouse button on the Task Bar, and select the **Task Manager** choice when the menu choices appear (Figure 157).

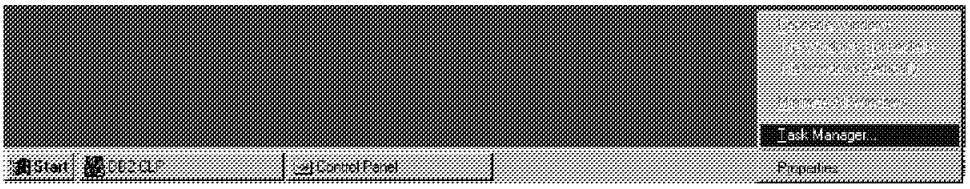


Figure 157. Windows NT Task Bar

- Press the **Ctrl-Alt-Del** key combination and select **Task Manager** from the resulting pop-up screen (Figure 158 on page 251).

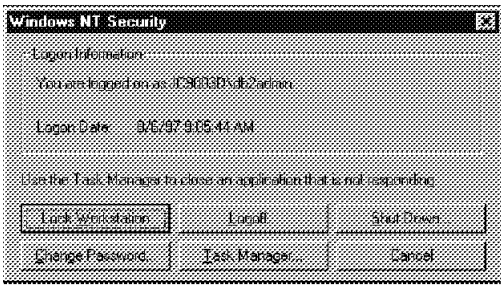


Figure 158. Windows NT Ctrl-Alt-Del

- Press the **Ctrl-Shift-Esc** key combination, which displays the Task Manager screen as in Figure 159.

  At the Task Manager screen, you can check for any DB2 processes and end any that are still running (Figure 159). Choose the Applications tab, then select the task in question, and click on the **End Task** button to end that task.

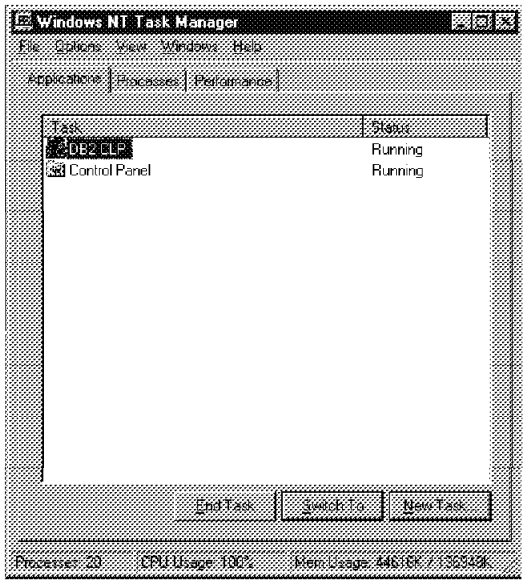


Figure 159. Windows NT Task Manager

5.6.2.2 Start the DB2 UDB V5 Installation

To start the installation for DB2 UDB V5, execute the command <drive>setup.exe, where <drive> is the drive letter of CD-ROM drive containing the DB2 V5 for Windows NT CD-ROM.

If the DB2 UDB V5 installation program detects that there are any DB2 applications still active, the following screen will pop-up (see Figure 160 on page 252). DB2 UDB V5 installation requires that any and all DB2 applications be terminated before the installation can continue.

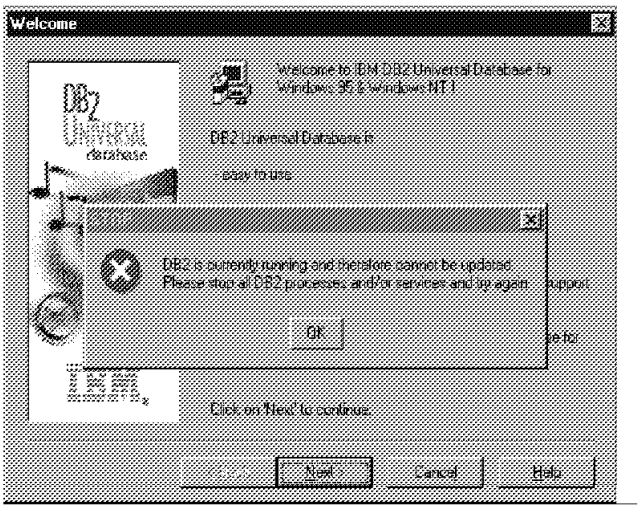


Figure 160. Windows NT DB2 V5 CAE DB2 is Currently Running

Click on the **OK** button to remove this screen. The installation of DB2 UDB V5 will terminate, allowing you to check for any running DB2 applications, including any CLP sessions. Terminate all the DB2 sessions and restart the DB2 UDB V5 installation.

5.6.2.3 Select Products

After space requirements are calculated, you are presented with the *Select Products* screen, listing a choice of DB2 products to install (Figure 161). Since we are migrating a CAE installation, select the DB2 Client Application Enabler option and click on the **Next** button to continue.

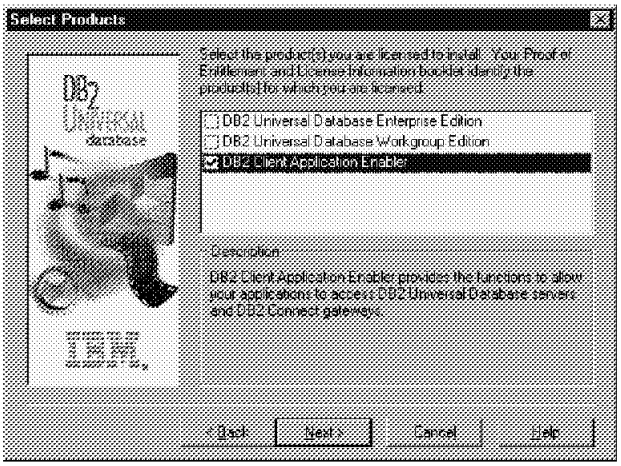


Figure 161. Windows NT DB2 V5 CAE Select Products Screen

An additional feature for the DB2 UDB V5 CAE installation is the option to install support for remote administration. The *Enable Remote Administration* screen (Figure 162 on page 253) allows you to choose this option. Select or deselect this option, and click on the **Next** button to continue:

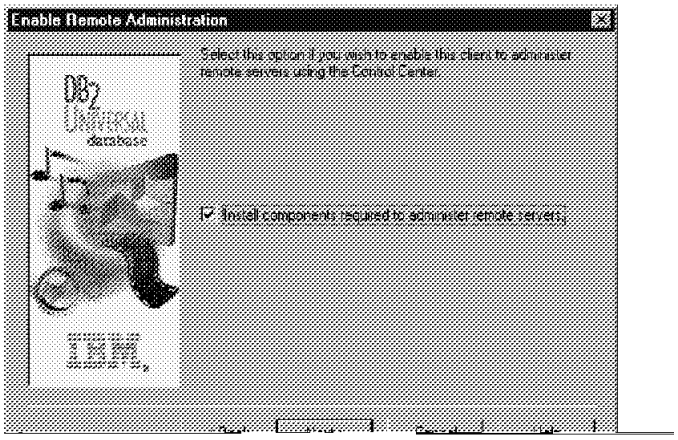


Figure 162. Windows NT DB2 V5 CAE Enable Remote Administration

5.6.2.4 Select Installation Type

The next screen, Select Installation Type (Figure 163) allows you to choose an installation type. In this example, we will choose Custom.

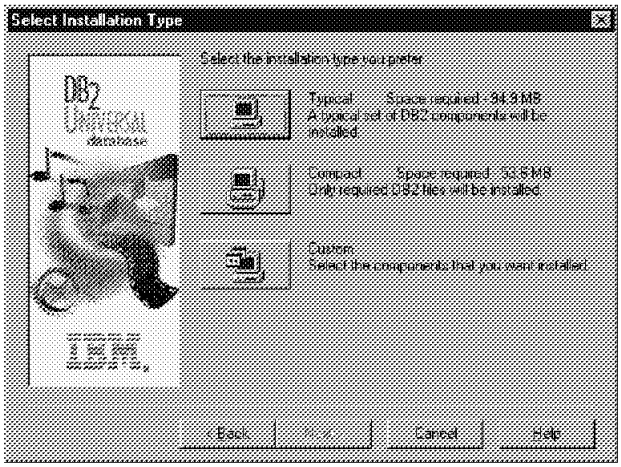


Figure 163. Windows NT DB2 V5 CAE Selection Installation Type

5.6.2.5 Select DB2 Components

The Select DB2 Components screen allows you to customize your choice of installation for the various DB2 components. You can also change the drive and base path information for the default sqllib directory. The Space Required portion of this screen will reflect the amount of disk space required, for the components you have chosen to install.

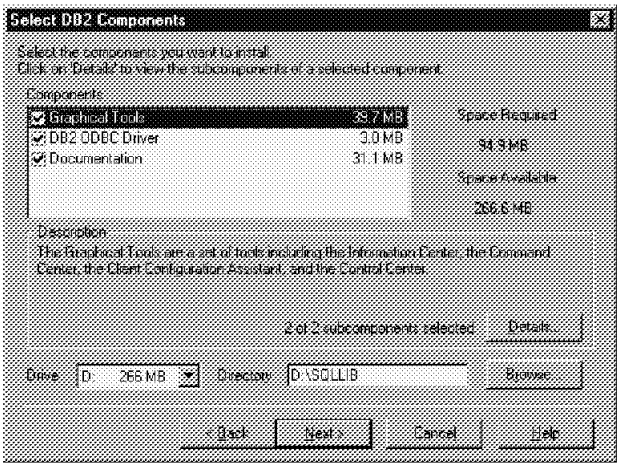


Figure 164. Windows NT DB2 V5 CAE Select DB2 Components

Ensure you have enough space as recommended by the installation program. If the amount of space available is less than the space required, you will receive a warning that there may not be enough space, as in Figure 165.

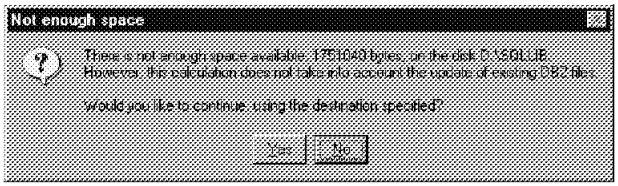


Figure 165. Windows NT DB2 V5 CAE Not Enough Space

Note: You may encounter a problem with the installation program if you run out of disk space during installation. This will be covered later in the section when the actual installation is started.

5.6.2.6 DB2 Start Options

As part of the Start Options for Windows NT DB2 UDB V5 (Figure 166 on page 255), you can choose for the DB2 UDB V5 Control Center to be automatically started at boot time. Choose the required option, and continue by clicking on the **Next** button.

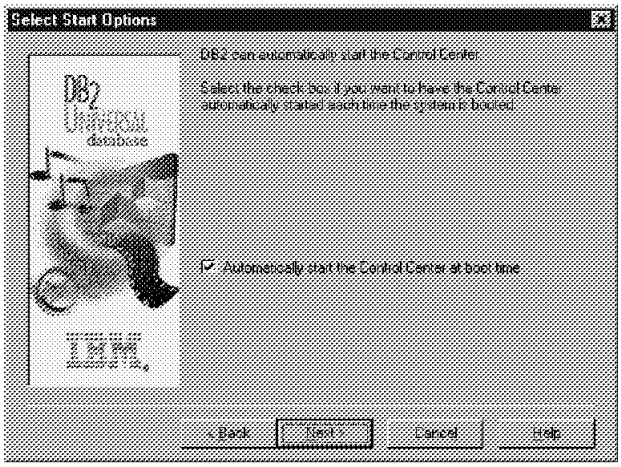


Figure 166. Windows NT DB2 V5 CAE Select Start Option

5.6.2.7 Start Copying Files

FInally, the last screen, labeled *Start Copying Files* (Figure 167), allows you to check the current settings for the DB2 UDB V5 CAE installation, before clicking on the **Install** button to start installing the UDB files onto disk. This is the last point at which you can click on **Back** to change any settings, or the **Cancel** button to cancel the installation.

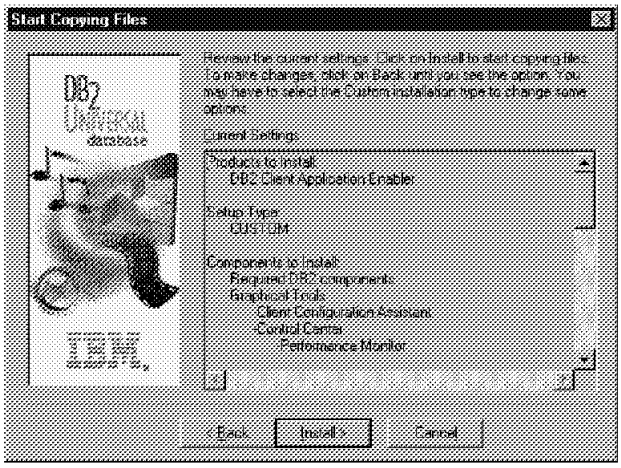


Figure 167. Windows NT DB2 V5 CAE Start Copying Files

Note: If you run out of space during the installation, you will see a pop-up panel as per Figure 168 on page 256.

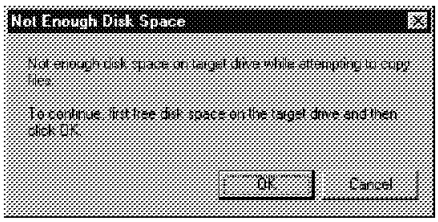


Figure 168. Windows NT DB2 V5 CAE Not Enough Disk Space

If you get this message, you may not be able to cancel the installation by clicking on the **Cancel** button. If you free up more disk space and then click on the **OK** button the installation may hang, and in this case the subsequent **Cancel** button will not work. You will have to terminate the application using the task manager, as outlined before, and kill the DB2 UDB for NT setup task.

It is recommended that you free up more space, then restart the installation. Note that the previous version of DB2 will be in an inconsistent and uncertain state, since the DB2 V5 installation will have overwritten certain parts of the code and support libraries.

Once the installation completes with no problems, restart the Windows NT system to allow all changes to take effect.

5.6.3 OS/2 DB2 Client Migration

After the pre-migration checks, you are ready to perform the migration process. This involves the installation of DB2 UDB V5, which also automatically migrates all database and node catalogs.

Note: If you need to restart installation for any reason, you may wish to clean up the TCP/IP services file located in mptnetc. DB2 UDB V5 adds in db2cDB2 and db2iDB2 services for the DB2 TCP/IP communications protocol. These usually are at ports 50000 and 50001 respectively. However, if you restart the installation, and these entries in the services file already exist, the DB2 UDB V5 installation will append db2cDB20 and db2iDB20 at ports 50002 and 50003 respectively (if these already exist, db2cDB21 at 50004, and db2iDB21 at 50005 and so on). Delete these entries if you see them in the services file, before restarting the DB2 UDB V5 installation.

5.6.3.1 Start the DB2 UDB V5 Installation

To start the DB2 V5 installation, execute the command `<drive>:<language>installinstall.exe`, where:

- `<drive>` is the drive letter of your CD-ROM drive containing the DB2 V5 for OS/2 CD-ROM
- `<language>` is the two-character file name that represents your language (for example, `en` for English)

Since we are migrating a CAE installation, from the screen as in Figure 169 on page 257, choose DB2 Client Application Enabler as the product to install, and click on the **Continue** button.

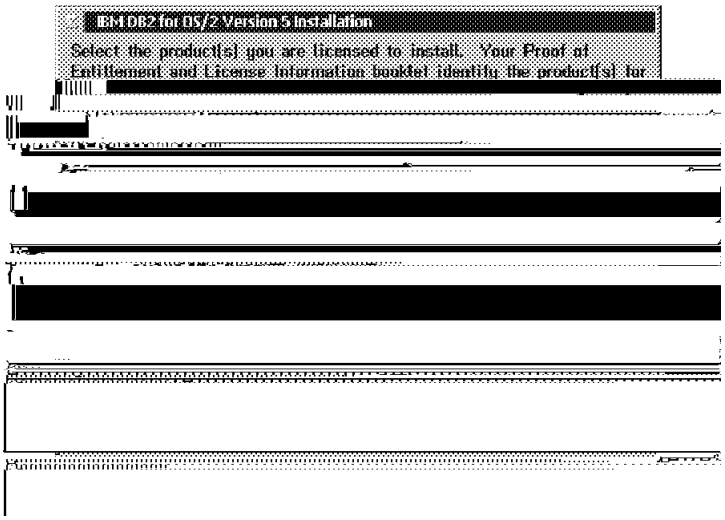


Figure 169. OS/2 DB2 V5 CAE Product Selection

The DB2 UDB V5 installation must update your CONFIG.SYS file (Figure 170). A backup is saved, called CONFIG.000 (if this already exists, CONFIG.001 and so on). Leave this option ticked and click on the **OK** button to continue, or click on the **Cancel** button to cancel the installation.

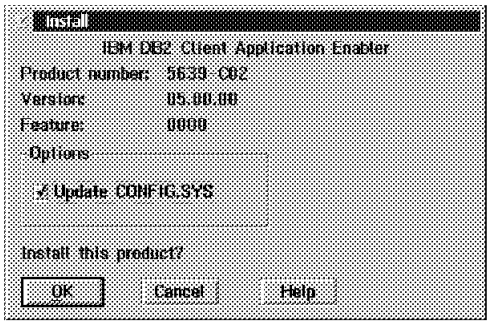


Figure 170. OS/2 DB2 UDB V5 CAE Update CONFIG.SYS

5.6.3.2 Install Directories

The next screen is the *Install - directories* screen, which allows you to customize which components to install change the drive and base path information for the default *sqllib* directory. Choose your components, and click on the *Disk space...* button to confirm you have enough disk space to perform the installation:

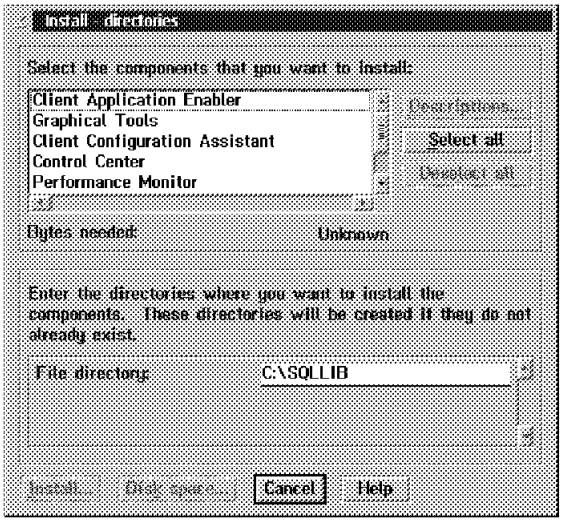


Figure 171. OS/2 DB2 UDB V5 CAE Install - directories

Click on the **Install** button to continue the installation. The DB2 UDB V5 CAE installation will automatically detect any communications protocols used by your system. For example, Figure 172 shows it has detected NetBIOS. You are then given the choice to enable NetBIOS support in DB2 UDB V5 CAE, and to customize the communications protocol if required.

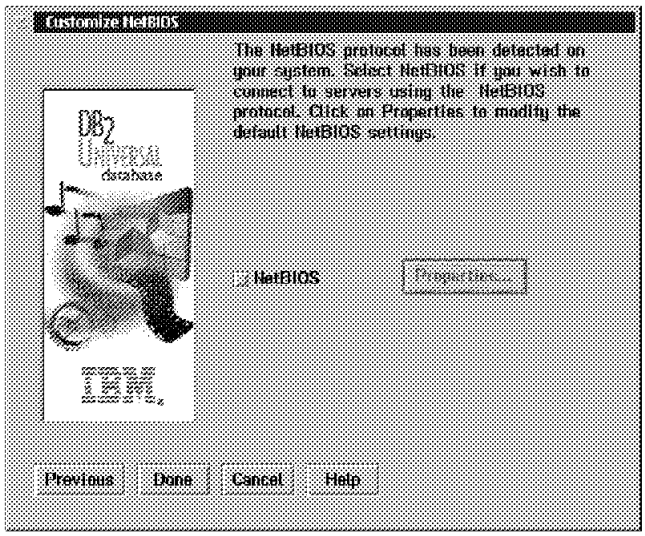


Figure 172. OS/2 DB2 UDB V5 CAE Customize NetBios

At this point, clicking on the **Done** button will start the actual DB2 UDB V5 CAE installation.

If you do run out of disk space, you should receive a pop-up screen like Figure 173 on page 259, which indicates the installation program has run out of disk space to copy the required DB2 UDB V5 files. In this case, free up more disk space, and try again with the **Retry** button.

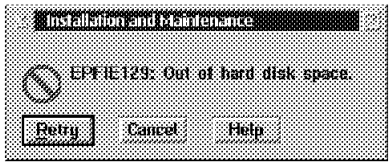


Figure 173. OS/2 DB2 UDB V5 CAE Out of Disk Space

However you may receive a pop-up screen like Figure 174. In this case, it is advisable to check that you do have enough free space in the drive to which you are installing DB2 UDB V5. If not, free up space and try again with the **Retry** button.

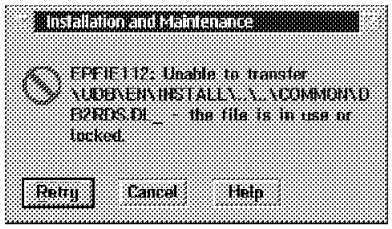


Figure 174. OS/2 DB2 UDB V5 CAE Installation Problem

Chapter 6. DB2 UDB V5 Migration on UNIX

This chapter provides an in depth analysis of migrating DB2 V1 and V2 systems to DB2 UDB V5 on UNIX platforms. It covers detailed examples of migration on the AIX V4 platform, migrating from DB2 V1, DB2 V2, and migrating a single partition database to a multi-partition database.

There are many considerations to take into account while planning your migration. The first section discusses how the migration will affect your database, and the planning required before your migration. Read this section before continuing any further:

- 6.1, “Migration Planning and Considerations.”

The next three sections discuss and take you through three different migration scenarios. Choose your scenario, and refer to the appropriate section as follows:

- 6.2, “Migrating DB2 V1/V2 Servers to DB2 UDB V5” on page 266.
- 6.3, “Migrating DB2 V1/V2 Clients to DB2 UDB V5” on page 281.
- 6.4, “Migrating a Single Partition Database to a V5 Multi-Partition Database” on page 286.

There is also a section on Post-Migration, which discusses how the database environment has changed after a migration, and how to take advantage of some of the new features of DB2 UDB V5:

- 6.5, “Post-Migration” on page 298

Note: Migration from an AIX V3 platform is not supported, To migrate your DB2 system on AIX V3, you must upgrade AIX to at least V4.1.4 first.

6.1 Migration Planning and Considerations

This section describes the implications of migrating your database to Version 5, and how it affects the database path, system catalog tables and the environment. It discusses some migration issues to consider such as database authentication, security and storage requirements to ensure a successful migration. It also considers the impact of some incompatibilities between two versions of the product.

6.1.1 Migration Implications on Databases

There is some basic planning and preparation to perform when migrating your database to DB2 UDB V5. The directory where the physical database files (for example, SQL00001) are stored has changed location.

During the migration of a database, the following events occur:

1. The following entities are migrated:
   - Database configuration file
   - Database system catalog tables
   - Database directories

- Database log file header
- Database index files
- Database data files.

2. The database is relocated to a new database path:

   For DB2 UDB V5, the database path is common across all platforms. The *instance/nodename* path convention used in DB2 Parallel Edition V1 is used. The nodename path is limited to eight characters. All back-level cataloged databases are required to be relocated to the new path, including DB2/PE V1 databases. As an example, assuming */database* is our local database directory, and *inst1* is our instance name, the database path changes as follows.

DB2 VERSION	DATABASE PATH
DB2 V1 / V2	/database/inst1/SQL00001/
UDB V5	/database/inst1/NODE0000/SQL00001

3. **System Catalog Tables are changed as follows:**
   - New columns are added.
   - New tables are created. These are the new tables introduced at DB2 UDB V5:

New System Catalog Tables	
SYSATTRIBUTES	SYSBUFFERPOOLNODES
SYSBUFFERPOOLS	SYSCOLPROPERTIES
SYSHIERARCHIES	SYSNODEGROUPDEF
SYSNODEGROUPS	SYSPARTITIONMAPS
SYSPROCEDURES	SYSPROCPARMS
SYSSCHEMAAUTH	SYSSCHEMATA
SYSTRIGDEP	SYSVIEWS

   - A set of catalog views is migrated, and a set of new catalog views is created, in the SYSCAT schema. A set of updateable catalog views is created in the SYSSTAT schema. If you are migrating from DB2 V1, you will notice SYSCAT and SYSSTAT views introduced since these schema do not exist at DB2 V1. These are the new SYSCAT views introduced at DB2 UDB V5:

New System Catalog Views	
BUFFERPOOLNODES	BUFFERPOOLS
COLAUTH	COLPROPERTIES
NODEGROUPDEF	NODEGROUPS
PARTITIONMAPS	PROCEDURES
PROCPARMS	SCHEMAAUTH
SCHEMATA	

- A set of general purpose scalar functions is kept, and a set of new general purpose scalar functions is created, in the SYSFUN schema. Only the SYSFUN.DIFFERENCE scalar function is dropped and re-created during database migration.

4. A new directory called db2event is created in the database directory.

   This directory did not exist at DB2 V1, and has been relocated under the new directory structure when migrating DB2 V2 to UDB V5. The new location at DB2 UDB V5 is */database/inst1/NODE0000/SQL00001/db2event* where */database* is the database path, and *inst1* is the name of the instance.

5. Buffer pool files are created in the database path.

   The buffer pool files created in the database path are SQLBP.1 and SQLBP.2.

6. A database history file and shadow are created in the database path.

   This is file contains a summary of backup information that can be used if a database must be restored, and it is updated whenever a backup, restore, or table load operation is performed on the database. A summary of backup information is also kept for backup and restore operations on a table space. This file exists as */database/inst1/NODE0000/SQL00001/db2rhist.asc* where */database* is your the database path and *inst1* is the name of the instance.

Note: Migration differences between UNIX and PC platforms. On OS/2 or Windows NT you need to run the *db2ckmig* tool first to ensure that your databases are ready for migration, and then install DB2 UDB V5, and run the `db2 migrate database` command to migrate your databases to V5. The two versions of DB2 cannot co-exist on a system (except via hidden partitions). However, on UNIX platforms, you can install DB2 UDB V5 before or after the Pre-Migration steps, as both versions of DB2 can co-exist.

6.1.2 Migration Considerations

To successfully migrate a database created with a previous version of DB2, you must consider the following:

- Migration Restrictions
- Security and Authorization
- Storage Requirements
- Release-to-Release Incompatibilities

6.1.2.1 Migration Restrictions

There are certain pre-conditions or restrictions that you should be aware of before attempting to migrate your database to DB2 UDB V5.

- Migration is only supported from V1.x or V2.x. Earlier versions of DB2 must be migrated to V1.x or V2.x before attempting to migrate to DB2 UDB V5.
- Migration on the AIX platform is only supported for AIX V4.1.4 and above. Earlier versions of AIX must be upgraded to AIX V4.1.4 before starting the migration of DB2.
- Issuing the migration command from a V5 client to migrate a database on a V5 Server is supported. However, issuing a migration command from earlier versions of DB2 clients to a V5 Server is not supported.
- Migration between platforms is not supported.
- User objects within your database cannot have V5 reserved schema names as object qualifiers. These reserved schema names include: SYSCAT, SYSSTAT, and SYSFUN.
- Database objects with a dependency on the SYSFUN.DIFFERENCE function must be dropped before migrating the database. Objects that might have a dependency on this function include: views, constraints, functions, and triggers.
- You can not migrate a database where there are user-defined distinct types using the names DATALINK or REFERENCE. These objects must be renamed before migrating the database.
- Your database cannot be in one of the following states:
  - Backup pending
  - Roll-forward pending
  - One or more table spaces not in a normal state
  - Transaction inconsistent
- Restore of down-level (V1 or V2) database backups to DB2 UDB V5 is supported but rolling forward of down-level logs is not supported.

6.1.2.2 Security and Authorization

You need SYSADM authority to migrate your database.

If migrating from DB2 V1, you should know that a database cannot be cataloged with a mix of authentication types. The authentication type of the instance in DB2 UDB V5 is defined in the database manager configuration file. If mixed types are detected during migration from Version 1, you can either stop the migration and change the directories or continue with the migration. If migration continues, all the authentication types are changed to blank, and the database uses the authentication type specified in the instance. The default authentication type created when creating an instance at DB2 UDB V5 is SERVER.

To use two databases with different authentication types, a new instance must be created for one of the databases. The database should be backed up and restored to a new database under the new instance. It can then be dropped under the old instance and migration can then be run.

Beginning with DB2 V2, users and groups are differentiated in SQL statements and the system catalog. As a result, if a user and a group have the same name

in the previous version, the authority and privileges granted to the group must be explicitly re-granted after migration.

During migration, the authorization catalog tables, SYSCAT.DBAUTH, SYSCAT.INDEXAUTH, SYSCAT.PLANAUTH, and SYSCAT.TABAUTH, are checked to determine if existing privileges are for users or groups, and the GRANTEETYPE is defined as follows:

- If the name in the GRANTEE column is a user or is not defined; the GRANTEETYPE is defined as U.
- If the name in the GRANTEE column is a group; the GRANTEETYPE is defined as G.
- If the name in the GRANTEE column is both a user and a group; the GRANTEETYPE is defined as U. Privileges must then be explicitly granted to the group.

6.1.2.3 Storage Requirements

Space is required for both the old and new catalogs during the migration, and the amount of disk space required will vary depending on the complexity of the database as well as the number and size of the database objects. These objects include all tables and views. You should make available at least two times the amount of disk space the database catalog currently occupies. If there is not enough disk space in the local database directory, the instance migration will fail. The only way of recovering from this failure is to drop and recreate the instance at DB2 V1 or DB2 V2, and then restore from a backup copy of your database. Recataloging the database back into the instance will not work since the database directories will also have been partially updated. The only solution is to restore from a backup.

You should also consider increasing the database configuration parameters associated with the log files. You should increase logfilsiz, logprimary, and logsecond to prevent the space for these files from running out. If log space is completely used, you will receive a SQLCODE of SQL1704N with a reason code of 3. If this happens, increase the log space parameters and re-issue the database migration command.

6.1.2.4 Release-to-Release Incompatibilities

To successfully migrate a database, you should consider the impact of the incompatibilities between the two versions of the product. The following incompatibilities deserve special attention before you begin your migration:

- **View Definitions**

  If a view defined in DB2 V1 involves ″SELECT *″, the view may be unusable after migration. If the view is unusable, attempts to use it, directly or indirectly, will result in SQLCODE -158. The view must be dropped and recreated in order to avoid this error. If fewer than the current number of columns in the SELECT * table is desired, the recreated view must specify the needed columns.

- **Configuration Parameters**

  – Configuration parameter values are preserved during the migration of the database, with the exception of the following parameters:

  – Application Heap Size (APPLHEAPSZ)

  – Package Cache Size (PCKCACHESZ)

- Maximum Storage for Lock List (LOCKLIST)
- Recovery Range and Soft Checkpoint Interval (SOFTMAX)

For these parameters, the use of the associated heap has changed significantly in DB2 UDB V5.

- APPLEHEAPSZ is reset to the DB2 UDB V5 default value if the current value is less than the Version 5 default value.
- PCKCACHESZ is always reset to the DB2 UDB V5 default value.

For locklist, the DB2 V1 value is multiplied by a factor of 32/25. This computed value will be used as the DB2 UDB V5 parameter value, if this value is greater than the Version 5 default. Otherwise, the Version 5 default will be used.

6.2 Migrating DB2 V1/V2 Servers to DB2 UDB V5

Instances were introduced in DB2 for AIX V1 to facilitate separate and unique database manager environments. This allowed you to have several database manager environments, with alternative configuration parameters or authentication, all on the same processing platform. When considering migration, you may want to retain these separate instances under DB2 UDB V5. So, in the UNIX environment, the migration is comprised of two distinct, but interrelated, phases: Instance Migration and Database Migration.

6.2.1, “DB2 Servers: Pre-Migration” on page 267 prepares your instance for migration, takes you thr ough installing DB2 UDB V5, and checks your databases within an instance with the utility db2ckmig. It also looks at some problems that you may come across while checking your instance, and how to resolve these.

6.2.2, “DB2 Servers: Migrating Instances” on page 278 takes you through the initial phase of migration, which is migrating your *instance* environment to DB2 UDB V5. It looks at some problems that you may come across during migration, how to overcome these problems, and how to check that your instance migration has worked.

6.2.3, “DB2 Servers: Migrating Databases” on page 281 takes you through the next phase of migration, which is migrating your databases within your instance environment to DB2 UDB V5. It looks at some problems that you may experience, how to overcome them, and then how to check that the database migration has worked.

Here is a summary of the steps that you need to follow to ensure a smooth migration of your DB2 V1 or V2 system:

- Prepare your instance for migration.
- Install DB2 UDB V5.
- Create a Database Administration Server instance (optional).
- Create a user ID for fenced User Defined Functions (optional).
- Run db2ckmig to check the databases that you wish to migrate.
- Run db2imigr to migrate an instance.
- Run the `db2 migrate database` command to migrate the databases within an instance.

6.2.1 DB2 Servers: Pre-Migration

DB2 UDB V5 provides the utility db2ckmig, to allow you verify whether cataloged databases within an instance can be migrated. When migrating an instance (using the db2imigr utility), the db2ckmig utility will be executed.

In this section we will:

- Prepare your instance for migration.
- Install DB2 UDB V5.
- Create the Database Administration Server instance user ID and a fenced user ID.
- Detail the steps required to verify and correct the database state.

It is advisable to perform these pre-migration steps before migrating to DB2 UDB V5. Migration is not a recoverable process. If you do not have a backup of your databases before you attempt the migration, and the migration fails, you will have no way of restoring your database using either DB2 UDB V5 or a previous version of DB2.

Note:

- All databases that you wish to migrate to DB2 UDB V5 must be cataloged, otherwise the migration of those databases will not occur.
- In order for a migration to be successful, all applications and end users must be disconnected from the database being migrated.

6.2.1.1 Preparing the Instance for Migration

Use the following checklist to ensure your instance is ready for migration:

1. Ensure all local databases that you want to migrate are cataloged.

   All databases, local and remote, that are to be migrated, must be cataloged. In a DB2 Command Window, check the database directory with the following command:

   ```
   db2 list database directory
   ```

   If you have any local databases that you do not want to migrate to DB2 UDB V5, then you should uncatalog them before migration.

2. Make a backup copy of all databases.

   Make a backup copy of the database before you start the next procedure since migration is not a recoverable process. If you do not have a backup of your databases from before you attempt to migrate them, and the migration fails, you will have no way of restoring your database using DB2 UDB V5 or a previous version of DB2. For additional information on the BACKUP command, refer to the *DB2 UDB V5 Administration Guide*.

   ```
   db2 backup database sample to dir/dev
   ```

3. Make copies of:

   - Database Manager Configuration
   - Database Configuration for each database
   - Database, node and DCS directories

   Use the following commands where *sample* in the name of the database and *inst1* is the name of the instance:

```
db2 get database manager configuration > dbmcfg.inst1
db2 get database configuration for sample > dbcfg.inst1.sample
db2 list database directory > dbdir.inst1
db2 list node directory > nodedir.inst1
db2 list dcs directory > dcsdir.inst1
```

4. Ensure all applications disconnect from all databases.

   First, make sure all database transactions have been completed. You can obtain a list of all applications using the databases owned by the instance using the following command:

   ```
   db2 list applications
   ```

   You can then force termination of all applications using the following command:

   ```
   db2 force application all
   ```

   Get a list of applications again to ensure that all applications have been terminated, using the following command:

   ```
   db2 list applications
   ```

   Stop all database server processes owned by the DB2 instance using the following command:

   ```
   db2stop
   ```

   Note: You can also try `db2stop -kill`

   Stop the DB2 license daemon with the following command:

   ```
   db2licd end
   ```

   Check that the DB2 license daemon has stopped with the following command:

   ```
   ps -ef | grep db2licd
   ```

   If the license daemon is still running, then it may have been invoked by another connection to a database, possibly by another instance. If this is the case, to stop the license daemon, either log in as that instance user and stop the license daemon again or login as root and invoke the following command, where `PID` is the process id of db2licd:

   ```
   kill -9 PID
   ```

   Stop all command line processor sessions, any user invoked DB2 sessions using the following command:

   ```
   db2 terminate
   ```

   Unload shared libraries with the following command:

   ```
   slibclean
   ```

   Check the state of IPC (Inter-Process Communications) resources owned by DB2 instances:

   ```
   ipcs
   ```

   Remove any IPC resources owned by DB2 instances. For example:

   ```
   ipcrm -m MESSAGEID -q SHAREDMEMID -s SEMAPHOREID
   ```

   where:

   `MESSAGEID` is the ID of a message queue.
   `SHAREMEMID` is the ID of a shared memory block.

SEMAPHOREID is the ID of a semaphore.

Now you are ready to install DB2 UDB V5.

6.2.1.2 Installing DB2 UDB V5

In this section we will install DB2 UDB V5, which contains the db2ckmig and db2imigr utilities that will be used later in the migration process. Unlike the Intel platforms, on UNIX you can have multiple versions of DB2 installed; the version your are migrating from, and the version your are migrating to.

To perform the installation you should use the db2setup tool provided in DB2 UDB V5. As well as installing the products (via installp on AIX), this tool allows you to:

1. Create a new instance, and for that instance:
   - Create a new user ID and group.
   - Create a fenced user ID and group for fenced UDFs.
   - Set up communications.
   - Configure environment variables and DB2 Profile Registry entries.

   Since we will be migrating from an instance that exists in DB2 V1 or V2, the step to create an instance should only be used for new instances that did not exist in the previous version of DB2.
2. Create a Database Administration Server instance, and for that instance:
   - Create a new user ID and group
   - Set up communications.
   - Configure environment variables and DB2 Profile Registry entries.

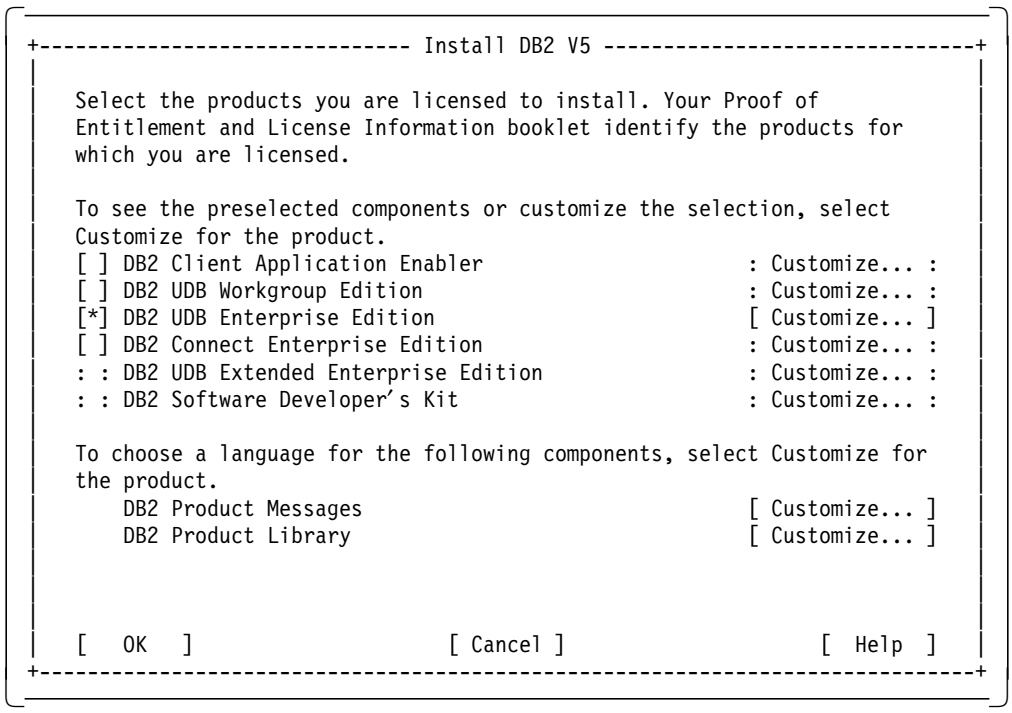
   The step to create a DB2 Administration Server (DAS) instance is new to DB2 UDB V5 and it is recommended that you complete this step. The DAS instance is used to provide services to support client tools that automate the configuration of connections to DB2 databases, and to enable administration of the DB2 Server system from remote clients.

Start the Installation of DB2 UDB V5 using db2setup: To start the installation, log in as root, mount the CD-ROM, and run the db2setup tool that is provided with DB2 UDB V5 from your mount directory:

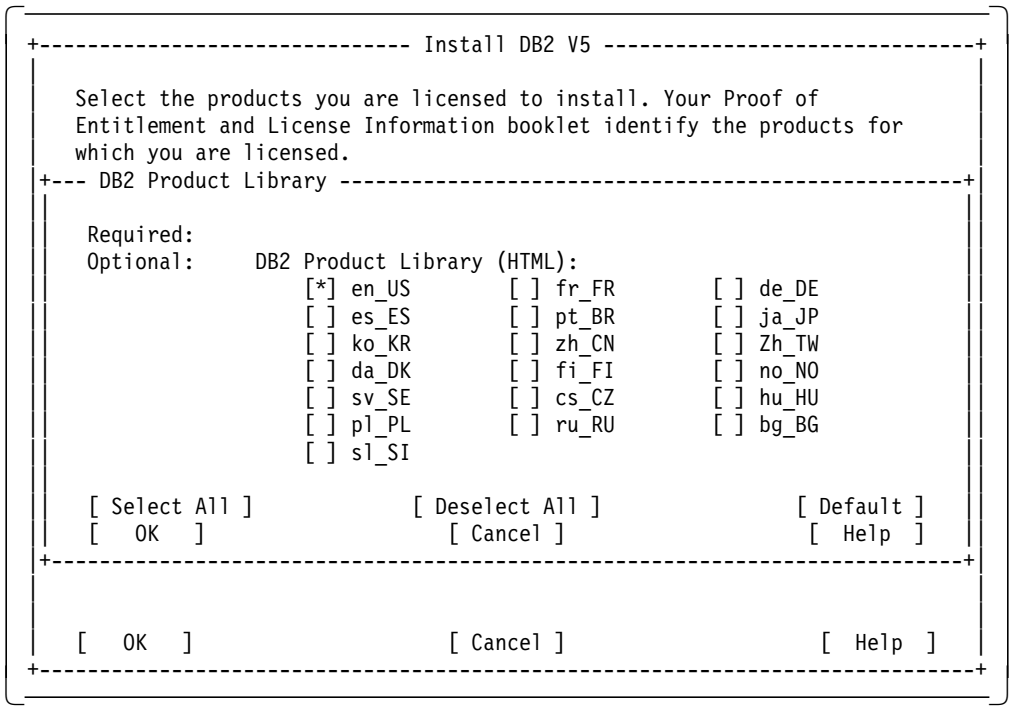
```
./db2setup
```

The following screens take you through an example of an installation using db2setup. The screen shown below prompts you to select the product that you wish to install. In this example we have selected DB2 UDB Enterprise Edition.

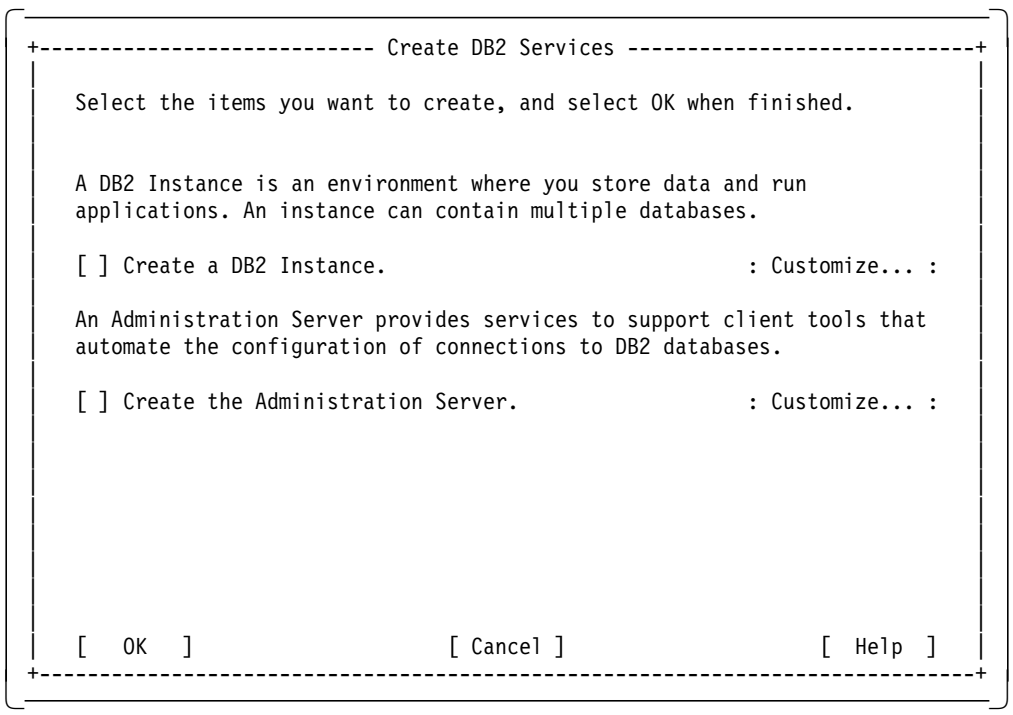
Note: Use the tab key to move to your selection, the space bar to select your choice, and the enter key to activate your selection.



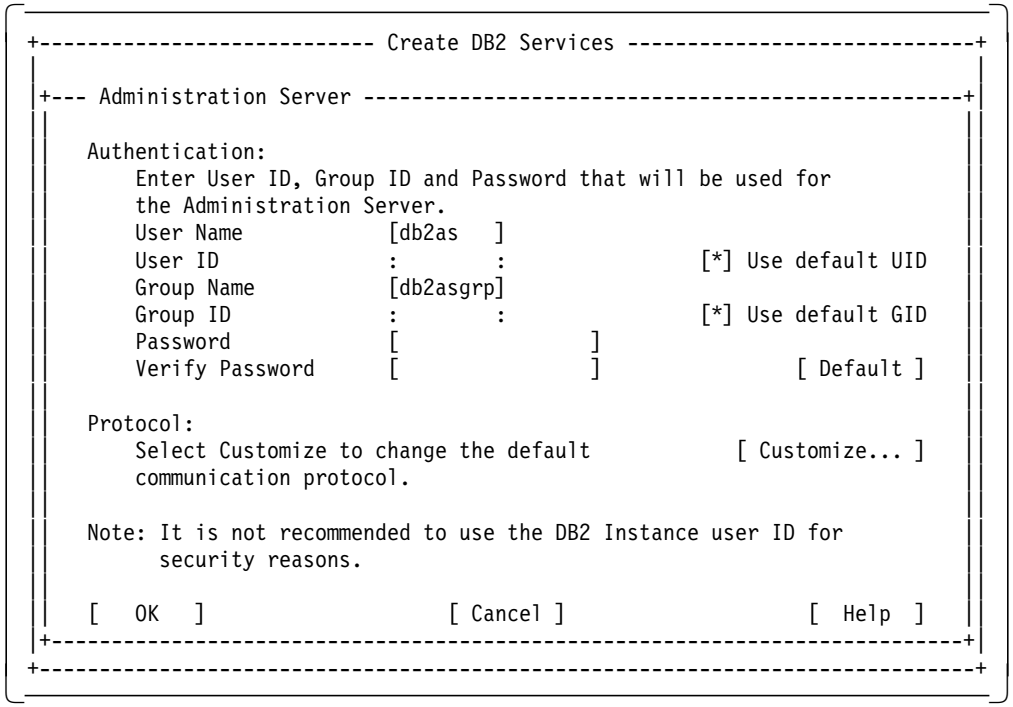
Now select your language for the IBM Product Messages (DB2 error messages) and for the DB2 Product Library (DB2 Books in HTML), as shown in the next screen, and select **OK** to activate the selection.



This will take you to the next screen which gives you the option to create a new DB2 instance and also a Database Administration Server instance.

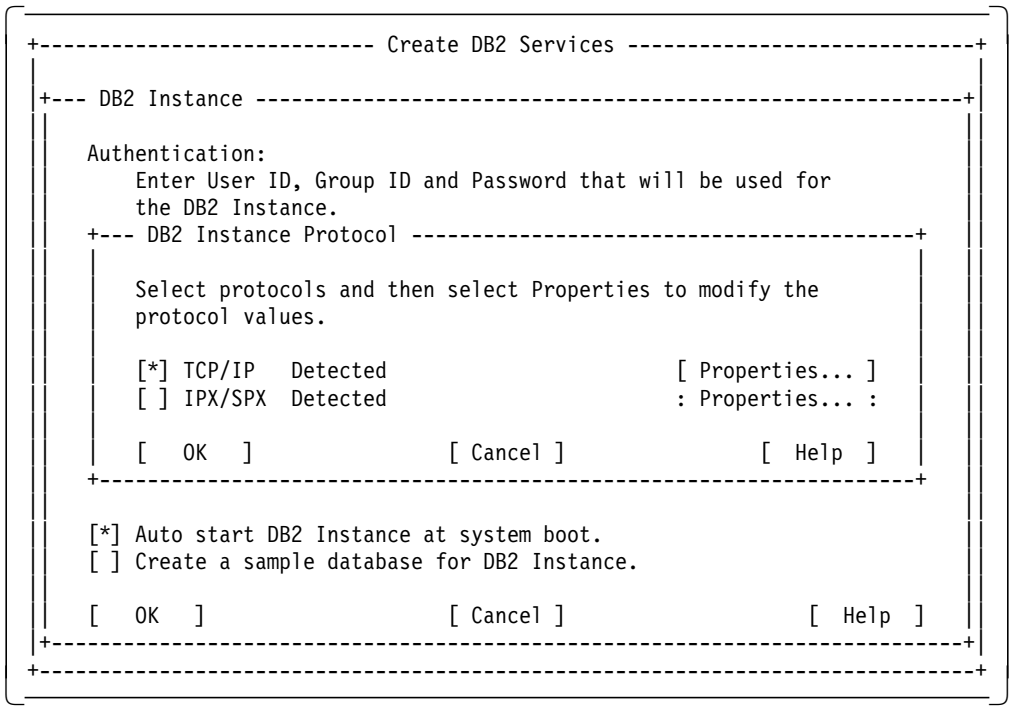


Since we will migrate existing instances, you do not need to create a new DB2 instance. The Administration Server instance is used to provide services to support client tools that automate the configuration of connections to DB2 databases, and to enable administration of the DB2 Server system from remote clients. You should select **Create the Administration Server**, which will display this screen:

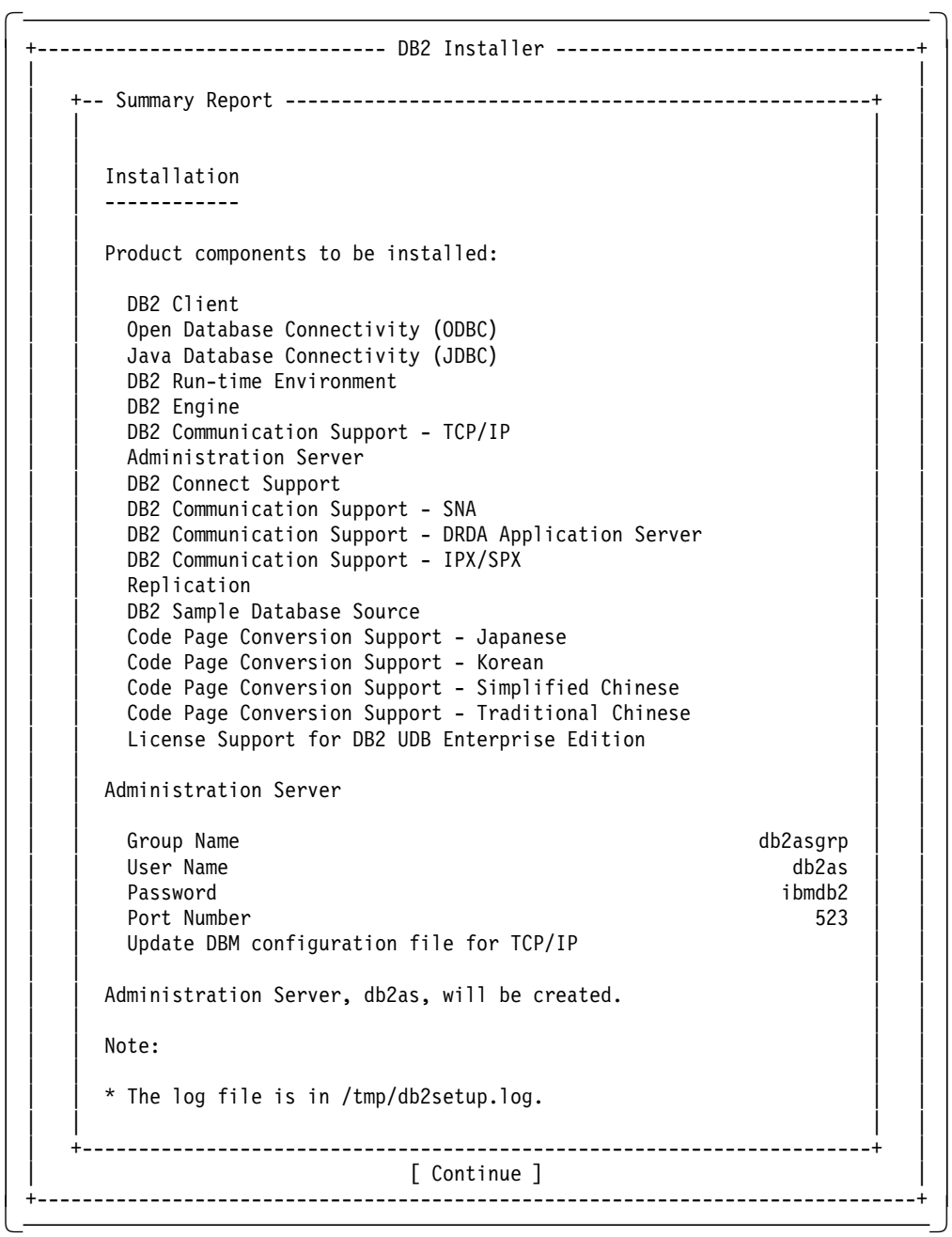


You can either accept the default user db2as and group db2asgrp, or provide your own user and group. You should enter the password for the user, both in the *Password* and *Verify Password* fields. If you do not enter a password, db2setup

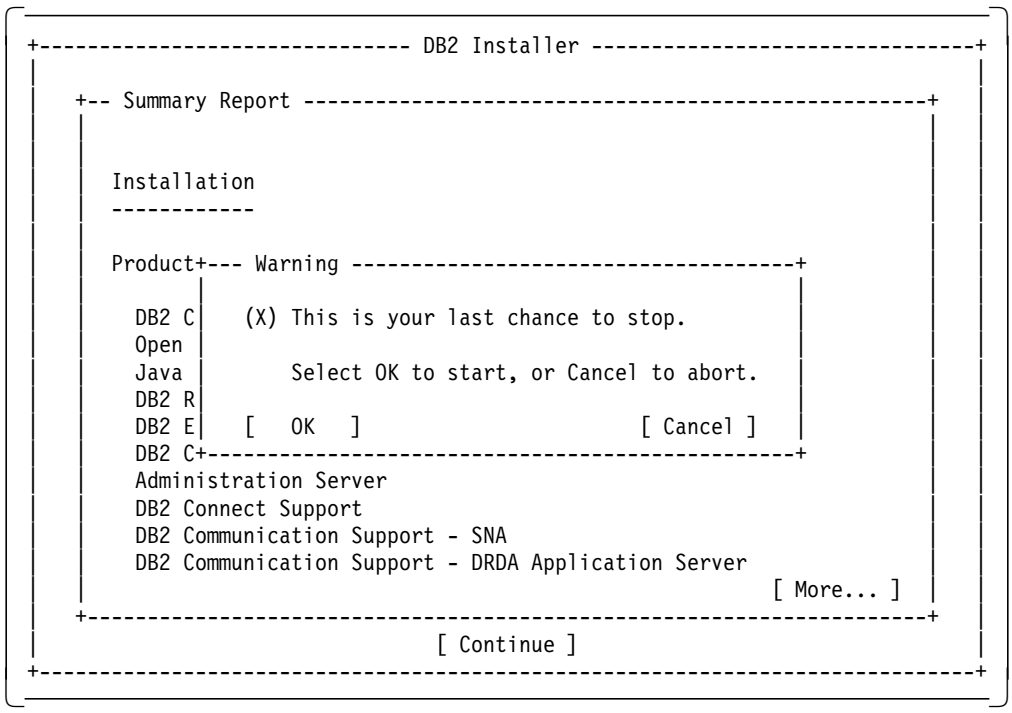
will use ibmdb2. You can choose to customize the communications protocols by selecting **Customize** on the Protocol section, as shown in this screen:



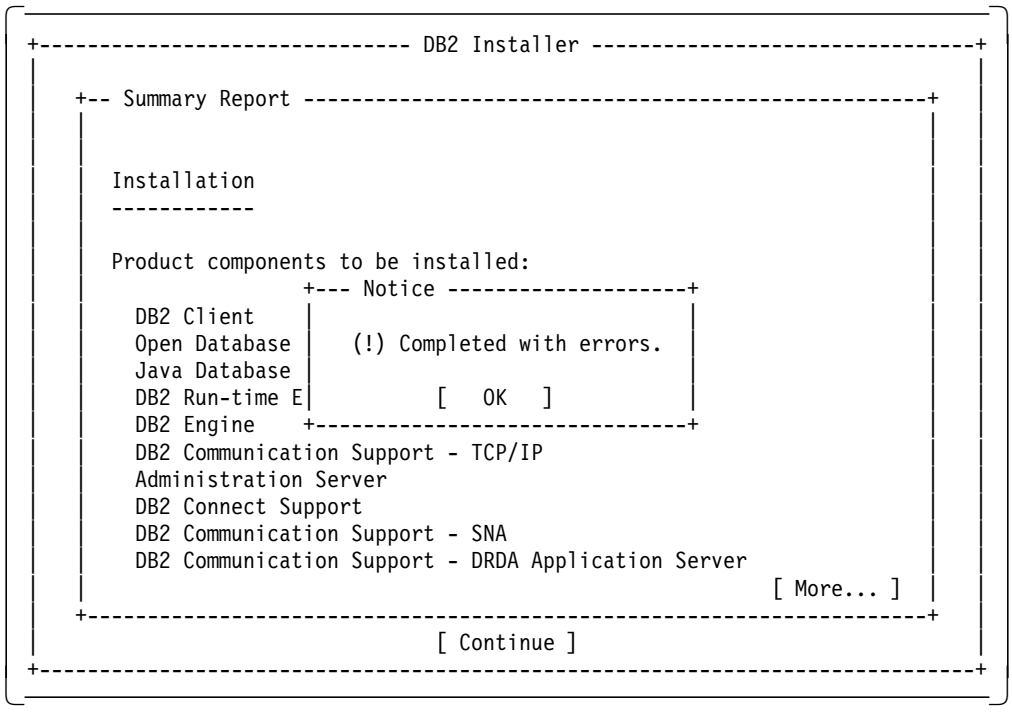
The Summary Report screen appears before the confirmation of the install is requested. The next screen is the result of our example install:



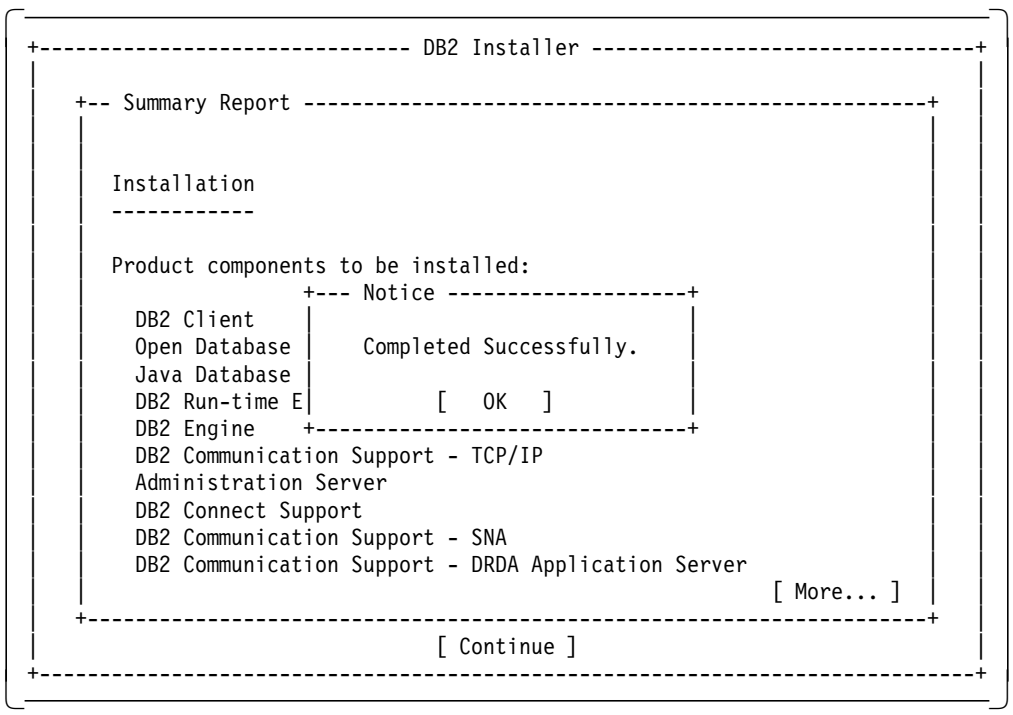
A final warning screen is displayed before the installation is activated:



If any part of db2setup fails, it will complete will the following message. To resolve any problems, look at the log file that db2setup generates in /tmp/db2setup.log.



If db2setup completes successfully, you will see the following message displayed:



Note: You can install DB2 UDB V5 using the standard installation tool (SMIT on AIX). If you choose this method, and want to setup a DAS instance, you should run the following command, where db2as is the DAS instance owner:

```
dasicrt db2as
```

You are now in a position to check and migrate your instance.

6.2.1.3 Using db2ckmig to Verify Your Database

DB2 UDB V5 comes with two utilities db2ckmig and db2imigr, which are used in the migration process. db2ckmig is a utility to check if a given database can be migrated. db2imigr, a utility used to migrate the instance, calls db2ckmig while migrating the instance. We recommend that you run db2ckmig before db2imigr to ensure that you can migrate all the databases in the instance. This section explains how to use db2ckmig to check your databases can be migrated.

db2ckmig checks the instance environment to find out the current DB2 version. It then executes the binary db2ckmig_se if it sees a DB2 V1.x or DB2 V2.x instance, or it executes the binary db2ckmig_pe if it sees a DB2/PE instance.

db2ckmig will only check databases that are cataloged in an instance. If your database is not cataloged, db2ckmig will not perform verification tests on those databases.

If you uncatalog those databases that you do not wish to migrate to DB2 UDB V5, the only way to access them after migration is to catalog them in another DB2 V1 or V2 instance.

To verify that cataloged databases can be migrated, as the instance owner, use the following command:

```
DB2DIR/instance/db2ckmig <database -l<logfile> [<user/password>]
```

where:

- DB2DIR is where DB2 has been installed on your UNIX platform, as shown in this table:

PLATFORM	DB2DIR
IBM AIX	/usr/lpp/db2_05_00
HP-UX, Solaris	/opt/IBMdb2/V5.0

- <database> can be a database alias as listed in the database directory, or specify all cataloged databases with the -e flag.
- -l <logfile> is a mandatory flag and filename, to specify a file where db2ckmig will place all error output. This file is overwritten each time db2ckmig is invoked.
- <user/password> is optional, specified as -u <userid> -p <password>, and can be used to specify a different userid and password to access a database.

Note:

- The database migration verification tool, db2ckmig, when used with the *-e* flag, will attempt to verify all cataloged databases, whether local or remote. If you have a mixture of local and remote databases, with different username and password combinations, be aware that running db2ckmig with the -e flag may log numerous error messages when it attempts to verify the remote databases. If this is the case, you may choose to uncatalog these remote databases, run the db2ckmig program, and then recatalog these remote databases again so that during installation, the database directories are migrated. You may also choose to run db2ckmig on each database separately, in which case you should check the log file after each iteration.
- There is an undocumented flag, -h, which makes db2ckmig check all local cataloged databases. Remote cataloged database are ignored.
- In DB2 UDB V5, databases cannot be cataloged with a mix of authentication types: all databases use the authentication defined at the instance level. If db2ckmig detects mixed authentication types in a DB2 V1 instance, it will write the warning AUTHENTICATION MISMATCH to its logfile. db2imigr will still migrate the instance and will change the authentication to that of the DB2 UDB V5 instance, which by default is SERVER.

If there are any errors in the log file, refer to the following for suggested corrective actions.

- A database is in backup pending state:

  Perform a backup of the database.

- A database is in roll-forward pending state:

  Recover the database as required; perform or resume a roll-forward database

- Table space not in normal state:

  Recover the database and table spaces as required; perform or resume a roll-forward database.

- A database is in database transaction inconsistent state:

Restart the database to return it to a consistent state.

- The database contains database objects that have a schema name of SYSCAT, SYSSTAT, or SYSFUN.

  These schema names are reserved for the Version 5 database manager. To correct this error, do the following:

  1. Backup the database.
  2. Export the data from the database object (catalogs or tables).
  3. Drop the object.
  4. Recreate the object with the corrected schema name.
  5. Import / Load the data into the object.
  6. Run db2ckmig against the database again, ensuring that the database passes the db2ckmig check.
  7. Make a backup copy of the database.

- The database contains database objects that have a dependency on the SYSFUN.DIFFERENCE function. Possible violated database objects are: Constraint, Function, Trigger, View.

  The SYSFUN.DIFFERENCE function must be dropped and recreated during database migration. However, if there is a database object that is dependent on this function, migration will fail. To correct this error, do the following:

  – Constraint

    Issue the ALTER TABLE command to drop the constraint.

  – Function

    Issue the DROP FUNCTION command to drop the function dependent on SYSFUN.DIFFERENCE.

  – Trigger

    Issue the DROP TRIGGER command to drop the trigger.

  – View

    Issue the DROP VIEW command to drop the view.

  Note: Any package dependent on SYSFUN.DIFFERENCE will be marked inoperative after migration. Therefore, the db2ckmig program will not report any package that is dependent on the SYSFUN.DIFFERENCE function.

- The database contains user-defined distinct types that use the type name of DATALINK or REFERENCE.

  The data type names, DATALINK and REFERENCE, are reserved for the Version 5 database manager. To correct this error, you must rename these objects as follows:

  1. Back up the database.
  2. Export the data from any tables that are dependent on the data types.
  3. Drop the tables dependent on the data types, and then drop the data types. These drops may drop other objects such as views, indexes, triggers, or functions.
  4. Create data types with different type names and recreate the tables using the new data type names. Recreate any dropped views, indexes, triggers, or functions.
  5. Import/load the data into the tables.
  6. Run db2ckmig against the database again, ensuring that you make a backup copy of the database.

6.2.2 DB2 Servers: Migrating Instances

This section describes the actual migration of your instances. By this stage, you should have installed DB2 UDB V5 and created your Database Administration Server instance if required.

When you run db2imigr to migrate an instance, it will do the following:

- Ensure that the migration of the DB2 version currently being used by the instance is supported.
- Check the authentication types.
- Verify that the fenced user ID information is correct.
- Take a backup of the database manager configuration file, database, node and/or dcs directories, and other useful instance information including the $INSTHOME/sqllib directory.
- Create a new DB2 UDB V5 instance and migrate all files and directories for that instance.
- Copy UDFs, install a userexit and update the DB2 Profiles Registry.

Note:

- You can only migrate the instance as a root user and not the instance user. db2imigr calls db2ckmig with the -h flag prompting it to only check that your local databases are ready for migration. If you have cataloged remote databases they will not be checked. The migration of the instance will carry over the database, node and DCS directory entries of any remote databases.
- If there is not enough disk space in the instance home directory, the instance migration will fail. The only way of recovering from this failure is to drop and recreate the instance at DB2 V1 or DB2 V2, and then restore from a backup copy of your database. Recataloging the database back into the instance will not work as the database directories will also have been partially updated.

6.2.2.1 Create a User ID for Fenced UDFs

A user ID for fenced UDFs and fenced Stored Procedures is mandatory when migrating to DB2 UDB V5. This user is known as the fenced user ID. It is recommended for security reasons that you do not use the instance owner as the fenced user ID. If this user ID is set to the instance owner, then it is very easy to create an application that becomes the instance owner and can perform actions as if it was the DBADM. However if you do not want to create a fenced user, then you can specify the instance username as the fenced user.

On AIX, you should use SMIT or the AIX commands mkgroup and mkuser to create this user ID. For example:

```
mkgroup db2fadm1
mkuser pgrp=db2fadm1 db2fenc1
```

6.2.2.2 Use db2imigr to Migrate an Instance

To migrate an instance, log in as root and run the following command:

```
DB2DIR/instance/db2imigr -u <fenced userid> <instance>
```

where:

- DB2DIR is where DB2 has been installed on your UNIX platform, as shown in this table:

PLATFORM	DB2DIR
IBM AIX	/usr/lpp/db2_05_00
HP-UX, Solaris	/opt/IBMdb2/V5.0

- <fenced userid> is the name of the user ID to be used for fenced UDFs and Stored Procedures.
- <instance> is the name of the instance to be migrated.

When the instance is migrated successfully, the following message will be displayed:

```
# /usr/lpp/db2_05_00/instance/db2imigr -u db2fenc1 inst1
DBI1070I Program db2imigr completed successfully.
```

If the instance migration is unsuccessful, you may see this message:

```
DBI1124E Instance inst1 cannot be migrated.

Cause:  An attempt was made to migrate an instance. This
instance cannot be migrated because:

o   The instance is still active.

o   Migration of this instance is not supported

o   This version of the "db2imigr" command cannot be used
    to migrate this instance.

Action:  Ensure that instance is ready for migration and
you are using the correct version of the "db2imigr"
command. For more information on instance migration, please
refer to the book "Quick Beginnings for UNIX Environments".

DBI1079I Output is saved in the log file /tmp/db2imigr.log.14118.

Cause:  All processed and failed operations have been saved
into this log file.

Action:  Do not modify this file in any way. This file is
for IBM Technical Support reference.
```

If you see this message, you should:

- Check that the instance is not active
- Check that no processes owned by the instance owner are running
- Check that no DB2 processes are running (including db2licd)
- Check with db2set -l to make sure that the instance does not already exist in DB2 UDB V5. If it does, then remove connections to all databases in the

instance, stop all instance processes and then use the following command to remove it, where inst1 is the name of the instance:

```
DB2DIR/instance/db2idrop inst1
```

6.2.2.3 Verify that the Instance was successfully migrated

To list the instances that exist in DB2 UDB V5, run:

```
db2set -l
```

Another way to verify that the instance is now at Version 5 is to check the links in the instance user's sqllib directory. These should now point to DB2 UDB V5 (db2_05_00) directories. If /home/inst1 is the instance home directory, run:

```
ls -al /home/inst1/sqllib
```

The output should be similar to this:

```
lrwxrwxrwx 1 root     db2adm  Aug 12 doc -> /usr/lpp/db2_05_00/doc
drwxrwsr-t 3 db2inst2 db2adm  Aug 12 function
lrwxrwxrwx 1 root     db2adm  Aug 12 include -> /usr/lpp/db2_05_00/include
lrwxrwxrwx 1 root     db2adm  Aug 12 java -> /usr/lpp/db2_05_00/java
lrwxrwxrwx 1 root     db2adm  Aug 12 lib -> /usr/lpp/db2_05_00/lib
drwxrwsr-t 2 db2inst2 db2adm  Aug 12 log
lrwxrwxrwx 1 root     db2adm  Aug 12 map -> /usr/lpp/db2_05_00/map
lrwxrwxrwx 1 root     db2adm  Aug 12 misc -> /usr/lpp/db2_05_00/misc
lrwxrwxrwx 1 root     db2adm  Aug 12 msg -> /usr/lpp/db2_05_00/msg
lrwxrwxrwx 1 root     db2adm  Aug 12 odbclib -> /usr/lpp/db2_05_00/odbclib
lrwxrwxrwx 1 root     db2adm  Aug 12 samples -> /usr/lpp/db2_05_00/samples
```

Note: If you migrate an instance that has a remote DB2 V1 database cataloged, you will get the following error message when trying to connect to that database. Connection from a DB2 UDB V5 instance to a DB2 V1 database is not supported (with the exception of DB2 Parallel Edition V1.2).

```
[jc900c]inst1(v5)$ db2 connect to bsample
SQL5048N  The release level of the database client is not supported by the
release level of the database server.
```

Connecting to a remote DB2 V2 or V5 database should result in this message being displayed:

```
[jc900c]inst1(v5)$ db2 connect to bsample user inst2 using inst2

   Database Connection Information

 Database product       = DB2/6000 5.0.0
 SQL authorization ID   = INST2
 Local database alias   = BSAMPLE
```

This completes instance migration and we are now ready to migrate databases.

6.2.3 DB2 Servers: Migrating Databases

After the instance has been migrated successfully, with no errors in the migration log files, you need to individually migrate the databases within that instance. If you try to connect to your database before running the db2 migrate command, you will see the following error displayed:

```
SQL5035N  The database requires migration to the current release.
SQLSTATE=55001
```

6.2.3.1 Use db2 migrate to migrate your databases

To migrate your database, log on as the instance owner, then migrate each database with the following command replacing sample with your database name.

```
db2 migrate database sample
```

A successful database migration will result in the following message being displayed:

```
[inst1]$ db2 migrate database sample
DB20000I  The MIGRATE DATABASE command completed successfully.
```

To verify that the database was successfully migrated, try connecting to the database:

```
db2 connect to sample
```

A successful connection will result in the following message being displayed:

```
[jc9003c]inst1(v5)$ db2 connect to sample

   Database Connection Information

 Database product       = DB2/6000 5.0.0
 SQL authorization ID   = INST1
 Local database alias   = SAMPLE
```

6.2.3.2 Restoring the Database from a Backup

Another method to migrate a database is to restore from a database backup made using DB2 V1 or DB2 V2. Be aware that rolling forward of down-level logs is not supported.

6.3 Migrating DB2 V1/V2 Clients to DB2 UDB V5

This section discusses how to migrate a DB2 V1 or DB2 V2 Client to Universal Database Version 5. A DB2 client is a DB2 instance that does not have any local databases, only cataloged remote databases. If your instance has a DB2 local database, then it is not only a client, but also a server, and you should refer to 6.2, “Migrating DB2 V1/V2 Servers to DB2 UDB V5” on page 266.

In the UNIX environment, migration of DB2 Client systems comprises of only one distinct phase; *Instance Migration*. When you migrate a Client, you are migrating

only the instance; hence you do not need to migrate any databases. This section takes you through the Client migration process on the UNIX platform.

6.3.1, “DB2 Clients: Pre-Migration” prepares your instance for migration, takes you through installing DB2 UDB V5, and checks your databases in the instance with the utility db2ckmig. It also looks at some problems that you may come across whilst checking your instance, and how to resolve these.

6.3.2, “DB2 Clients: Migrating Instances” on page 283 takes you through migrating your instance environment to UDB V5. It looks at some problems that you may come across during migration, how to overcome these, and how to check that the instance migration has worked.

Note: DB2 Clients on UNIX platforms are migrated differently to DB2 Clients on Intel platforms. On OS/2 and Windows NT DB2 Clients, the migration of database, node and DCS directories is done during the installation of DB2 UDB V5.

6.3.1 DB2 Clients: Pre-Migration

To prepare your instance for migration, you need to make sure that all applications are disconnected from the instance, and the database server is not running.

1. Make copies of:
   - Database Manager Configuration
   - Database, node and DCS directories

   Use the following commands where inst1 is the name of the instance:

   ```
   db2 get database manager configuration > dbmcfg.inst1
   db2 list database directory > dbdir.inst1
   db2 list node directory > nodedir.inst1
   db2 list dcs directory > dcsdir.inst1
   ```

2. Ensure all applications disconnect from all databases.

   First, make sure all database transactions have been completed. You can obtain a list of all applications using the databases owned by the instance using the following command:

   ```
   db2 list applications
   ```

   You can then force termination of all applications using the following command:

   ```
   db2 force application all
   ```

   Get a list of applications again to ensure that all applications have been terminated, using the following command:

   ```
   db2 list applications
   ```

   Stop all database server processes owned by the DB2 instance using the following command:

   ```
   db2stop
   ```

   Note: You can also try `db2stop -kill`.

   Stop all command line processor sessions, any user invoked DB2 sessions using the following command:

   ```
   db2 terminate
   ```

Unload shared libraries with the following command:

```
slibclean
```

Check the state of IPC (Inter-Process Communications) resources owned by DB2 instances:

```
ipcs
```

Remove any IPC resources owned by DB2 instances. For example:

```
ipcrm -m MESSAGEID -q SHAREDMEMID -s SEMAPHOREID
```

where:

MESSAGEID is the ID of a message queue.
SHAREMEMID is the ID of a shared memory block.
SEMAPHOREID is the ID of a semaphore.

6.3.1.1 Install DB2 UDB V5 Client Application Enabler

In this section we will install DB2 UDB V5, which contains the db2imigr utility that will be used later in the migration process. Unlike the Intel platforms, on UNIX you can have multiple versions of DB2 installed; the version you are migrating from, and the version you are migrating to.

To perform the installation you should use one of these two methods:

1. Use the db2setup tool provided in DB2 UDB V5 to install the DB2 Client Application Enabler. For more details on the db2setup tool, refer to 6.2.1.2, “Installing DB2 UDB V5” on page 269.
2. Use the standard installation tool (SMIT on AIX) to install the DB2V5-Client bundle.

You are now in a position to move onto the next section on migrating instances.

6.3.2 DB2 Clients: Migrating Instances

This section describes the actual migration of your instances. By this stage, you should have installed DB2 UDB V5 Client Application Enabler.

When you run db2imigr to migrate an instance, it will do the following:

- Ensure that the migration of the DB2 version currently being used by the instance is supported.
- Check the authentication types.
- Verify that the fenced user ID information is correct.
- Take a backup of the database manager configuration file, database, node and/or dcs directories, and other useful instance information including the $INSTHOME/sqllib directory.
- Create a new DB2 UDB V5 instance and migrate all files and directories for that instance.
- Copy UDFs, install a userexit and update the DB2 Profiles Registry.

Note:

- You can only migrate the instance as a root user and not the instance user. The migration of the instance will carry over the database, node and DCS directory entries of any remote databases.

- If there is not enough disk space in the instance home directory, the instance migration will fail. The only way of recovering from this failure is to drop and recreate the instance at DB2 V1 or DB2 V2.

6.3.2.1 Create a User ID for Fenced UDFs

A user ID for fenced UDFs and fenced Stored Procedures is mandatory when migrating to DB2 UDB V5. This user is known as the fenced user ID. It is recommended for security reasons that you do not use the instance owner as the fenced user ID. If this user ID is set to the instance owner, then it is very easy to create an application that becomes the instance owner and can perform actions as if it was the DBADM. However if you do not want to create a fenced user, then you can specify the instance username as the fenced user.

On AIX, you should use SMIT or the AIX commands `mkgroup` and `mkuser` to create this user ID. For example:

```
mkgroup db2fadm1
mkuser pgrp=db2fadm1 db2fenc1
```

6.3.2.2 Use db2imigr to Migrate an Instance

To migrate an instance, log in as *root* and run the following command:

```
DB2DIR/instance/db2imigr -u <fenced userid> <instance>
```

where:

- DB2DIR is where DB2 has been installed on your UNIX platform, as shown in this table:

PLATFORM	DB2DIR
IBM AIX	/usr/lpp/db2_05_00
HP-UX, Solaris	/opt/IBMdb2/V5.0

- `<fenced userid>` is the name of the userid to be used for fenced UDFs and Stored Procedures.
- `<instance>` is the name of the instance to be migrated.

When the instance is migrated successfully, the following message will be displayed:

```
# /usr/lpp/db2_05_00/instance/db2imigr -u db2fenc1 inst1
DBI1070I Program db2imigr completed successfully.
```

If the instance migration is unsuccessful, you may see this message:

```
DBI1124E Instance inst1 cannot be migrated.

Cause:  An attempt was made to migrate an instance. This
instance cannot be migrated because:

o   The instance is still active.

o   Migration of this instance is not supported

o   This version of the "db2imigr" command cannot be used
    to migrate this instance.

Action:  Ensure that instance is ready for migration and
you are using the correct version of the "db2imigr"
command. For more information on instance migration, please
refer to the book "Quick Beginnings for UNIX Environments".

DBI1079I Output is saved in the log file /tmp/db2imigr.log.14118.

Cause:  All processed and failed operations have been saved
into this log file.

Action:  Do not modify this file in any way. This file is
for IBM Technical Support reference.
```

If you see this message, you should:

- Check that the instance is not active.
- Check that no processes owned by the instance owner are running.
- Check that no DB2 processes are running.
- Check with db2set -l to make sure that the instance does not already exist in DB2 UDB V5. If it does, then remove connections to all databases in the instance, stop all instance processes and then use the following command to remove it, where inst1 is the name of the instance:

  ```
  DB2DIR/instance/db2idrop inst1
  ```

6.3.2.3 Verify that the Instance was successfully migrated

To list the instances that exist in DB2 UDB V5, run:

```
db2set -l
```

Another way to verify that the instance is now at Version 5 is to check the links in the instance user's sqllib directory. These should now point to DB2 UDB V5 (db2_05_00) directories. If /home/inst1 is the instance home directory, run:

```
ls -al /home/inst1/sqllib
```

The output should be similar to this:

```
lrwxrwxrwx 1 root     db2adm  Aug 12 doc -> /usr/lpp/db2_05_00/doc
drwxrwsr-t 3 db2inst2 db2adm  Aug 12 function
lrwxrwxrwx 1 root     db2adm  Aug 12 include -> /usr/lpp/db2_05_00/include
lrwxrwxrwx 1 root     db2adm  Aug 12 java -> /usr/lpp/db2_05_00/java
lrwxrwxrwx 1 root     db2adm  Aug 12 lib -> /usr/lpp/db2_05_00/lib
drwxrwsr-t 2 db2inst2 db2adm  Aug 12 log
lrwxrwxrwx 1 root     db2adm  Aug 12 map -> /usr/lpp/db2_05_00/map
lrwxrwxrwx 1 root     db2adm  Aug 12 misc -> /usr/lpp/db2_05_00/misc
lrwxrwxrwx 1 root     db2adm  Aug 12 msg -> /usr/lpp/db2_05_00/msg
lrwxrwxrwx 1 root     db2adm  Aug 12 odbclib -> /usr/lpp/db2_05_00/odbclib
lrwxrwxrwx 1 root     db2adm  Aug 12 samples -> /usr/lpp/db2_05_00/samples
```

Note: If you migrate an instance that has a remote DB2 V1 database cataloged, you will get the following error message when trying to connect to that database. Connection from a DB2 UDB V5 instance to a DB2 V1 database is not supported (with the exception of DB2 Parallel Edition V1.2).

```
[jc900c]inst1(v5)$ db2 connect to bsample
SQL5048N  The release level of the database client is not supported by the
release level of the database server.
```

Connecting to a remote DB2 V2 or V5 database should result in this message being displayed:

```
[jc900c]inst1(v5)$ db2 connect to bsample user inst2 using inst2

   Database Connection Information

 Database product       = DB2/6000 5.0.0
 SQL authorization ID   = INST2
 Local database alias   = BSAMPLE
```

6.4 Migrating a Single Partition Database to a V5 Multi-Partition Database

This section describes how to migrate your DB2 V1, or DB2 V2 single partition database to a multi-partitioned DB2 UDB V5 database. We will be covering two methods of achieving this migration:

- Export the data from your database, split the data into flat files, then use these flat data files to reload the data into each database partition on the participating nodes.
- Redistribute an existing database across several nodes.

We will be discussing Export and Reload and Redistribution in this chapter. The method you decide to use will depend upon the complexity of your database environment. Before your single partitioned database can be migrated over to a parallel environment, set up your environment as described in the Pre-Migration section below.

6.4.1, “Single to Multi-Partition: Pre-Migration” on page 287 discusses the scenario where database migration from single partition to a multi-partitioned database is possible. It compares the methods to achieve this.

6.4.2, “Using Export and Reload” on page 288 discusses how your data is relocated when moving from a single partition to a multi-partitioned database using the *Export and Reload* method, before taking you through an example migration.

6.4.3, “Using Redistribute” on page 294 discusses how your data is relocated when moving from a single partition to a multi-partitioned database using the *Redistribute* method, before taking you through an example migration.

6.4.1 Single to Multi-Partition: Pre-Migration

Before you migrate your single partition database to a multi-partition database, you should install DB2 UDB V5 on each of your participating nodes. Migrating your database from a DB2 V1 or DB2 V2 single partition database directly to a multi-partitioned DB2 UDB V5 database is not supported. The following steps take you through some pre-migration requirements.

- Make sure you have taken a recent backup of your database before continuing any further.
- We recommend that you install DB2 UDB V5 EEE on all participating nodes on your SP2 system (or network of RS/6000s).To install the product, refer to 6.2.1.2, “Installing DB2 UDB V5” on page 269.
- If you choose to use the Redistribute method, then you must migrate your single partition database to DB2 UDB V5 before continuing further. To migrate your database to DB2 UDB V5, follow the steps in 6.2, “Migrating DB2 V1/V2 Servers to DB2 UDB V5” on page 266.

6.4.1.1 Using Export and Reload versus Redistribute

Using Export and Reload:

- No logging is performed during the load stage. This method is usually used when creating a new multi-partitioned database environment, and loading the partitioned data onto each node. The administrator can use either Load or Import to load the data into the new database environment. We recommend the Load utility for large volumes of data, as it is typically much faster.
- Although there is more administrator interaction preparing the database for migration, during the actual migration there is less downtime compared to the Redistribute method.
- Taking this method in it’s entirety, (exporting, splitting and ftp-ing the data files) it is more time consuming than Redistribute.
- It is easier to recover from failure than Redistribute, as you have your flat data files to reload from.

Using Redistribute:

- Logging is performed during the `db2 redistribute` command; if a failure occurs, the database can be rolled back. If the table is large, the rollback can take a long time.
- Two partition maps exist - one of your previous nodegroup environment, and one that includes your new nodes.
- This method requires lots of log space. If a failure occurs during redistribution, and if you choose to rollback, then redistribution uses the old partition map and the logfiles to roll the database back to its original state.

You can, however, increase your logspace and choose to continue your redistribution.

- Can in certain cases be faster than using Export and Reload, but can be time consuming if you need to rollback to the original state.

6.4.2 Using Export and Reload

This method describes the steps required to migrate a DB2 V1 or DB2 V2 single partition database system to a DB2 UDB V5 multi-partitioned database system, using Export and Reload. This method uses the db2split program to partition the data across nodes. It can be used in two ways.

- You can run db2split using the default partitioning map to divide the data into separate files, one for each node. In this situation, an input partitioning map is not required.
- You can run db2split in analysis mode to generate your own customized partitioning map then run db2split in partition mode using the generated partition map to customize the distribution of data across nodes.

For our example we will run db2split using the default partitioning map.

6.4.2.1 The Splitter Utility, db2split

The db2split program uses the Partitioning Key for a table to determine (by a hashing algorithm) the node where the data will be loaded. The Partitioning Key is made up of one or more columns in the table. The output of db2split is typically one file per node; each file contains the rows to be loaded at that node. To process data with db2split the data must be a sequential file. You can run db2split on MVS, VM, or AIX. You can choose to run db2split where the source data resides or where you have DB2 UDB V5 installed.

The db2split utility requires the data file, a configuration file and, optionally, an input partitioning map. The configuration contains information such as the name of the input file, the position and length of the partitioning key, and the number of nodes. To see a sample of a db2split.cfg, refer to the file DB2DIR/samples/splitter/db2split.cfg, or refer to Appendix A, “db2split.cfg” on page 353.

Note: db2split splits the data at table level. You must follow the db2split procedure for each table that you wish to move to a multi-partitioned database.

How db2split partitions data: The *db2split* program takes a partitioning key value from each line in the input data file and then uses a hashing algorithm to generate a vector into the partitioning map. This vector is a value between 0 and 4095. The partitioning map determines which node the row will be loaded at. For example, in Figure 175 on page 289, we see that a input line with the value 38 in the partitioning key is hashed to the vector 140. The node value for vector 140 in the partitioning map is 0, so this row will be loaded at node 0. Any other row with 38 as the partitioning key value will similarly be loaded at node 0. By contrast, a row with the value 45 in the partitioning key is hashed to the vector 3050. The node value for vector 3050 in the partitioning map is 1, so this row will be loaded at node 1.

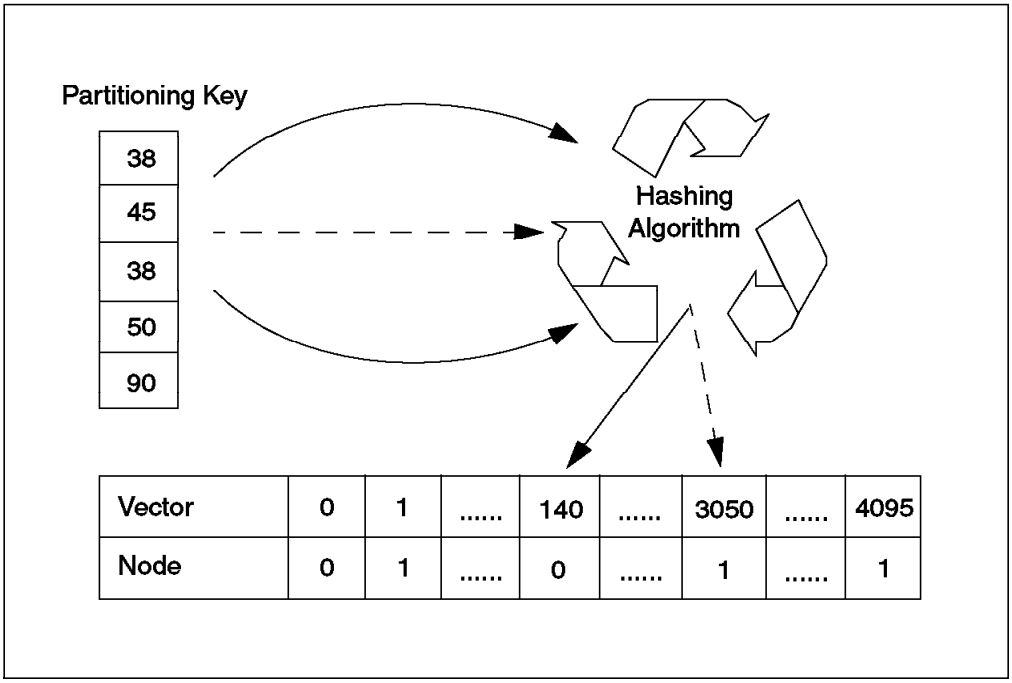


Figure 175. How db2split Partitions Data

Partitioning Maps: By default, partitioning maps use a round-robin system of node numbers. For example, for a 2 partition system, we would see the default partitioning map defined as follows:

Vector	0	1	2	3	4	5	6	7	8	....	4095
Node	0	1	0	1	0	1	0	1	0	....	1

For a 3 partition system, we would see a default partitioning map as follows:

Vector	0	1	2	3	4	5	6	7	8	....	4095
Node	0	1	2	0	1	2	0	1	2	....	2

6.4.2.2 Export and Reload Example

The following example details the steps to move the data in a single partition database to a multi-partition database using export and reload. First, we will go through the steps necessary to configure a 2-node DB2 UDB V5 EEE environment, then the steps to move data from a single to multi-partition database.

1. **Configure DB2 UDB V5 EEE**

   The installation of DB2 UDB V5 EEE must be completed successfully on all of the participating nodes before starting these steps. For more details on installation, see 6.2.1.2, “Installing DB2 UDB V5” on page 269

- Create a user ID for your Instance.

Create the instance owner user ID and also a fenced user ID to run fenced UDFs. Set passwords for these users, and then distribute these users across all nodes participating in the EEE environment. For example, in a 2-node system:

```
mkgroup udb ; mkuser pgrp=udb inst1
mkgroup db2fadm1; mkuser pgrp=db2fadm1 db2fenc1
passwd inst1 ; passwd db2fenc1
dsh -w node0,node1 /var/sysman/supper update
```

In this example, these commands should be issued from the SP Control Workstation since we are using File Collections to manage users and groups.

- Create a shared instance home directory

  Choose which node to hold your local home directory. This directory must be shared across your nodes within your EEE environment. Add this directory to the /etc/exports file, and nfs-mount this directory from all the other nodes. For example:

  On node 0: `smit mknfsexp`

  On node 1: `mount node0:/home/inst1 /home/inst1`

- Create an instance

  Create an instance using the db2icrt utility. For example:

  On Node 0: `DB2DIR/instance/db2icrt -u db2fenc1 inst1`

  Add a line as below in the instance owner's .profile file to source the db2profile file.

  ```
  . $INSTHOME/sqllib/db2profile
  ```

- Update db2nodes.cfg

  Update db2nodes.cfg (in $INSTHOME/sqllib) to include the list of nodes in the EEE environment. For example, for our two node parallel environment, db2nodes.cfg contains:

  ```
  0 node0 0
  1 node1 0
  ```

- Update /etc/services

  On all the nodes in the EEE environment, add entries in the /etc/services file to used by the instance. For example:

  ```
  DB2_inst1     50000/tcp
  DB2_inst1_END 50001/tcp
  ```

- Update remote hosts file

  Remote execution permission must be granted for the instance owner across all the nodes in the EEE environment. For example, add the following 2 lines to the .rhosts file in the instance owner's home directory:

  ```
  +node0
  +node1
  ```

  To test that this has worked, form node0, try:

  ```
  rsh node1 ls
  ```

  This should give a directory listing without prompting for a password.

- Start DB2

  Start the database manager by running:

```
db2start
```

If this is successful, you will see messages from each node indicating that DB2 processes have been started on that node. If you see any error messages, you must resolve the related problems before proceeding any further.

- Create a database

  For the multi-partitioned database, you must create a directory on each node that is not shared. This directory must have the same name across the nodes and should be a separate filesystem. For example, as root:

  On Node 0,1: `smitty jfs to create /database; mount /database`

  On Node 0,1: `chown inst1.udb /database/inst1`

  As the instance owner:

  On Node 0: `db2 "create database TPCD on /database"`

- Create nodegroups and table spaces

  Create nodegroups and table spaces as required. If you do not specify a nodegroup when you create a table space, DB2 will use IBMDEFAULTGROUP which includes all the nodes in your EEE environment. If you do not specify a table space when you create a table, DB2 will place the table in the last table space created by the user running the command. If the user has not created any table spaces, the table will be created in the USERSPACE1 table space in the IBMDEFAULTGROUP nodegroup.

  For example, to create a nodegroup called Mynodegrp:

  ```
  db2 "create nodegroup Mynodegrp on nodes (0,1)"
  ```

  To create a table space called Mytablesp in Mynodegrp using a raw container of 4 MB:

  ```
  db2 "create tablespace Mytablesp in nodegroup Mynodegrp managed by
  database using (device '/dev/rlv05' 1000)"
  ```

  Note: In DB2/PE V1, a table was directly associated with a nodegroup. In DB2 UDB V5, a table is in a table space within a node group. When a user issues a CREATE TABLE statement, the name following the IN keyword is a table space name, not a nodegroup name.

- Create tables

  The partitioning key for each table is specified in the CREATE TABLE command. This partitioning key must match the partitioning key used during the db2split processing of the flat files to be loaded into the table. If you do not specify a partitioning key, DB2 will use the first column by default (as long as it is not a long data type).

  For example, to create a table called lineitem in the table space Mytables*p*:

  ```
  create table tpcd.lineitem        ( l_orderkey   integer not null,
    l_partkey       integer not null, l_suppkey    integer not null,
    l_linenumber    integer not null, l_quantity     float not null,
    l_extendedprice   float not null, l_discount     float not null,
    l_tax             float not null, l_returnflag char(1) not null,
    l_linestatus    char(1) not null, l_shipdate      date not null,
    l_commitdate      date not null, l_receiptdate   date not null,
    l_shipinstruct char(25) not null, l_shipmode  char(10) not null,
    l_comment   varchar(44) not null)
  in Mytablesp;
  ```

The column l_orderkey will be used as the partitioning key.

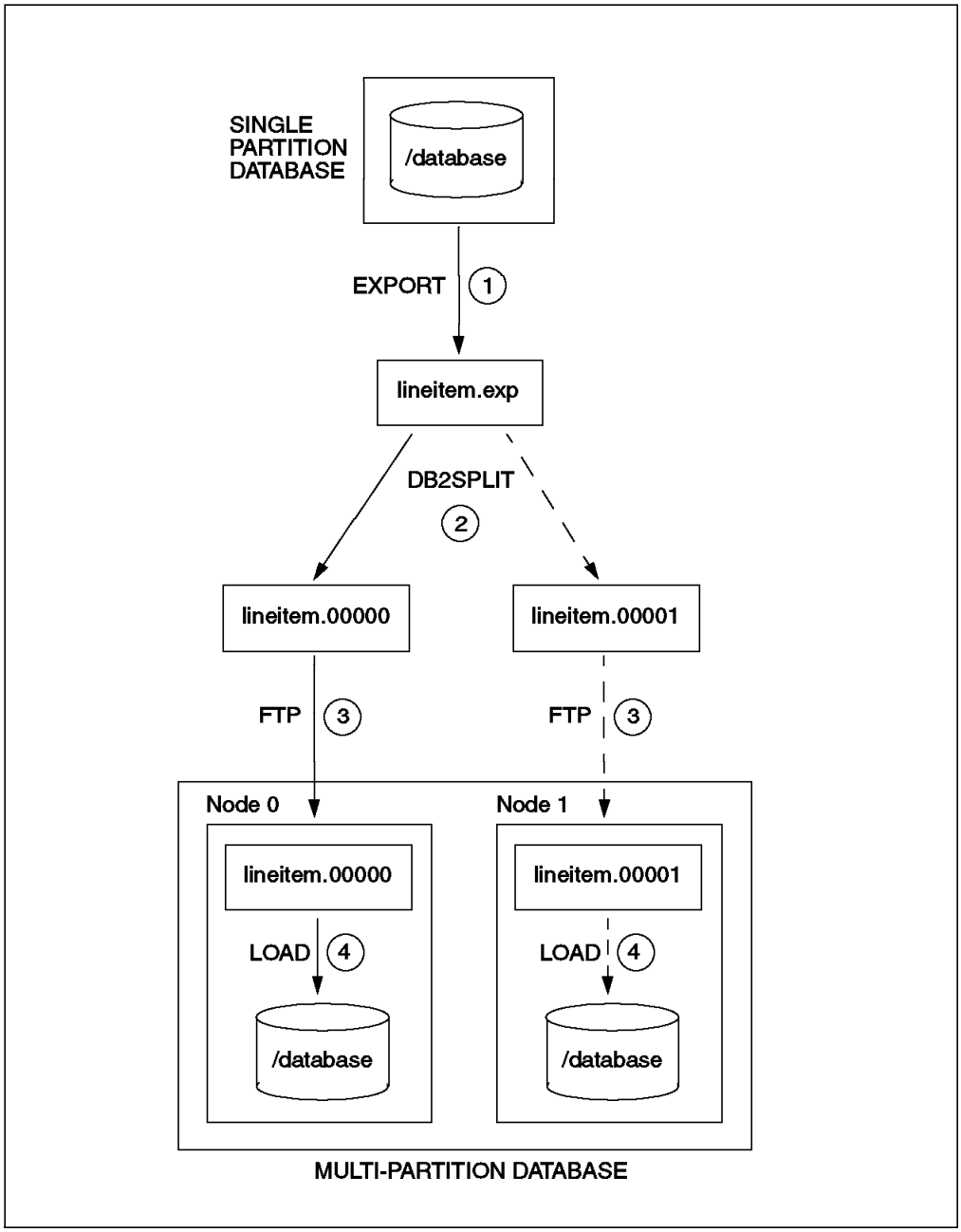


Figure 176. Moving Data to a Multi-Partition Database

1. Move the data from a single to a multi-partition database

- Export your data from your tables

  From your single partition database, run *export* against your tables. For example:

  ```
  db2 "export to /data/lineitem.exp of del modified by coldel | select *
  from lineitem"
  ```

  This is step 1 in Figure 176.

- Run db2split to partition the data

Before running db2split you should edit the db2split configuration file. In our example we have chosen l_orderkey as the partitioning key for the lineitem table. Make sure you have sufficient space in the target location for the output files. The parameters we used for our 2 node environment are as follows:

Customized Parameters	Values
InFile	/data/lineitem.exp
FileType	DEL
Nodes	(0,1)
LogFile	/home/inst1/hashing.log
Partition	L_ORDERKEY,1,,,NN,INTEGER
header	yes
CDelimiter	|
RecLen	3200
RunType	PARTITION

Now you are ready to run db2split with the customized configuration file. This command will generate an output file for each node as defined in the OUTFILE parameter. In our example the output files are called lineitem.00000 and lineitem.00001. Run the following command, where db2split.cfg is the configuration file:

```
db2split -c db2split.cfg
```

This is step 2 in Figure 176 on page 292.

For more details about these parameters, refer to the db2split.cfg file in DB2DIR/samples/splitter listed in Appendix A, “db2split.cfg” on page 353.

- Send the partitioned output files to the appropriate node

  Use a program such as the file transfer program (FTP) to send each data file to the appropriate node.

  This is step 3 in Figure 176 on page 292

- Load your data simultaneously across the nodes

  Use the Load utility to load the data from the output files generated by db2split as follows:

  On node 0:

  ```
  db2 "load from /split/lineitem.00000 of del modified by coldel | insert
  into lineitem"
  ```

  On node 1:

  ```
  db2 "load from /split/lineitem.00001 of del modified by coldel | insert
  into lineitem"
  ```

  This is step 4 in Figure 176 on page 292

- Test a select against the partitioned table

  To ensure that all rows were successfully loaded, you should run:

  ```
  db2 "select count(*) from tpcd.lineitem"
  ```

  The number of rows displayed should equal the number of rows in the single partition version of the table.

1. Backup the database.

Backup the database, as the changes made by the LOAD command are not logged.

6.4.3 Using Redistribute

This method describes the steps required to migrate either a DB2 V1 or DB2 V2 single partition database system to a DB2 UDB V5 multi-partition database system, using Redistribute.This method redistributes data across the nodes in a multi-node system. This section gives a working example of how to achieve this. It assumes that your database is already at DB2 UDB V5 on one of your nodes. If this is not the case, follow the steps in 6.2, “Migrating DB2 V1/V2 Servers to DB2 UDB V5” on page 266. Migrating from a DB2 V1/V2 single partition database directly to a multi-partition DB2 UDB V5 database is not supported. You must first migrate to a single partition DB2 UDB V5 database. Once this is complete, you can follow the steps below.

6.4.3.1 How redistribute works

This method uses the db2 redistribute command to rebalance data across a multi-node system after the instance has been extended to include new nodes. The db2 redistribute command requires a nodegroup name and a list of new nodenames as input.

Unlike the db2split utility, the redistribute command uses two partitioning maps; the original partitioning map plus the new partitioning map which includes the new nodes. If the redistribute process fails, then the original partitioning map is used as reference if you choose to rollback to the original state.

Note: Redistribute allows you to choose to rollback to the original state if it fails. The rows to be moved between nodes are first inserted into the new node then deleted from the old node. The units of work can be large if the table is large and the command will fail if it does not have a sufficient amount of log space.

The following is an example of using Redistribute to migrate data from a single partition database to a multi-partition database.

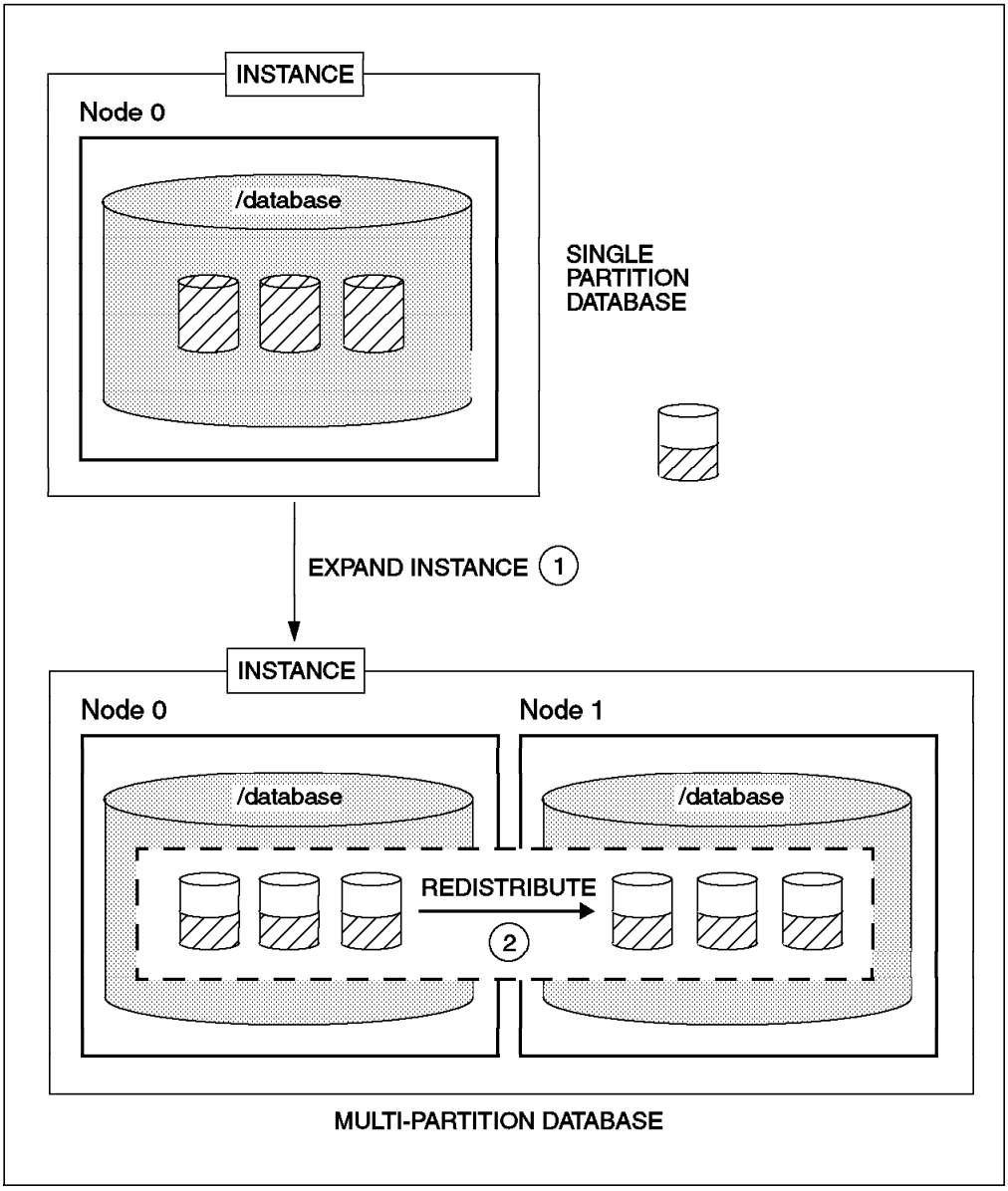


Figure 177. The Redistribute Method

6.4.3.2 Redistribute Example

The following example details the steps to move the data in a single partition database to a multi-partition database using Redistribute. First, we will go through the steps necessary to convert a DB2 UDB V5 single partition database (for example, Enterprise Edition or EE) to a 2-node DB2 UDB V5 EEE environment, then the steps to move data from a single to multi-partition nodegroup within the database.

1. Convert DB2 UDB V5 Environment from single to multi-partition

   Before starting these steps, make sure that the migration of the DB2 V1/V2 single partition database system has been completed successfully. For more details on this migration, see 6.2, “Migrating DB2 V1/V2 Servers to DB2 UDB V5” on page 266.

   This is step 1 in Figure 177.

- Install DB2 UDB V5 EEE

  On the node that has been migrated to DB2 UDB V5, use SMIT to install the DB2 UDB V5 EEE package, db2_05_00.pext. On the node which is being added to the environment, if DB2 is not installed, follow the procedure for installing DB2 UDB V5 and make sure that the EEE package is included in the install. For more details on installation, see 6.2.1.2, “Installing DB2 UDB V5” on page 269.

- Check that the instance owner exists on the new node

  If the instance owner userid and fenced userid do not exist on the new node, then distribute these users across all nodes participating in the EEE environment. For example:

  ```
  dsh -w node0,node1 /var/sysman/supper update
  ```

- Share the instance home directory

  The instance home directory must be shared across your nodes within your EEE environment. Add this directory to the /etc/exports file, and nfs-mount this directory from all the other nodes. For example:

  On node 0: `smit mknfsexp`

  On node 1: `mount node0:/home/inst1 /home/inst1`

- Create a filesystem for the database files

  For the database, you must create a directory on each node which is not shared. This directory must have the same name across the nodes and should be a separate filesystem. For example, as root:

  On Node 1: `smitty jfs to create /database; mount /database`

  On Node 1: `chown inst1.udb /database/inst1`

- Update /etc/services

  On all the nodes in the EEE environment, add entries in the /etc/services file used by the instance. For example:

  ```
  DB2_inst1     50000/tcp
  DB2_inst1_END 50001/tcp
  ```

- Update remote hosts file

  Remote execution permission must be granted for the instance owner across all the nodes in the EEE environment. For example, add the following 2 lines to the .rhosts file in the instance owner’s home directory:

  ```
  +node0
  +node1
  ```

  To test that this has worked, form node0, try:

  ```
  rsh node1 ls
  ```

  This should give a directory listing without prompting for a password.

- Update your instance to a EEE instance

  You should use the db2iupdt command to update your instance to a EEE instance. To check your instance type, run this command:

  ```
  db2 get dbm cfg | grep Node
  ```

  For a DB2 UDB V5 single partition system, you will see:

  ```
  Database server with local and remote clients
  ```

To update you instance, run the following command from node 0:

```
/usr/lpp/db2_05_00/instance/db2iupdt -u db2fenc1 inst1
```

To check that this has worked, rerun:

```
db2 get dbm cfg | grep Node
```

Now you should see:

```
Partitioned Database Server with local and remote clients
```

Note: Running db2iupdt will update the specified instance by performing the following changes:

- Add or replace the files in the $INSTHOME/sqllib directory, where $INSTHOME is the home directory of your instance.
- Create a new database manager configuration by merging relevant values from the existing database manager configuration with the default database manager configuration for the new node type. If a new database manager configuration file is created the old file is backed up to $INSTHOME/sqllib/backup/db2systm.old

- Use db2start addnode to add the new node.

  Use the db2start command with the addnode clause to add the new node to the instance. This command will add the new node to the db2nodes.cfg file. It will also create the necessary directory structure in the filesystem which stores the database files on the new node (/database in our example). Run:

  On node 0: db2start nodenum 1 addnode hostname node1 port 0

  If this command is successful you will see a message similar to the following:

```
f01n03:/home/inst1 > db2start nodenum 1 addnode hostname node1 port 0

09-08-1997 15:10:03     1   0   SQL6075W  The Start Database Manager
operation successfully added the node. The node is not active until all
nodes are stopped and started again.

SQL6033W  Stop command processing was attempted on "1" node(s).  "0" node(s)
were successfully stopped.  "0" node(s) were already stopped.  "0" node(s)
could not be stopped.
```

  To activate the new node, you should stop and then restart DB2. If your nodes do not restart successfully the first time, you may have to wait awhile, stop DB2 and start it again.

```
db2stop
db2start
```

1. Redistribute data with the db2 redistribute command

   You have now extended your existing database environment to include your the new node. You are ready to redistribute the data across all the nodes in each nodegroup.

- Activate archival logging and increase log space

  Before you run the redistribute command you should set logretain on to activate archival logging. You may also need to increase your log space since during the execution of the redistribute, when a row is moved from one node to another, a delete and insert are performed. This activity is logged. This typically requires a considerable amount of log space. For example, to set logretain on, run:

On all nodes: db2 update db cfg for TPCD using logretain yes

- Redistribute the data

  Connect to your database. You are now ready to complete your migration by redistributing your data across all the nodes in the EEE environment. The redistribute command automatically expands all the tables in a specified nodegroup across all participating nodes. For example, on node 0:

  ```
  db2 connect to TPCD
  db2 "redistribute nodegroup IBMDEFAULTGROUP uniform add node (1)"
  ```

  This is step 2 in Figure 177 on page 295.

6.5 Post-Migration

There are several optional activities you may wish to undertake following database migration:

- Unique index conversion to DB2 Universal Database Version 5 semantics

  Version 5 of DB2 supports deferred unique constraint checking until end of statement. This can result in correct processing of multiple row updates, that in previous releases of DB2, returned an error because the updates temporarily created duplicate values in the transient state. Deferred unique constraint checking will guarantee that updates, that result in a table with only unique keys (for example, key = key + 1) will succeed regardless of the order of the data.

  Note: This change only applies for unique indexes that are created in DB2 UDB V5.

  All unique indexes, in a migrated database, do not automatically migrate to Version 5 semantics during database migration because of the following reasons:

  – Converting unique indexes is a very time-consuming operation. You may have applications that depend on the previous version's unique index semantics. You may wish to manage the staged conversion of unique indexes on your own schedule, when needed.

  – All existing applications will continue to work even if the unique indexes are not converted to Version 5 semantics. You have to convert unique indexes to Version 5 semantics only if support for deferred uniqueness checking is required.

  To convert unique indexes, you need to perform the following steps:

  1. Log on with a user ID that has SYSADM authority.
  2. Issue the db2start command.
  3. Run the db2uiddl command against your migrated database. The syntax of this command is as follows:

     ```
     db2uiddl -d <database-name> [<-u table-schema>] [<-t table-name>]
     [<-f filename>] [<-h help>]
     ```

     where:

  – <database-name> is the name of the database to be queried.

  – <-u table-schema> is optional, and can be used to specify the schema (creator user ID) of the tables that are to be processed. The default action is to process tables created by all user IDs.

- <-t table-name> is optional, and can be used to specify the name of a table that is to be processed. The default action is to process all tables.
- <-f filename> is optional, and can be used to specify the name of a file to which output is to be written. The default action is to write output to standard output.
- <-h help> can be used to display help information. When this option is specified, all other options are ignored, and only the help information is displayed.

  Note: This tool was not designed to handle certain types of names. If a specific table name or table schema is a delimited identifier containing lower case characters, special characters, or blanks, it is preferable to request processing of all tables or schemas. The resulting output can be edited.

  The db2uiddl command searches the database catalog tables and generates all the CREATE UNIQUE INDEX statements for user tables in an output file.

1. Review the output generated from the db2uiddl command, and make changes, if needed. Comments in the output will flag any situations that require your attention.
2. Execute the file as a DB2 Command Line Processor command file, using a command similar to the following:

   ```
   db2 -tvf filename
   ```

   where filename is output file from *db2uiddl*.

DB2 interprets the recreation of an already-existing unique index to signal that the index is ready to be converted to Version 5 semantics.

- Update Statistics

  When database migration is completed, the old statistics, used to optimize query performance, are retained in the catalogs. However, Version 5 of DB2 has statistics that are modified or do not exist in the previous version. To take advantage of these, you may wish to issue the runstats command on tables, particularly those tables that are critical to the performance of your SQL queries.

  Refer to the *DB2 UDB V5 Command Reference* for the syntax of the runstats command. For details on the statistics, refer to the *DB2 UDB V5 Administration Guide*.

- Rebind Packages

  During database migration, all existing packages are invalidated during catalog migration. After the migration, each package is rebuilt when it is used for the first time by the Version 5 database manager.

  However, for better performance, we recommend that you run the db2rbind command to rebuild all packages stored in the database, after database migration is complete. The following gives an example using *sample* as our database:

  ```
  db2rbind sample -l /tmp/db2rbind.log
  ```

  Refer to the *DB2 UDB V5 Command Reference* for more information about this command.

- Update database and database manager configuration

Some of the database configuration parameters are changed to Version 5 defaults or to other values during database migration. The same is true for database manager configuration parameters that may have changed, during instance migration, to Version 5 defaults or to other values. For better performance, you may wish to tune your database and database manager configuration parameters to take advantage of Version 5 enhancements. Refer to the *DB2 UDB V5 Command Reference* for the syntax of updating database and database manager configuration.

The following database manager configuration parameters are changed to Version 5 defaults:

- Database configuration release level

  This is changed to 0x0800.

- Sort Heap threshold (SHEAPTHRES)

  If the migrating database configuration file has this parameter at a value which is less than the Version 5 Default, the parameter is reset to its Version 5 default value of 10000 x 4 K pages.

- Backup buffer (BACKBUFSZ)

  If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to its Version 5 default value of 1024 x 4 K pages.

- Restore Buffer (RESTBUFSZ)

  If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to the default value of 1024 x 4 K pages.

- Number of concurrent databases running against the database manager (NUMDB)

  If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to its Version 5 default value, as follows:

Type of Database	Number
Database server with local and remote clients	8
Database server with local clients	3

- Transaction Manager database name (TM_DATABASE)

  If the migrating database configuration file has this parameter set to NULL, the parameter is reset to 1ST_CONNECT (1ST_CONN).

- Query heap size (QUERY_HEAP_SZ)

  If the migrating database configuration file has this parameter at a value less than the Application Support Layer heap size (ASLHEAPSZ) of the same file, the parameter is reset to ASLHEAPSZ + 1. The default value is 1000 x 4 K pages.

- Maximum number of idle agents (MAX_IDLEAGENTS)

If the migrating database configuration file has this parameter greater than the Maximum Simultaneous agents (MAXAGENTS) of the same file, the parameter is reset to MAXAGENTS -1. See the NUM_INITAGENTS and MAXAGENTS parameters for further information.

– TCP/IP Service Name (SVCENAME)

  If the service name in the TCP/IP services file is changed, the SVCENAME parameter is correspondingly changed. For example, from db2c to db2cDB2.

The following database configuration parameters are changed to Version 5 defaults:

– Database configuration release level

  This is changed to 0x0800.

– Database release level

  This is changed to 0x0800.

– Application control heap size (*app_ctl_heap_sz*)

  If the migrating database configuration file has this parameter at a value that is less than the Version 5 Default, the parameter is reset to its Version 5 default value, as follows:

Type of Database	Number (4 K pages)
Database server with local and remote clients	128
Database server with local clients	64
Partitioned Database server with local and remote clients	256

– Lock list (LOCKLIST)

  For Version 1 DB2 databases, the LOCKLIST parameter will be first adjusted to LOCKLIST * 32/ 25. If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to its Version 5 default value, as follows:

Type of Database	Number (4 K pages)
Database server with local and remote clients	50
Database server with local clients	25

– Database heap (DBHEAP)

  For Version 1 DB2 databases, the database heap size will first be adjusted to dbheap * 16. If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to its Version 5 default value, as follows:

Type of Database	Number (4 K pages)
Database server with local and remote clients	600
Database server with local clients	300

– Default log space (LOGFILSIZ)

  The database migration process will attempt to increase the LOGFILSIZ value, if the log file related parameters have a total logfilsiz that is less than the default logfilsiz value of 250 x 4 K pages.

– Application Heap size (APPLHEAPSZ)

  For Version 1 DB2 databases, application heap size will be first adjusted to APPLHEAPSZ * 16. If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to its Version 5 default value, as follows:

Type of Database	Number (4 K pages)
Database server with local and remote clients	128
Database server with local clients	64

– Sort Heap (SORTHEAP)

  For Version 1 DB2 databases, the value of the SORTHEAP parameter will be adjusted to SORTHEAP * 16.

– Statement Heap Size (STMTHEAP)

  If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to the default value of 2048 x 4 K pages.

– Recovery Range and Soft Checkpoint Interval (SOFTMAX)

  If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to the default value of 100.

– Package Cache Size (PCKCACHESZ)

  This is always reset to the Version 5 default, of -1, which means the value is 8 * maxappls, unless 8 * maxappls is less than 32, in which case the default of -1 will set PCKCACHESZ to 32.

• Migrate Explain Tables.

  If you are not using explain tables in DB2 Version 2, skip this task.

  Version 5 of DB2 has added several new columns to the explain tables. These columns provide for the capture of:

  – Data for new SQL features added in Version 5.

  – More detailed access plan information.

While the explain function in Version 5 will continue to work with explain tables created for Version 2, the new Version 5 data will not be captured in them.

For better performance of SQL statements, we recommend that the Version 2 explain tables be dropped and new explain tables be created; see the SQL Reference and the Administration Guide for details on creating new explain tables. If, however, there are Version 2 explain tables that you need for ongoing comparison, you can use the EXPLMIG.DLL script to migrate them.

To migrate the explain tables in a database that has been migrated to Version 5, connect to the database and run the following command from the sqllibmisc directory:

```
db2 -tf sqllibmiscEXPLMIG.DLL
```

The explain tables belonging to the user ID that is used to connect to the database will be migrated. To migrate explain tables belonging to another user, connect to the database with that user ID and run the command.

6.5.1 Other Changes Introduced in DB2 UDB V5

During a database migration to DB2 UDB V5, the database path structure is modified and DB2's use of environment variables is modified. The DB2 Profiles Registry is introduced. The system catalog tables and views are changed. This section explains some of the changes that have occurred as a result of migrating your database system to DB2 UDB V5.

- The DB2 Profiles Registry

  Registry values, environment variables and configuration parameters control your database environment. In DB2 UDB V5, almost all of the environment variables have been moved to the DB2 Profiles Registry. The DB2 Profiles Registry is particularly advantageous for OS/2 and Windows NT DB2 systems since every time a variable is changed, the system does not need to be rebooted for the parameters to take effect. The DB2 Profiles Registry will also support applications that have no environmental settings. It centralizes control of environment variables on a global machine-wide level as well as supporting multiple environment profiles (one per instance).

  The DB2 Profiles Registry is defined in three distinct levels: the system or node registry, the global registry and the instance registry.This table displays the location of these flat files on the UNIX platform:

PROFILE REGISTRY	AIX	HP, SUN
System/Node Registry	/var/db2/v5/profiles.reg	/var/opt/db2/v5/profiles.reg
Global Registry	/var/db2/v5/default.env	/var/opt/db2/v5/default.opt
Instance Registry	$HOME/sqllib/profile.env	$HOME/sqllib/profile.env

- Profiles Registry after Migration

  During the migration process, only the files that hold the system and global registries are created. The file that holds the instance registry is not created, but a DB2 administrator can create this file to override and

customize the environment for that instance. The values in the instance registry override the values in the global registry.

The list of variables that these files hold after a migration differs between a migration from DB2 V1 and DB2 V2:

System Registry	Global Registry
List of DB2 UDB V5 Instances	(Empty)

The registries after migrating from DB2 V1 to DB2 UDB V5 are as follows:

The registries after migrating from DB2 V2 to DB2 UDB V5 are as follows:

System Registry	Global Registry
List of DB2 UDB V5 Instances	DB2SYSTEM
	DB2ADMINSERVER

- Customizing the Profiles Registry

  The db2set command is used to update the profile registry. It displays, sets, or removes DB2 profile variables.

  To list all instance profiles at UDB V5, including the Database Administration Server, run the following command:

  ```
  db2set -l
  ```

  To list all the possible environment variables that can be configured and stored in the profiles registry at any of the profile registry levels (system, global or instance), run the following command:

  ```
  db2set -lr
  ```

  This produces the following list of variables:

Customizable Variables within the DB2 Profiles Registry		
DB2ACCOUNT	DB2LPORT	DB2SORCVBUF
DB2ADMINSERVER	DB2NBADAPTERS	DB2SORT
DB2ADMINSTART	DB2NBCHECKUPTIME	DB2SOSNDBUF
DB2BQTIME	DB2NBINTRLISTENS	DB2SYSTEM
DB2BQTRY	DB2NBRECVBUFFSIZE	DB2THREADIF
DB2CHKPTR	DB2NBRECVNCBS	DB2TIMEOUT
DB2CLIENTADPT	DB2NBRESOURCES	DB2UPMPR
DB2CLIENTCOMM	DB2NBSENDNCBS	DB2YIELD
DB2CLIINIPATH	DB2NBSESSIONS	DB2_AVOID_PREFETCH
DB2CODEPAGE	DB2NBXTRANCBS	DB2_FORCE_TRUNCATION
DB2COMM	DB2NODE	DB2_GRP_LOOKUP
DB2COUNTRY	DB2NTNOCACHE	DB2_INDEX_FREE
DB2DBDFT	DB2NTPRICLASS	DB2_RR_TO_RS
DB2DBMSADDR	DB2NTWORKSET	DB2_SHARE_MMU
DB2DEFPREP	DB2OPTIONS	DB2NPIPE
DB2DIRPATHNAME	DB2PATH	DB2_MMAP_READ
DB2DMNBCKCTLR	DB2PGCHK	DB2_MMAP_WRITE
DB2INCLUDE	DB2PRIORITIES	DB2AUTOSTART
DB2INSTDEF	DB2RAWLOGS	DB2DISCOVERYTIME
DB2INSTPROF	DB2REMOTEPREG	DB2NBDISCOVERRCVBUFS
DB2IQTIME	DB2ROUTE	DB2ENVLIST
DB2LOADREC	DB2RQTIME	DB2LIBPATH
DB2LOCK_TO_RB	DB2SERVICETPINSTANCE	DB2_LOADSORT_STACKSZ

Note:

- The ADSM parameters have been moved from the DB2 V2 database manager configuration to the DB2 UDB V5 database configuration. to DB2 V5 database level. These parameters include: ADSM_MGMTCLASS, ADSM_NODENAME, ADSM_OWNER and ADSM_PASSWORD.

- Database and Node Directories and the db2profile file

  Migration should successfully copy over the database and node directory entries for any remote databases. You should be able to connect to your remote databases without recataloging them. You may experience problems connecting to your database if you have not exported the environment variable DB2COMM to the communications protocol that you are using. During migration, a new DB2 UDB V5 instance is created, along with a new $HOME/sqllib/db2profile file. During the Instance migration, the DB2COMM environment variable is unset. If you migrated your instance from DB2 V1, you can find a copy of your previously customized db2profile file in your instance owner's $HOME/sqllib_v1 directory. If you migrated your instance from DB2 V2, you can find a copy of your previously customized db2profile file in your instance owner's $HOME/sqllib_v2 directory. If you had set any environment variables within the db2profile file (such as DB2COMM), then we recommend that you set these variables in the DB2 Profiles Registry.

- Environment Variables

  During migration from DB2 V1/V2 to DB2 UDB V5, some of the environment variables used in the previous version of DB2 are not set in DB2 UDB V5. The following lists the variables that are affected.

Environment variables no longer present	Environment variables set in DB2 UDB V5
DB2BQTIME, DB2BQTRY, DB2CHKPTR, DB2COMM, DB2DBDFT, DB2GROUPS, DB2IQTIME, DB2RQTIME,	DB2DIR=/usr/lpp/db2_05_00 INSTHOME=/home/inst1

Environment variables after migrating from DB2 V1 to DB2 UDB V5:

Environment variables no longer present	Environment variables set in DB2 UDB V5
DB2DBDFT, DB2COMM	(None)

Environment variables after migrating from DB2 V2 to DB2 UDB V5:

- Authentication

  All local databases now have the same authentication type as the instance where they reside. The authentication type in the database directory is ignored by DB2 UDB V5 servers. Authentication exists at the instance for DB2 V2 and DB2 UDB V5. For DB2 V1, it exists at the database level.

- User Defined Functions after migration

  All User Defined Functions (UDFs) created on a DB2 V2 system will remain unaffected, and should run as before in DB2 UDB V5.

- View Definitions

  If a view that was defined in DB2 V1 involves ″SELECT *″, the view may be unusable after migration. If the view is unusable, attempts to use it, directly or indirectly, will result in SQLCODE -158. The view must be dropped and recreated in order to avoid this error. If fewer than the current number of columns in the SELECT * table is desired, the recreated view must specify the needed columns.

Chapter 7. DB2 UDB V5 Migration for DB2/PE Systems

This chapter gives an in depth analysis of migrating DB2/PE systems to DB2 UDB V5 Enterprise-Extended Edition (EEE). It covers detailed examples of migration on the AIX V4 platform, and also covers some post migration tasks such as moving data from SMS to DMS table spaces.

There are many considerations to make when planning your migration. The first section discusses how the migration will affect your database, and the planning required before your migration. Read this section before continuing any further:

- 6.1, “Migration Planning and Considerations” on page 261.

A standard database migration will result in the creation of a DB2 UDB V5 EEE database that uses SMS table spaces. If you are considering the use of DMS table spaces, see:

- 7.3, “Moving Data from SMS to DMS Table Spaces” on page 334

There is also a section on Post-Migration that discusses how the database environment has changed after a migration, and how to take advantage of some of the new features of DB2 UDB V5:

- 7.4, “Post-Migration” on page 345

Note: Migration from an AIX V3 platform is not supported. To migrate your DB2 system on AIX V3, you must upgrade AIX to at least V4.1.4 first.

7.1 Migration Planning and Considerations

This section describes the implications of migrating your database to Version 5, and how it affects the database path, system catalog tables and the environment. It discusses some migration issues to consider, such as database authentication, security and storage requirements to ensure a successful migration. It also considers the impact of some incompatibilities between two versions of the product.

7.1.1 Migration Implications on Databases

There is some basic planning and preparation to perform when migrating your database to DB2 UDB V5. The directory where the physical database files (for example, SQL00001) are stored has changed location.

During the migration of a database, the following events occur:

1. The following entities are migrated:
   - Database configuration file
   - Database system catalog tables
   - Database directories
   - Database log file header
   - Database index files
   - Database data files
2. The database is relocated to a new database path:

For DB2 UDB V5 the database path is common across all platforms. The instance/nodename path convention used in DB2 Parallel Edition V1 is used. The nodename path is limited to 8 characters. All back-level cataloged databases are required to be relocated to the new path, including DB2/PE V1 databases. For example, assuming /database is our local database directory, and inst1 is our instance name, the database path changes as follows.

DB2 VERSION	DATABASE PATH
DB2/PE V1	/database/inst1/NODE00000/SQL00001/
DB2 UDB V5 EEE	/database/inst1/NODE0000/SQL00001

3. The database is migrated to SMS table space containers. For more details on migrating your database to DMS table space containers, see 7.3, “Moving Data from SMS to DMS Table Spaces” on page 334.
4. One table space is created for each nodegroup; the name of the table space is the same as the name of the nodegroup. Also, the default table spaces, SYSCATSPACE, TEMPSPACE1 and USERSPACE1 are created. For example, in addition to any user-defined nodegroups, migrated databases will have the following nodegroups and table spaces:
   - IBMCATGROUP nodegroup, defined on the catalog node, containing:
     - SYSCATSPACE table space - for the migrated system catalogs
   - IBMTEMPGROUP nodegroup, defined on all nodes, containing:
     - TEMPSPACE1 table space - for temporary objects
   - IBMDEFAULTGROUP nodegroup, defined on all nodes, containing:
     - USERSPACE1 table space - the default table space for user-defined tables
     - IBMDEFAULTGROUP table space - for any tables that were defined in this nodegroup in DB2/PE

   Note: It is possible to allocate a different table space for each table during the database migration. For details, see 7.1.2.6, “Table Spaces Created during Database Migration” on page 313.
5. For each segment directory that existed in the DB2/PE database an SMS directory container will be created. The naming convention is as follows:

DB2/PE V1 Segment Directory	V5 EEE SMS Directory Container
SQLS0000 SQLS0001 .... SQLS0016	SQLS0000/SQLT0000.0 SQLS0001/SQLT0000.1 ... SQLS0016/SQLT0000.16

That is, the path name of the container includes all of the path of the segment directory that existed in DB2/PE V1 plus an additional directory whose name is:

- SQLTxxxxx.y

where:

- xxxxx is the tablespace ID
- y is the DB2/PE segment ID

So if you had 16 segment directories in the DB2/PE database, after migration you will have 16 SMS directory containers per table space.

6. System Catalog Tables are changed as follows:
   - New columns are added.
   - New tables are created. These are the new tables introduced at DB2 UDB V5:

New System Catalog Tables	
SYSATTRIBUTES	SYSBUFFERPOOLNODES
SYSBUFFERPOOLS	SYSCOLPROPERTIES
SYSHIERARCHIES	SYSNODEGROUPDEF
SYSNODEGROUPS	SYSPARTITIONMAPS
SYSPROCEDURES	SYSPROCPARMS
SYSSCHEMAAUTH	SYSSCHEMATA
SYSTRIGDEP	SYSVIEWS

   - A set of catalog views is migrated, and a set of new catalog views is created, in the SYSCAT schema. A set of updateable catalog views is created in the SYSSTAT schema. If you are migrating from DB2 V1, you will notice SYSCAT and SYSSTAT views introduced as these schema do not exist at DB2 V1. These are the new SYSCAT views introduced at DB2 UDB V5:

New System Catalog Views	
BUFFERPOOLNODES	BUFFERPOOLS
COLAUTH	COLPROPERTIES
NODEGROUPDEF	NODEGROUPS
PARTITIONMAPS	PROCEDURES
PROCPARMS	SCHEMAAUTH
SCHEMATA	

   - A set of general purpose scalar functions is kept, and a set of new general purpose scalar functions is created, in the SYSFUN schema. Only the SYSFUN.DIFFERENCE scalar function is dropped and re-created during database migration.

7. A new directory called db2event is created in the database directory.

   This directory did not exist at DB2/PE V1. The new location at DB2 UDB V5 is /database/inst1/NODE0000/SQL00001/db2event where /database is the database path, and inst1 is the name of the instance.

8. Buffer pool files are created in the database path.

   The buffer pool files created in the database path are SQLBP.1 and SQLBP.2.

9. A database history file and shadow are created in the database path.

   This is file contains a summary of backup information that can be used if a database must be restored, and it is updated whenever a backup, restore, or table load operation is performed on the database. A summary of backup information is also kept for backup and restore operations on a table space. This file exists as /database/inst1/NODE0000/SQL00001/db2rhist.asc where /database is your the database path and inst1 is the name of the instance.

7.1.2 Migration Considerations

To successfully migrate a database created with a previous version of DB2, you must consider the following:

- Migration restrictions
- Security and authorization
- Storage requirements
- Release-to-release incompatibilities
- Segment directories as JFS filesystems
- Table spaces created during database migration

7.1.2.1 Migration Restrictions

There are certain pre-conditions or restrictions that you should be aware of before attempting to migrate your database to DB2 UDB V5.

- Migration on the AIX platform is only supported at AIX V4.1.4 and above. Earlier versions of AIX must be upgraded to AIX V4.1.4 before starting the migration of DB2.
- Issuing the migration command from a V5 client to migrate a database on a V5 Server is supported. However, issuing a migration command from earlier versions of DB2 clients to a V5 Server is not supported.
- User objects within your database cannot have V5 reserved schema names as object qualifiers. These reserved schema names include: SYSCAT, SYSSTAT, and SYSFUN.
- Your database cannot be in one of the following states:
  - Backup pending
  - Roll-forward pending
  - One or more table spaces not in a normal state
  - Transaction inconsistent
- Restore of down-level (DB2/PE V1) database backups to DB2 UDB V5 is supported but rolling forward of down-level logs is not supported.

7.1.2.2 Security and Authorization

You need SYSADM authority to migrate your database.

When migrating from DB2/PE V1, you should know that a database cannot be cataloged with a mix of authentication types. The authentication type of the instance in DB2 UDB V5 is defined in the database manager configuration file. If mixed types are detected during migration from DB2/PE V1, you can either stop the migration and change the directories or continue with the migration. If migration continues, all the authentication types are changed to blank, and the database uses the authentication type specified in the instance. The default authentication type created when creating an instance at DB2 UDB V5 is SERVER.

To use two databases with different authentication types, a new instance must be created for one of the databases. The database should be backed up and restored to a new database under the new instance. It can then be dropped under the old instance and migration can then be run.

In DB2 UDB V5, users and groups are differentiated in SQL statements and the system catalog. As a result, if a user and a group have the same name in the previous version, the authority and privileges granted to the group must be explicitly re-granted after migration.

During migration, the authorization catalog tables, SYSCAT.DBAUTH, SYSCAT.INDEXAUTH, SYSCAT.PLANAUTH, and SYSCAT.TABAUTH, are checked to determine if existing privileges are for users or groups, and the GRANTEETYPE is defined as follows:

- If the name in the GRANTEE column is a user or is not defined, the GRANTEETYPE is defined as U.
- If the name in the GRANTEE column is a group, the GRANTEETYPE is defined as G.
- If the name in the GRANTEE column is both a user and a group, the GRANTEETYPE is defined as U. Privileges must then be explicitly granted to the group.

7.1.2.3 Storage Requirements

Space is required for both the old and new catalogs during the migration, and the amount of disk space required will vary depending on the complexity of the database as well as the number and size of the database objects. These objects include all tables and views. You should make available at least two times the amount of disk space than the database catalog currently occupies. If there is not enough disk space in the local database directory, the instance migration will fail. The only way of recovering from this failure is to drop and recreate the instance at DB2/PE V1, and then restore from a backup copy of your database. Recataloging the database back into the instance will not work since the database directories will also have been partially updated. The only solution is to restore from a backup.

You should also consider increasing the database configuration parameters associated with the log files. You should increase logfilsiz, logprimary, and logsecond to prevent the space for these files from running out. If log space is completely used, you will receive a SQLCODE of SQL1704N with a reason code of 3. If this happens, increase the log space parameters and re-issue the database migration command.

7.1.2.4 Release-to-Release Incompatibilities

To successfully migrate a database, you should consider the impact of the incompatibilities between the two versions of the product. The following incompatibilities deserve special attention before you begin your migration:

- **View Definitions**

  If a view defined in DB2/PE V1 involves ″SELECT *″, the view may be unusable after migration. If the view is unusable, attempts to use it, directly or indirectly, will result in SQLCODE -158. The view must be dropped and recreated in order to avoid this error. If fewer than the current number of columns in the SELECT * table is desired, the recreated view must specify the needed columns.

- **Configuration Parameters**

  – Configuration parameter values are preserved during the migration of the database, with the exception of the following parameters:

  – Application Heap Size (APPLHEAPSZ)

  – Package Cache Size (PCKCACHESZ)

  – Maximum Storage for Lock List (LOCKLIST)

  – Recovery Range and Soft Checkpoint Interval (SOFTMAX)

For these parameters, the use of the associated heap has changed significantly in DB2 UDB V5.

- APPLHEAPSZ is reset to the DB2 UDB V5 default value if the current value is less than the Version 5 default value.
- PCKCACHESZ is always reset to the DB2 UDB V5 default value.

For LOCKLIST, the DB2/PE V1 value is multiplied by a factor of 32/25. This computed value will be used as the DB2 UDB V5 parameter value, if this value is greater than the Version 5 default. Otherwise, the Version 5 default will be used.

7.1.2.5 Segment Directories as JFS filesystems

An important consideration during the migration of databases is the segment directory filesystems. Most DB2/PE users are using large databases and have a separate JFS filesystem for each segment directory in order to support more than 2 GB of data per node. When migrating a DB2/PE database to DB2 UDB V5 EEE, users can continue using the same allocation of disk used by DB2/PE filesystems. Note that the filesystems themselves will be renamed. For instance, if /MIG is the database path and inst1 the instance owner the filesystem:

```
/MIG/inst1/NODE00000/SQL00001/SQLS0000
```

will be renamed to:

```
/MIG/inst1/NODE0000/SQL00001/SQLS0000
```

Note: The change comes in the directory thatl identifies the node (partition). It changes from NODE00000 (5 zeroes) to NODE0000 (4 zeroes).

This is an overview of the process:

1. All segment directories are first checked to determine whether any of these directories are mounted JFS filesystems.

2. If any of the segment directories are mounted filesystems then DB2 creates a list of original mount points (in the original database path) and new mount points (in the new database path) pairs and writes them out to a file. This file is created in the $INSTHOME/sqllib/ctrl directory. The first line in the file stores the original database path and the new database path. DB2 writes out the original and new mount point pairs starting from the second line of the file. The file name convention is DBNAME.N.#.rmt where DBNAME is the database name, N is the node number and # is the file extension. The file extension is used in case there are multiple databases using the same database name but under a different alias name.
3. Once the list of original and new mount point pairs is complete, db2vmgr (the DB2 volume manager) is invoked to unmount all the original mount points. Then DB2 relocates the database and invoke db2vmgr to remount the original (now unmounted) directories to their new mount points based on this file.

7.1.2.6 Table Spaces Created during Database Migration

When you migrate DB2 Parallel Edition Version 1.x to DB2 UDB Version 5, the migration utility will automatically create a default set of SMS table spaces for you and migrate your tables to one of these table spaces. This default behavior will work well for many of your tables. You may, however, want to migrate some of your tables to specific table spaces that you identify ahead of time. For example, one very important consideration is that with UDB, when you load a table, its table space is placed in a quiesce state for the duration of the load. This means that other tables in the table space are not accessible during the load. (This is not the case with DB2 Parallel Edition, since it does not support table spaces). So, if you regularly perform large loads and depend on other tables being available during the loads, you should identify the target tables of these loads, and, using the information described below, migrate each to its own dedicated table space.

Other reasons you may want to customize table spaces at migration time include:

- UDB′s point-in-time table space roll-forward recovery can be simplified if related tables are in the same table space.
- To take advantage of UDB′s ability to restore a specified subset of table spaces from any backup.

By default, when you migrate from DB2 Parallel Edition to DB2 Universal Database Enterprise - Extended Edition for AIX, the following will occur:

- An SMS table space will be created for every nodegroup (the table space name will be the same as the nodegroup′s name).
- Every table will be migrated to the table space created for the nodegroup.

For example, if you have:

- Nodegroup NG1 consisting of nodes 1,2 and 3
- Nodegroup NG2 consisting of nodes 3 and 4
- Table TA in NG1
- Table TB in NG2
- Table TC in NG1

- Table TD in NG1

This configuration is shown in Figure 178.

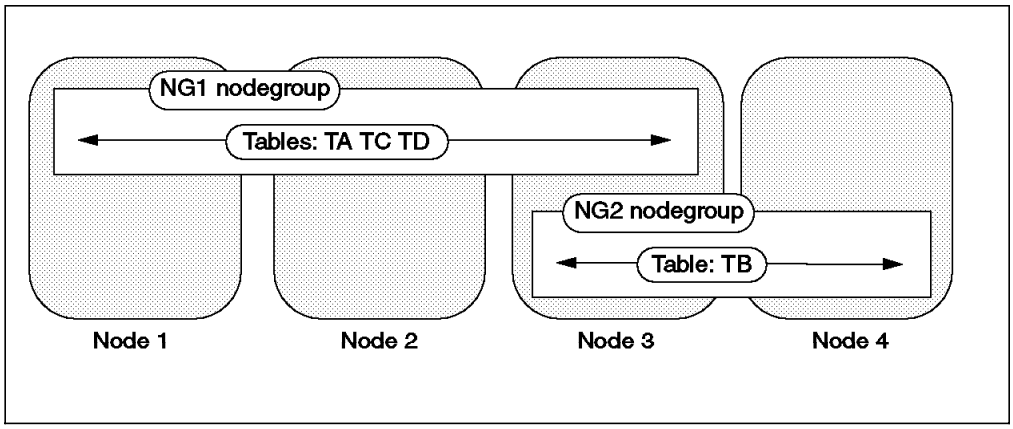


Figure 178. Example Scenario before Database Migration

Migration will create two SMS table spaces named NG1 and NG2 and migrate TA,TC and TD to NG1 and TB to NG2. Table space NG1 will exist in nodegroup NG1 (nodes 1, 2, and 3) and table space NG2
will exist in nodegroup NG2 (nodes 3 and 4). This is the default behavior, as shown in Figure 179.

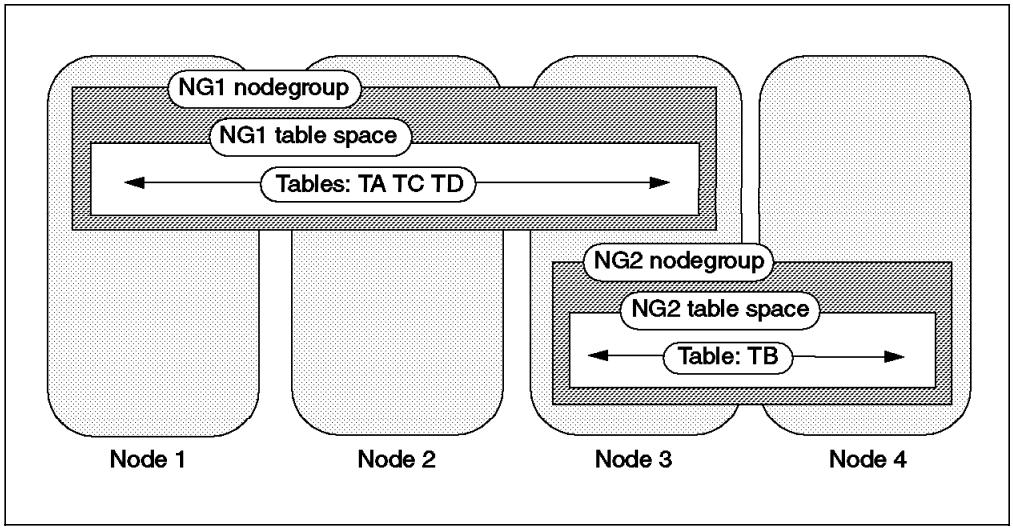


Figure 179. Default Table Spaces Created during Database Migration

You can override the default behavior as follows:

1. Create a file that lists specific tables and which table space each table should be migrated to. The format of the file is:

```
line 1: <d>
line 2: <d><schema><d><table><d><tablespace><d>
line 3: <d><schema><d><table><d><tablespace><d>
.
.
.
line n: <d><schema><d><table><d><tablespace><d>
```

   Where <d> is a single character delimiter. An example is as follows:

```
″
″S1″TC″TS1″
″S1″TD″TS2″
```

This file indicates to create SMS table spaces TS1 and TS2, and to migrate table TC (under schema S1) to TS1 and TD (under schema S1) to TS2. The delimiter used in this file is the ″ character.

Note: If you use a delimiter of ′ ′ (a blank), ensure that the first line actually contains a blank in the first column (a newline character in the first column is not treated as a blank).

2. Ensure that the DB2 instance owner has permission to read the file.
3. Issue the following command:

   ```
   db2set DB2_MIGRATE_TS_INFO=/a/b/c
   ```

   Where /a/b/c is the fully qualified path to the above file. (Alternatively, you can specify DB2_MIGRATE_TS_INFO in your db2profile file. Using db2set, however, is the recommended method.)
4. Issue the db2stop command.
5. Issue the db2start command.
6. Perform the migration.

This results in two additional SMS table spaces being created during migration: TS1 and TS2. Table TC will be migrated to TS1 and table TD will be migrated to TS2. Any table not listed in the file will be migrated according to the default behavior. So, continuing with the previous example, table TA will be migrated to table space NG1 and TB will be migrated to table space NG2. This is shown in Figure 180:

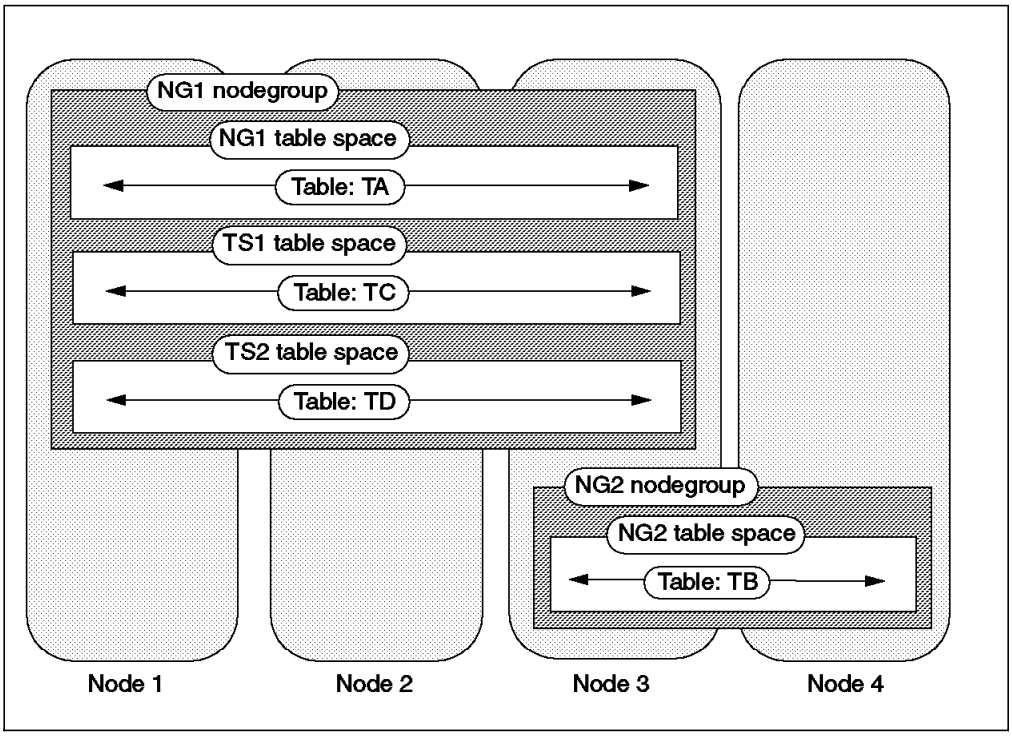


Figure 180. Customized Table Spaces Created during Database Migration

You should be aware of the following:

- All identifiers in the file are case sensitive.
- You can specify the same table space more than once in the file. If you do, ensure that all tables to be migrated to the table space exist in the same nodegroup.
- You cannot specify the same table more than once.
- Any errors that occur when the file is processed are reported in the db2diag.log file. An SQLCODE of -901 will be reported and the migration will be rolled back. If this occurs, correct the problems reported in the db2diag.log file and try the migration again.
- The schema for all table spaces created during migration is V5R1MIGR.
- This procedure has been tested to create up to 100 additional table spaces. You should be able to create more, but this has not been tested. (Note that the maximum number of table spaces per database is 4096).

7.2 Migrating DB2/PE Systems to DB2 UDB V5 EEE

Instances were introduced in DB2 for AIX V1 to facilitate separate and unique database manager environments. This allowed you to have several database manager environments, with alternative configuration parameters or authentication, all on the same processing platform. When considering migration, you may want to retain these separate instances under DB2 UDB V5. So, in the UNIX environment, the migrations comprised of two distinct, but interrelated, phases; Instance Migration and Database Migration.

7.2.1, “DB2/PE Systems: Pre-Migration” on page 317 prepares your instance for migration, takes you through installing DB2 UDB V5 EEE, and checks your databases within an instance with the utility *db2ckmig*. It also looks at some problems that you may come across while checking your instance, and how to resolve these.

7.2.2, “DB2/PE Systems: Migrating Instances” on page 326 takes you through the initial phase of migration, which is migrating your *instance* environment to DB2 UDB V5. It looks at some problems that you may come across during migration, how to overcome these, and how to check that your instance migration has worked.

6.2.3, “DB2 Servers: Migrating Databases” on page 281 takes you through the next phase of migration, which is migrating your databases within your instance environment to DB2 UDB V5. It looks at some problems that you may experience, how to overcome them, and then how to check that the database migration has worked.

Here is a summary of the steps that you need to follow to ensure a smooth migration of your DB2/PE system:

- Prepare your instance for migration.
- Install DB2 UDB V5 EEE.
- Create a Database Administration Server instance (optional).
- Create a userid for fenced User Defined Functions (optional).
- Run `db2ckmig` to check the databases that you wish to migrate.

- Run db2imigr to migrate an instance.
- Run the db2 migrate database command to migrate the databases within an instance.

7.2.1 DB2/PE Systems: Pre-Migration

DB2 UDB V5 provides the utility db2ckmig, to allow you verify whether cataloged databases within an instance can be migrated. When migrating an instance (using the db2imigr utility), the db2ckmig utility will be executed.

In this section we will:

- Prepare your instance for migration.
- Install DB2 UDB V5 EEE.
- Create the Database Administration Server instance user ID and a fenced user ID.
- Detail the steps required to verify and correct the database state.

It is advisable that you perform these pre-migration steps before migrating to DB2 UDB V5. Migration is not a recoverable process. If you do not have a backup of your databases before you attempt the migration, and the migration fails, you will have no way of restoring your database using either DB2 UDB V5 or a previous version of DB2.

Note:

- All databases that you wish to migrate to DB2 UDB V5 must be cataloged, otherwise the migration of those databases will not occur.
- In order for a migration to be successful, all applications and end users must be disconnected from the database being migrated.

7.2.1.1 Preparing the Instance for Migration

Use the following checklist to ensure your instance is ready for migration:

1. Ensure all local databases that you want to migrate are cataloged.

   All databases that are to be migrated, must be cataloged. In a DB2 Command Window, check the database directory with the following command:

   ```
   db2 list database directory
   ```

   If you have any local databases that you do not want to migrate to DB2 UDB V5, then you should uncatalog them before migration.

2. Make a backup copy of all databases.

   Make a backup copy of the database before you start the next procedure since migration is not a recoverable process. If you do not have a backup of your databases from before you attempt to migrate them and the migration fails, you will have no way of restoring your database using DB2 UDB V5 or a previous version of DB2. For additional information on the backup command, refer to the *DB2/PE Administration Guide*.

   ```
   db2_all "db2 backup database sample to dir/dev"
   ```

3. Make copies of:

   - Database Manager Configuration

- Database Configuration for each database

Use the following commands where sample is the name of the database and inst1 is the name of the instance:

```
db2 get database manager configuration > dbmcfg.inst1
```

On each node:

```
db2 get database configuration for sample > dbcfg.inst1.sample
```

4. Ensure all applications disconnect from all databases.

   First, make sure all database transactions have been completed. You can obtain a list of all applications using the databases owned by the instance using the following command:

   ```
   db2_all "db2 list applications"
   ```

   You can then force termination of all applications using the following command:

   ```
   db2_all "db2 force application all"
   ```

   Get a list of applications again to ensure that all applications have been terminated, using the following command:

   ```
   db2_all "db2 list applications"
   ```

   Stop all database server processes owned by the DB2 instance using the following command:

   ```
   db2stop
   ```

   Note: You can also try db2stop -kill

   Stop all command line processor sessions, any user invoked DB2 sessions using the following command:

   ```
   db2 terminate
   ```

   Unload shared libraries with the following command:

   ```
   db2_all slibclean
   ```

   Check the state of IPC (Inter-Process Communications) resources owned by DB2 instances:

   ```
   db2_all ipcs
   ```

   Remove any IPC resources owned by DB2 instances. For example, on each node:

   ```
   ipcrm -m MESSAGEID -q SHAREDMEMID -s SEMAPHOREID
   ```

   where:

   MESSAGEID is the ID of a message queue
   SHAREMEMID is the ID of a shared memory block
   SEMAPHOREID is the ID of a semaphore

Now you are ready to install DB2 UDB V5 EEE.

7.2.1.2 Installing DB2 UDB V5 EEE

In this section we will install DB2 UDB V5 EEE, which contains the db2ckmig and db2imigr utilities that will be used later in the migration process. On AIX, you can have multiple versions of DB2 installed; the version your are migrating from, and the version your are migrating to.

There are several ways to install DB2 UDB V5 EEE:

- Using the db2setup tool provided with V5

  This method is practical for systems with small numbers of nodes.

- Using SMIT

  Again, this method is practical for systems with small numbers of nodes

- Using dsh to run installp across all nodes

  This method should be used for larger systems and allows for the installation to be automated and run simultaneously across all the nodes.

Installation using db2setup: If you choose to use the db2setup tool provided in DB2 UDB V5 to install the DB2 products (via installp), you should be aware that this tool also allows you to:

1. Create a new instance, and for that instance:
   - Create a new user ID and group.
   - Create a fenced user ID and group for fenced UDFs.
   - Setup communications.
   - Configure environment variables and DB2 Profile Registry entries.

   Since we will be migrating from an instance that exists in DB2/PE V1, the step to create an instance should only be used for new instances that did not exist in the previous version of DB2.
2. Create a Database Administration Server instance, and for that instance:
   - Create a new user ID and group.
   - Setup communications.
   - Configure environment variables and DB2 Profile Registry entries.

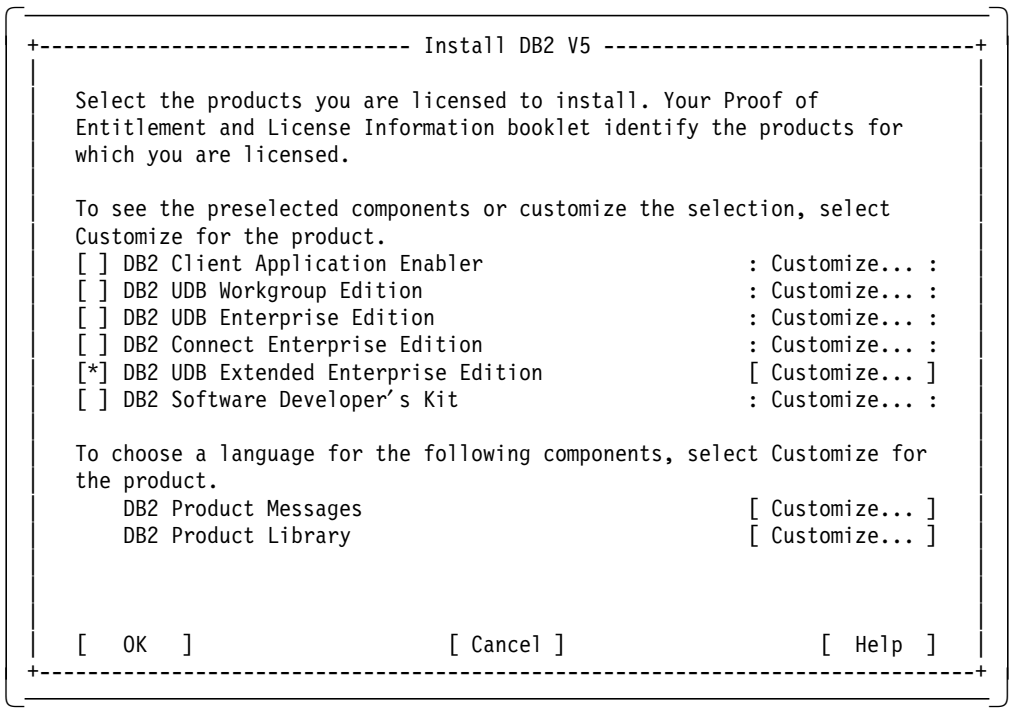
   The step to create a Database Administration Server (DAS) instance is new to DB2 UDB V5 and it is recommended that you complete this step. The DAS instance is used to provide services to support client tools that automate the configuration of connections to DB2 databases, and to enable administration of the DB2 Server system from remote clients.

To start the installation, log in as root, mount the CD-ROM, and run the db2setup tool that is provided with DB2 UDB V5 from your mount directory:

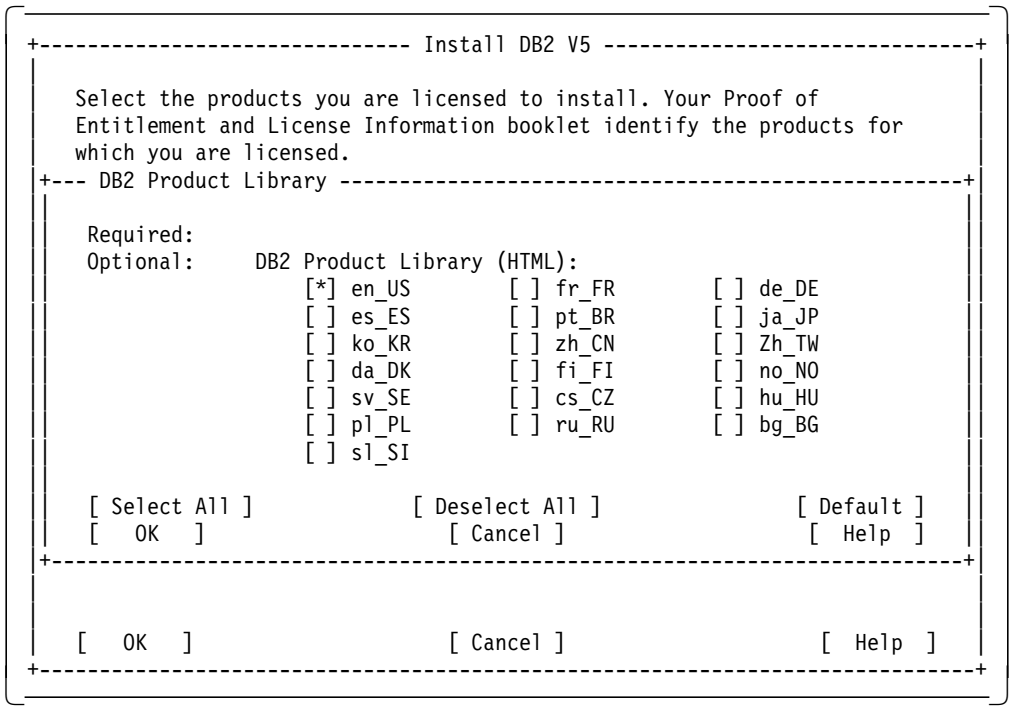
```
./db2setup
```

The following screens take you through an example of an installation using db2setup. The screen shown below prompts you to select the product that you wish to install. You should select DB2 UDB Extended Enterprise Edition.

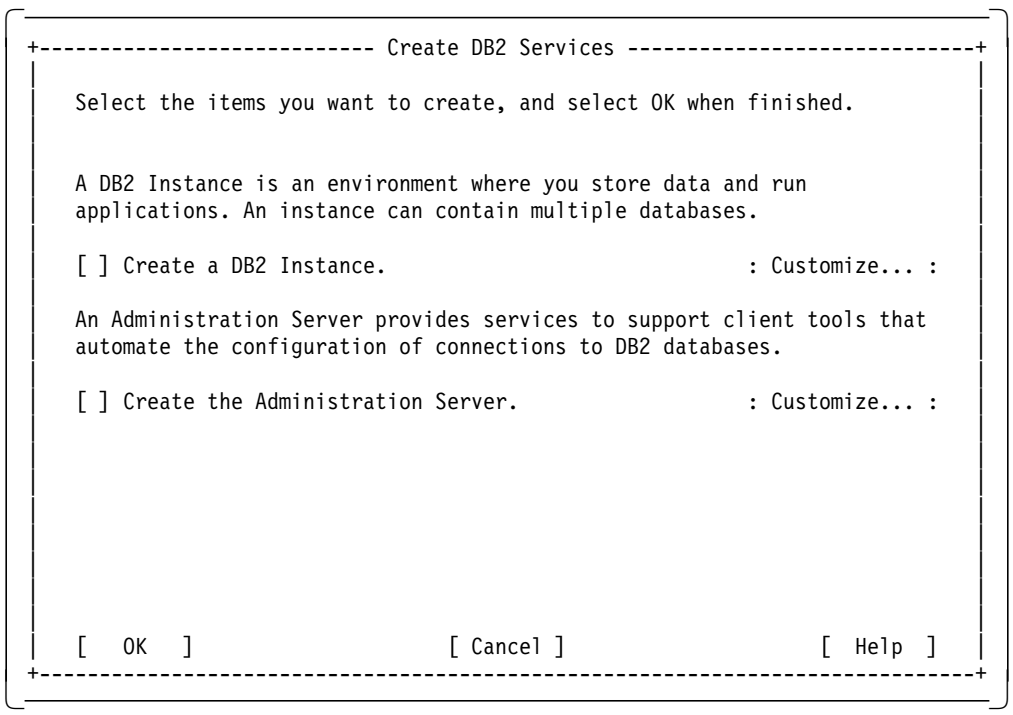
Note: Use the tab key to move to your selection, the space bar to select your choice, and the enter key to activate your selection.



Now select your language for the IBM Product Messages (DB2 error messages) and for the DB2 Product Library (DB2 Books in HTML), as shown in the next screen, and select **OK** to activate the selection.



This will take you to the next screen, which gives you the option to create a new DB2 instance and also a Database Administration Server instance.

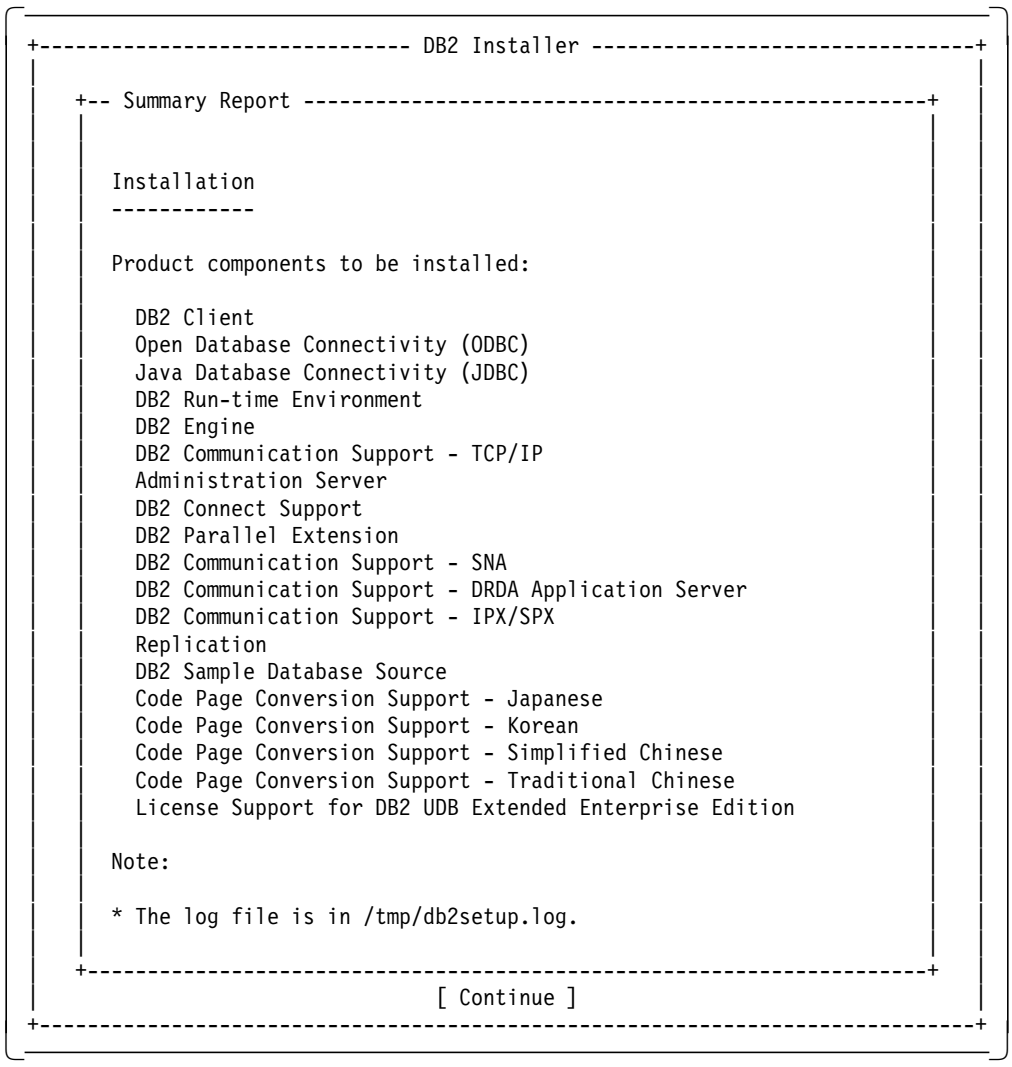


As we will migrate existing instances, you do not need to create a new DB2 instance.

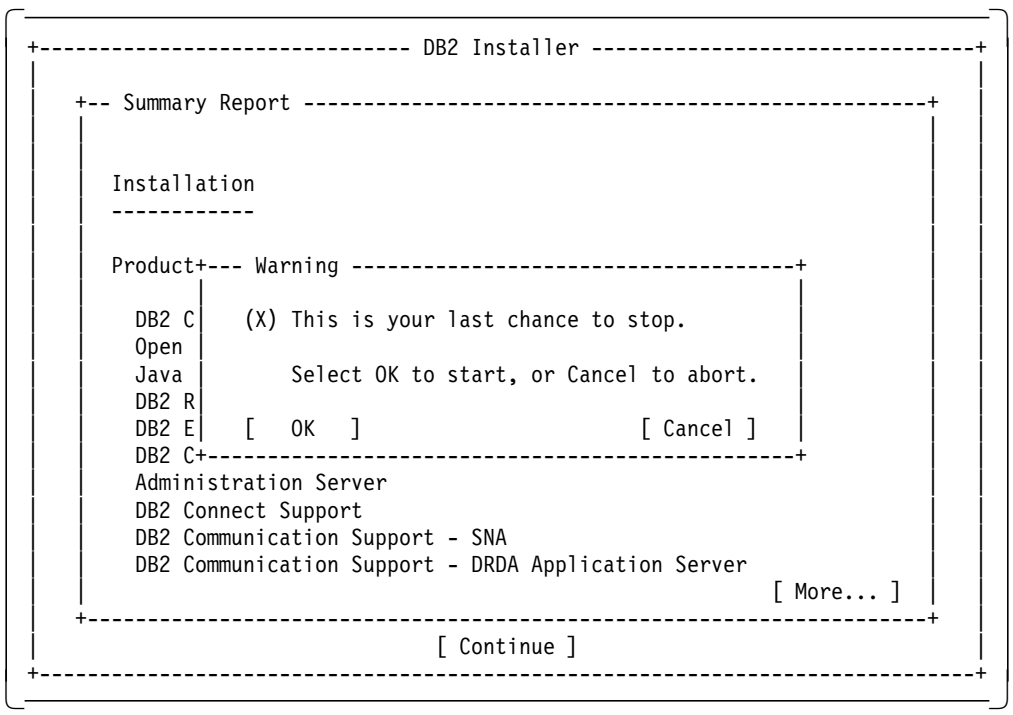
The Administration Server (DAS) instance is used to provide services to support client tools that automate the configuration of connections to DB2 databases, and to enable administration of the DB2 Server system from remote clients.

Note: db2setup will use mkuser and mkgroup to create a user and group for the DAS instance. If you are using File Collections, NIS or Kerberos to manage the replication of users and groups across nodes then you should not use db2setup to create a DAS instance. Instead, see “Installation using SMIT” on page 324 details on how to setup a DAS instance.

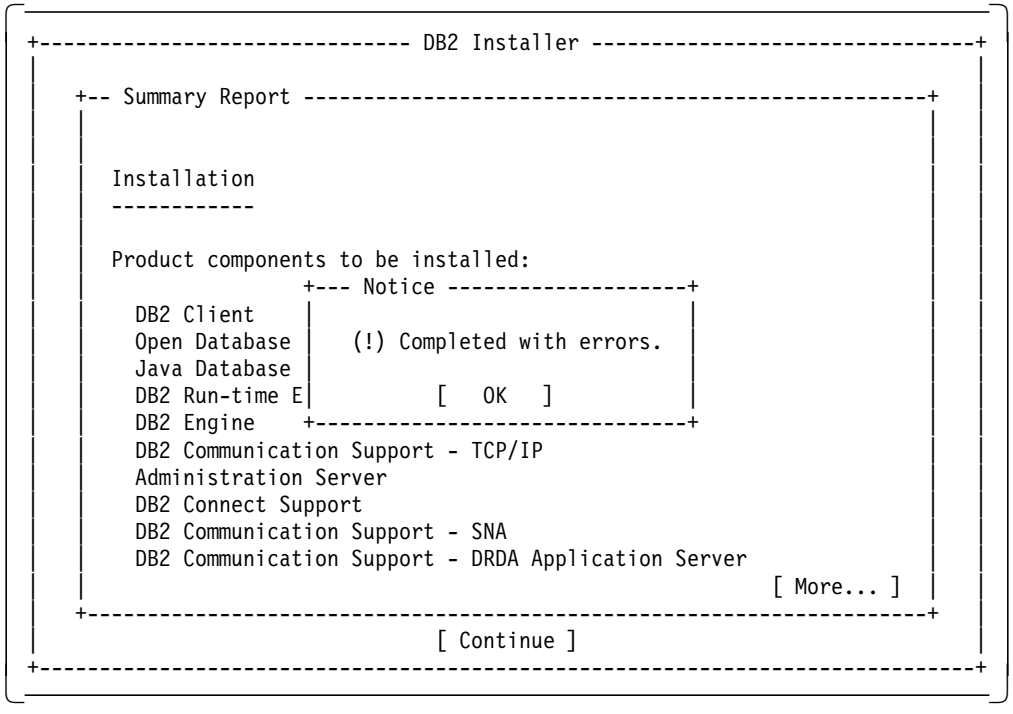
The Summary Report screen appears before the confirmation of the install is requested. The next screen is the result of our example install:



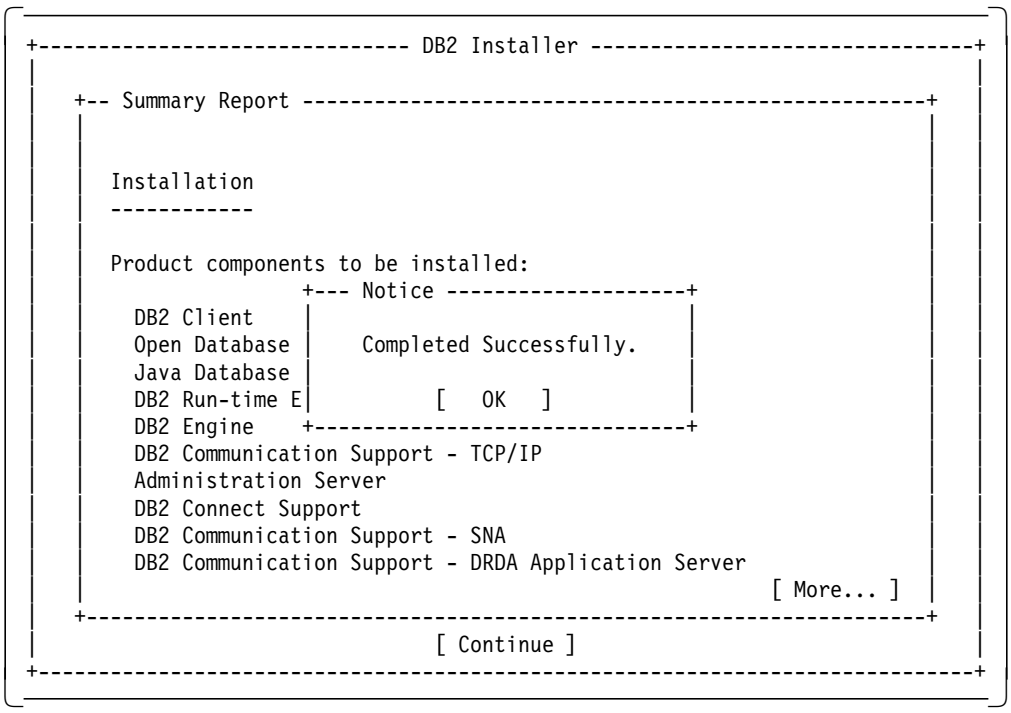
A final warning screen is displayed before the installation is activated:



If any part of db2setup fails, it will complete will the following message. To resolve any problems, look at the log file that db2setup generates in /tmp/db2setup.log.



If db2setup completes successfully, you will see the following message displayed:



After the completion of *db2setup* on one node, the DB2 products will be installed in the */usr/lpp/db2_05_00* directory on that node. You must now repeat the above steps on all nodes (partitions).

Installation using SMIT: You can install DB2 UDB V5 using the SMIT tool. If you choose this method, and want to setup a DAS instance, you should:

- Create a user and group for the DAS instance owner. Set the password for this user, and then distribute these users across all nodes participating in the EEE environment. For example, in a 2-node system:

  ```
  mkgroup db2asgrp; mkuser pgrp=db2asgrp db2as
  passwd db2as
  dsh -w node0,node1 /var/sysman/supper update
  ```

  In this example, these commands should be issued from the SP Control Workstation since we are using File Collections to manage users and groups.

- Run the following command from node 0, where db2as is the DAS instance owner:

  ```
  dasicrt db2as
  ```

It is possible to create additional DAS instances on other nodes in the EEE environment. You might consider this if you wish to distribute the load generated by users performing administration tasks via the Control Center. To do this:

- Create a DAS instance on the other nodes in the EEE environment.
- Catalog the DAS instance as a separate system in the Control Center.
- Under each new system entry, catalog the same DB2 EEE instance. Each time, specify the same host that you used to catalog the DAS instance.

Installation using dsh scripts: For larger systems, you should use the dsh command to automate the installation process. This will avoid having to repeat the installation on each node. For example, to install DB2 UDB EEE on nodes 0 and 1:

```
dsh node0,node1 installp -qagXd /cdrom db2_05_00.xrsv
```

where:

> /cdrom is the mounted filesystem for the CD-ROM containing the DB2 UDB product. This directory must be available on all nodes.

For more details, refer to the *DB2 UDB EEE V5 Quick Beginnings Guide*.

You are now in a position to check and migrate your instance.

7.2.1.3 Using db2ckmig to verify your database

DB2 UDB V5 comes with two utilities db2ckmig and db2imigr, which are used in the migration process. db2ckmig is a utility to check if a given database can be migrated. db2imigr, a utility used to migrate the instance, calls db2ckmig while migrating the instance. It is recommended to run db2ckmig before db2imigr to ensure that you can migrate all the databases in the instance. This section explains how to use db2ckmig to check your databases can be migrated.

*d*b2ckmig checks the instance environment to find out the current DB2 version. It then executes the binary db2ckmig_pe if it sees a DB2/PE instance.

db2ckmig will only check databases that are cataloged in an instance. If your database is not cataloged, db2ckmig will not perform verification tests on those databases.

If you uncatalog those databases that you do not wish to migrate to DB2 UDB V5, the only way to access them after migration is to catalog them in another DB2/PE V1 instance.

To verify that cataloged databases can be migrated, as the *instance owner*, use the following command:

```
/usr/lpp/db2_05_00/instance/db2ckmig <database> -l<logfile> [<user/password>]
```

where:

- <database> can be a database alias as listed in the database directory, or specify all cataloged databases with the -e flag.
- -l <logfile> is a mandatory flag and filename, to specify a file where db2ckmig will place all error output. This file is overwritten each time db2ckmig is invoked.
- <user/password> is optional, specified as -u <userid> -p <password>, and can be used to specify a different user ID and password to access a database.

Note:

- The database migration verification tool, db2ckmig, when used with the -e flag, will attempt to verify all cataloged databases, whether local or remote. If you have a mixture of local and remote databases, with different username and password combinations, be aware that running db2ckmig with the -e flag may log numerous error messages when it attempts to verify the remote databases. If this is the case, you may choose to uncatalog these remote

databases, run the db2ckmig program, and then recatalog these remote databases again so that during installation, the database directories are migrated. You may also choose to run db2ckmig on each database separately, in which case you should check the log file after each iteration.

- There is an undocumented flag, **-h**, which makes db2ckmig check all local cataloged databases. Remote cataloged database are ignored.
- In DB2 UDB V5, databases cannot be cataloged with a mix of authentication types: all databases use the authentication defined at the instance level. If db2ckmig detects mixed authentication types in a DB2/PE V1 instance, it will write the warning AUTHENTICATION MISMATCH to its logfile. db2imigr will still migrate the instance and will change the authentication to that of the DB2 UDB V5 instance, which by default is SERVER.

If there are any errors in the log file, refer to the following for suggested corrective actions.

- A database is in backup pending state:

  Perform a backup of the database.

- A database is in roll-forward pending state:

  Recover the database as required; perform or resume a roll-forward database.

- A database is in database transaction inconsistent state:

  Restart the database to return it to a consistent state.

- The database contains database objects that have a schema name of SYSCAT, SYSSTAT, or SYSFUN.

  These schema names are reserved for the Version 5 database manager. To correct this error, do the following:

  1. Backup the database.
  2. Export the data from the database object (catalogs or tables).
  3. Drop the object.
  4. Recreate the object with the corrected schema name.
  5. Import / Load the data into the object.
  6. Run db2ckmig against the database again, ensuring that the database passes the db2ckmig check.
  7. Make a backup copy of the database.

7.2.2 DB2/PE Systems: Migrating Instances

This section describes the actual migration of your instances. By this stage, you should have installed DB2 UDB V5 EEE and created your Database Administration Server instance if required.

When you run db2imigr to migrate an instance, it will do the following:

- Ensure that the migration of the DB2 version currently being used by the instance is supported.
- Check the authentication types.
- Verify that the fenced user ID information is correct.
- Take a backup of the database manager configuration file, database, node and/or dcs directories, and other useful instance information including the $INSTHOME/sqllib directory.

- Create a new DB2 UDB V5 EEE instance and migrate all files and directories for that instance.
- Installs a userexit and update the DB2 Profiles Registry.

Note:

- You can only migrate the instance as a root user and not the instance user. db2imigr calls db2ckmig with the -h flag prompting it to only check that your local databases are ready for migration.
- If there is not enough disk space in the instance home directory, the instance migration will fail. The only way of recovering from this failure is to drop and recreate the instance at DB2/PE V1, and then restore from a backup copy of your database. Recataloging the database back into the instance will not work as the database directories will also have been partially updated.

7.2.2.1 Create a User ID for Fenced UDFs

A user ID for fenced UDFs and fenced Stored Procedures is mandatory when migrating to DB2 UDB V5. This user is known as the fenced user ID. It is recommended for security reasons that you do not use the instance owner as the fenced user ID. If this user ID is set to the instance owner, then it is very easy to create an application that becomes the instance owner and can perform actions as if it was the DBADM. However, if you do not want to create a fenced user, then you can specify the instance username as the fenced user.

Create the fenced user ID to run fenced UDFs. Set the password for this user, and then distribute these user across all nodes participating in the EEE environment. For example, in a 2-node system:

```
mkgroup db2fadm1; mkuser pgrp=db2fadm1 db2fenc1
passwd db2fenc1
dsh -w node0,node1 /var/sysman/supper update
```

In this example, these commands should be issued from the SP Control Workstation since we are using file collections to manage users and groups.

7.2.2.2 Use db2imigr to Migrate an Instance

To migrate an instance, log in as root and run the following command:

```
/usr/lpp/db2_05_00/instance/db2imigr -u <fenced userid> <instance>
```

where:

- <fenced userid> is the name of the user ID to be used for fenced UDFs and Stored Procedures.
- <instance> is the name of the instance to be migrated.

Note: Because the $INSTHOME is the same for all nodes (NFS mounted) you only need to run the db2imigr command once for all nodes.

When the instance is migrated successfully, the following message will be displayed:

```
# /usr/lpp/db2_05_00/instance/db2imigr -u db2fenc1 inst1
DBI1070I Program db2imigr completed successfully.
```

If the instance migration is unsuccessful, you may see this message:

```
DBI1124E Instance inst1 cannot be migrated.

Cause:  An attempt was made to migrate an instance. This
instance cannot be migrated because:

o   The instance is still active.

o   Migration of this instance is not supported

o   This version of the "db2imigr" command cannot be used
    to migrate this instance.

Action:  Ensure that instance is ready for migration and
you are using the correct version of the "db2imigr"
command. For more information on instance migration, please
refer to the book "Quick Beginnings for UNIX Environments".

DBI1079I Output is saved in the log file /tmp/db2imigr.log.14118.

Cause:  All processed and failed operations have been saved
into this log file.

Action:  Do not modify this file in any way. This file is
for IBM Technical Support reference.
```

If you see this message, you should:

- Check that the instance is not active.
- Check that no processes owned by the instance owner are running.
- Check that no DB2 processes are running.
- Check with db2set -l to make sure that the instance does not already exist in DB2 UDB V5. If it does, then remove connections to all databases in the instance, stop all instance processes and then use the following command to remove it, where inst1 is the name of the instance:

  ```
  /usr/lpp/db2_05_00/instance/db2idrop inst1
  ```

7.2.2.3 Verify that the Instance was successfully migrated

To list the instances that exist in DB2 UDB V5, run:

```
db2set -l
```

Another way to verify that the instance is now at Version 5 is to check the links in the instance user's *sqllib* directory. These should now point to DB2 UDB V5 (db2_05_00) directories. If */home/inst1* is the instance home directory, run:

```
ls -al /home/inst1/sqllib
```

The output should be similar to this:

```
lrwxrwxrwx 1 root     db2adm  Aug 12 doc -> /usr/lpp/db2_05_00/doc
drwxrwsr-t 3 db2inst2 db2adm  Aug 12 function
lrwxrwxrwx 1 root     db2adm  Aug 12 include -> /usr/lpp/db2_05_00/include
lrwxrwxrwx 1 root     db2adm  Aug 12 java -> /usr/lpp/db2_05_00/java
lrwxrwxrwx 1 root     db2adm  Aug 12 lib -> /usr/lpp/db2_05_00/lib
drwxrwsr-t 2 db2inst2 db2adm  Aug 12 log
lrwxrwxrwx 1 root     db2adm  Aug 12 map -> /usr/lpp/db2_05_00/map
lrwxrwxrwx 1 root     db2adm  Aug 12 misc -> /usr/lpp/db2_05_00/misc
lrwxrwxrwx 1 root     db2adm  Aug 12 msg -> /usr/lpp/db2_05_00/msg
lrwxrwxrwx 1 root     db2adm  Aug 12 odbclib -> /usr/lpp/db2_05_00/odbclib
lrwxrwxrwx 1 root     db2adm  Aug 12 samples -> /usr/lpp/db2_05_00/samples
```

This completes instance migration and we are now ready to migrate databases.

7.2.3 DB2/PE Systems: Migrating Databases

After the instance has been migrated successfully, with no errors in the migration log files, you need to individually migrate the databases within that instance. If you try to connect to your database before running the db2 migrate command, you will see the following error displayed:

```
SQL5035N  The database requires migration to the current release.
SQLSTATE=55001
```

7.2.3.1 Use db2 migrate to migrate your databases

To migrate your database, log on as the instance owner, then migrate each database with the following command replacing *sample* with your database name.

```
db2 migrate database sample
```

A successful database migration will result in the following message being displayed:

```
[inst1]$ db2 migrate database sample
DB20000I  The MIGRATE DATABASE command completed successfully.
```

To verify that the database was successfully migrated, try connecting to the database:

```
db2 connect to sample
```

A successful connection will result in the following message being displayed:

```
[jc9003c]inst1(v5)$ db2 connect to sample

   Database Connection Information

 Database product       = DB2/6000 5.0.0
 SQL authorization ID   = INST1
 Local database alias   = SAMPLE
```

7.2.3.2 Restoring the Database from a Backup

Another method to migrate a database is to restore from a database backup made using DB2/PE V1. Be aware that rolling forward of down-level logs is not supported.

7.2.3.3 Database Structure Before and After Migration

There are many changes to the database structure during the migration of a DB2/PE V1 to a DB2 UDB EEE V5 database. These changes affect these database objects: nodegroups, table spaces and containers. For example, if we have a DB2/PE system comprising of:

- An instance called inst1 defined on node0 and node1.
- A database called TPCD using three segment directories in the path /MIG.
- A Table called lineitem defined in the default nodegroup, IBMDEFAULTGROUP.

Figure 181 is a logical view of the database before migration:

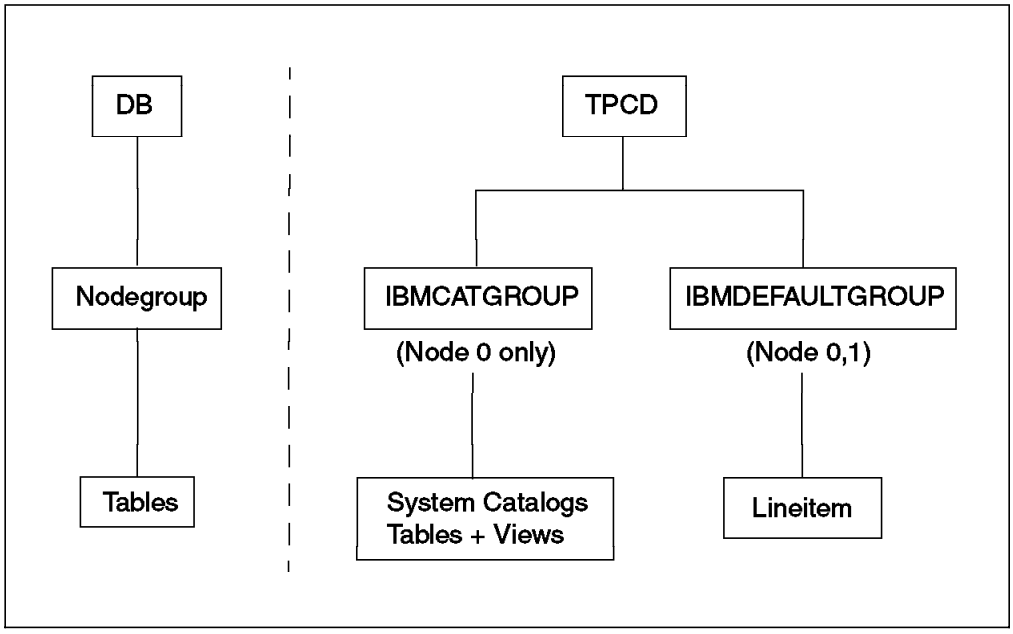


Figure 181. Logical View before Migration

The next diagram, shown in Figure 182 on page 331, shows how DB2/PE stores the files that make up the database:

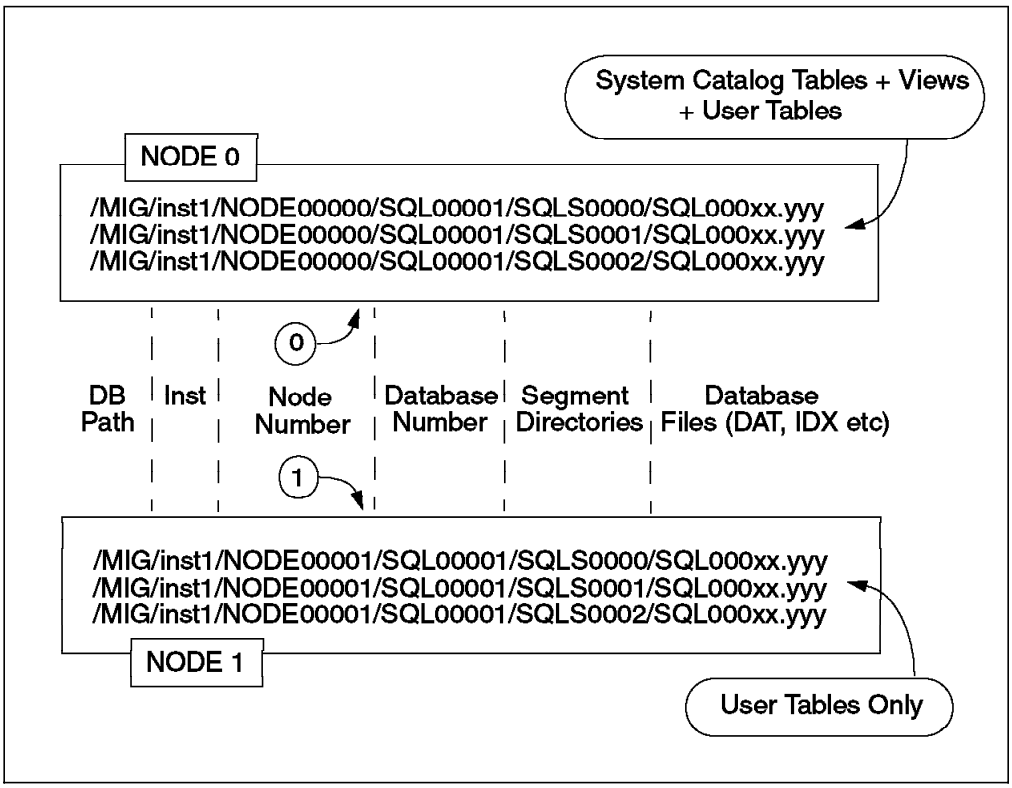


Figure 182. Physical View before Migration

Figure 183 is a logical view of the database after migration:

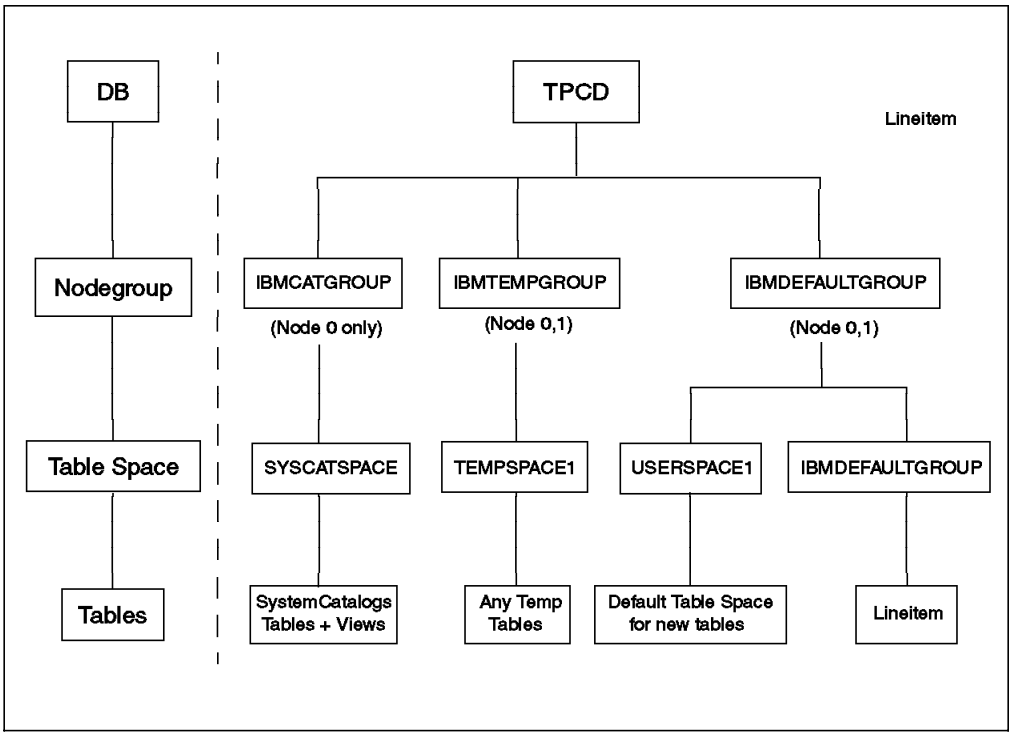


Figure 183. Logical View after Migration

Note the introduction of table spaces which are defined within nodegroups. In summary:

- The system catalogs are migrated and placed in the SYSCATSPACE table space (SMS) in the IBMCATGROUP nodegroup (catalog node only).
- The tables that existed in the IBMDEFAULTGROUP nodegroup are migrated into a table space called IBMDEFAULTGROUP (SMS) in the IBMDEFAULTGROUP nodegroup.
- A new table space called USERSPACE1 (SMS) is created in the IBMDEFAULTGROUP nodegroup. This is the default table space for new tables.
- A new nodegroup, IBMTEMPGROUP (across all nodes) is created and a new table space TEMPSPACE1 (SMS) is created in this nodegroup. This is for temporary objects.

Note: Any user-defined nodegroups will be migrated in a similar way to the nodegroup IBMDEFAULTGROUP:

- A nodegroup will be created with the same name
- An SMS table space will be created in this nodegroup using the same name as the nodegroup.
- All tables that existed in the DB2/PE nodegroup will be migrated into this table space.

A physical view of the database after migration is shown in Figure 184 on page 333:

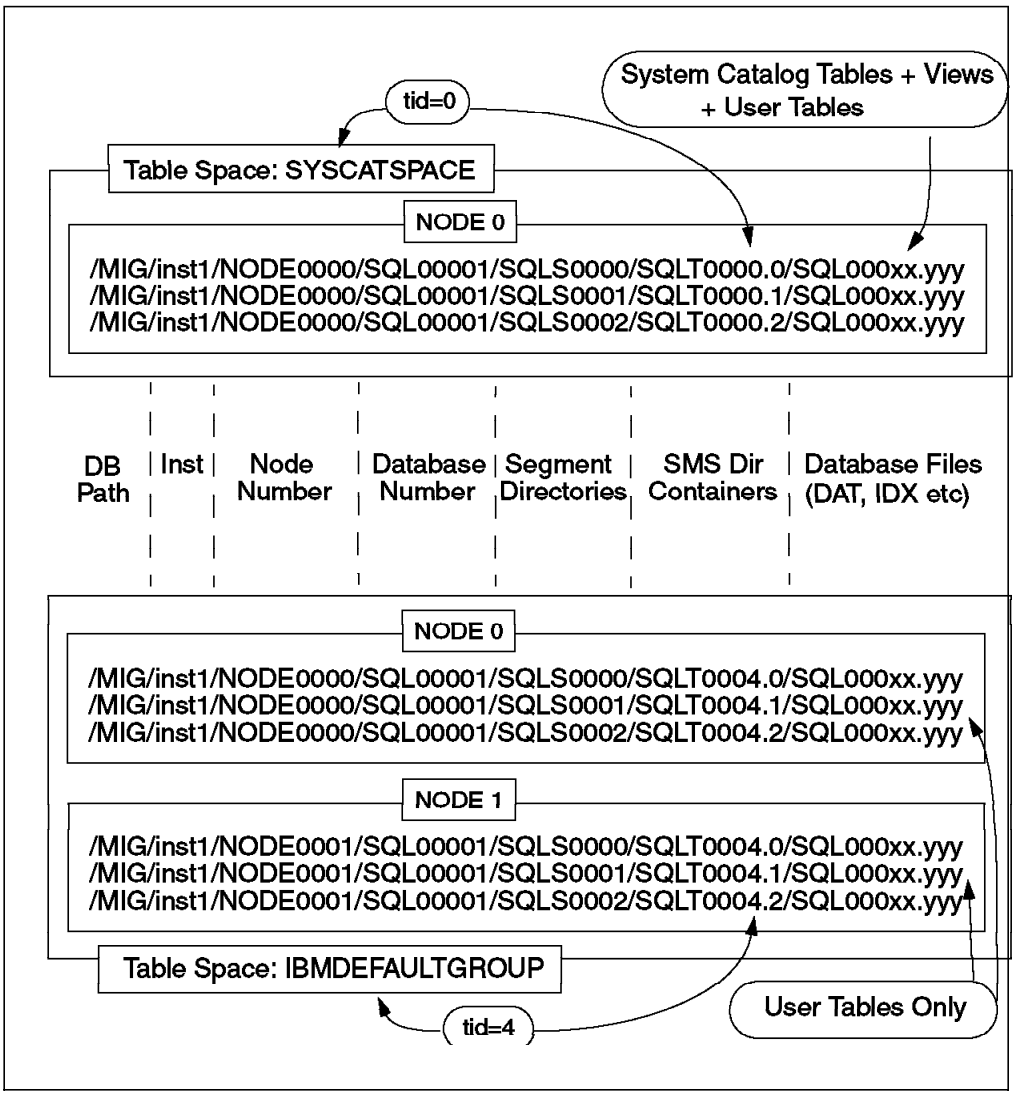


Figure 184. Physical View after Migration

Here we have shown details on two of the new table spaces, SYSCATSPACE and IBMDEFAULTGROUP. For the each table space:

- Since there were 3 segment directories in DB2/PE, three SMS directory containers are created. The directory path is made up of the entire path used by the DB2/PE segment directory plus an additional directory SQLTxxxx.y, where *xxxx* is the table space ID, and *y* is the number of the DB2/PE segment directory.
- The directory representing the node number changes from NODEnnnnn to NODEnnnn (5 digits changing to 4 digits). Note that if the segment directories were mounted JFS file systems in DB2/PE, the names of these file systems will be changed during the database migration.

7.3 Moving Data from SMS to DMS Table Spaces

This section will discuss the possible methods for getting data into DMS tablespaces for databases that have been migrated from DB2/PE. After database migration using the db2 `migrate database` command, the resulting database uses SMS tablespaces. This transition is optional. You may choose to leave their SMS table spaces since they are created after database migration.

The relative merits of SMS and DMS table spaces are explained in detail in 3.1, “Storage Objects” on page 79.

To summarize the key differences between DMS and SMS table spaces which are important for partitioned databases:

- DMS table spaces:
  - Allow faster insert performance, especially for disk-bound operations such as insert subselect.
  - Allow slightly faster select (read) performance compared to SMS.
  - can be dynamically increased in size by adding containers.
  - Allow for the separation on data, index and long data.
- SMS table spaces:
  - Are easier to administer than DMS. This is especially true if you choose to allocate one table per table space.
  - Do not have their disk space pre-allocated.
  - Can only be increased in size by increasing the underlying file systems or by performing a backup and then a redirected restore.

When moving to DMS table spaces, we should first consider how the tables are being used in the system. DB2/PE tables can be considered to be either:

1. Reloadable tables

- Data is loaded from an external source, usually mainframe.
- Data is not backed up.
- It is relatively easy to redefine disk allocation to use DMS file or raw containers and load using normal methods.
- 2. Incremental tables
- Data is added regularly (nightly/weekly).
- Data is backed up regularly (very often using ADSM and mainframe tape drives).
- It is difficult to redefine the disk allocation since most disk space is already taken in the existing file systems.

If the database consists only of reloadable tables, then the move to DMS table spaces is relatively simple. However if the database has a least some incremental tables then this task becomes more complex, because:

- A backup taken of a database that uses SMS tablespaces can not be restored with a redefine option to use DMS tablespaces.
- Export is relatively slow. There is no fast unload (the opposite of load).

It is common to see most of the disk space taken up for the DB2/PE system, so there is the problem of space to consider. For instance, a typical configuration would have, perhaps, eight mounted filesystems for the segment directories, each one defined as 2 GB, with the file systems being 30-50% full, plus some other disk space used to manage the incremental inserts, and so on. Often there is not the option of having new DMS raw containers co-existing with the DB2/PE file systems due to lack of space.

If the DB2/PE system uses mirrored disks, however, there exists the option to break the mirror during the move from SMS to DMS. This would free up a considerable amount of space while obviously increasing the likelihood of problems due to disk failure.

The methods covered in detail are:

- *Export Reload* - can be used to move data into DMS raw containers.
- *I*nsert Subselect - moves data into a DMS tablespace defined with file containers.
- Redirected Restore - an extra step to be added to the above so that data can be moved into DMS tablespaces defined with raw containers.

The Test System: Tests were carried out on an RS/6000 SP system using:

- A 2 node instance called INST1.
- A database called TPCD defined with 3 segment directories on /MIG. Each segment directory was then reconfigured to be a mounted filesystem.
- A table called LINEITEM in the default nodegroup IBMDEFAULTGROUP.

The TPCD database has been migrated to DB2 UDB EEE V5 database, and therefore is using SMS tablespaces.

7.3.1 SMS to DMS: Export Reload

Summary: This method is the most direct way to get data into DMS table spaces. It does, however, require that data can be exported to either disk or other media (such as ADSM). If mirrored disks are being used, the mirror may be broken during migration to free up disk space. For large databases, the time to export data can be significant.

Description:

- Export all tables. The exported datafiles are stored on disk or other media (such as ADSM).
- Drop all tables.
- Reallocate disk space:
  - Create file systems for the system catalogs and temp table spaces.
  - Create logical volumes to be used as raw containers.
- Recreate the tables that were dropped in the new DMS table spaces.
- Use the Load utility to reload the data back into DMS table spaces

Process:

1. Disconnect all applications and then take a full database backup. In this example we backup to disk; most users who have a backup system in place

use ADSM. Note that the catalog node must finish before other nodes can start.

On Node 0: `db2 "backup db TPCD to /BK"`

When this is complete:

On Node 1: `db2 "backup db TPCD to /BK"`

2. Table by table, export each table's data to flat files. For example, using LINEITEM:
   - Drop the indexes defined on LINEITEM:

     Note: This will free up a significant amount of space when dealing with the larger tables. For example the indexes on lineitem take up 20% of the space that lineitem data uses.
   - Export the table's data. If the table is not large (< 50,000 rows) you can use a simple export:

     ```
     db2 "export to /exported/lineitem.ixf of ixf select * from lineitem"
     ```

     In this example the exported datafile is stored on disk and we can use the Import utility later to reload the table. For larger tables we must use the Load utility to reload the data. In preparation for the Load, we must export the data by partition by using a WHERE CURRENT NODE = NODENUMBER() clause on the select in the export. For example, to write one output file per partition, run:

     On Node 0:

     ```
     db2 "export to /exported/lineitem.del.00000 of del modified by coldel|
     select * from lineitem where current node = nodenumber(l_orderkey)"
     ```

     On Node 1:

     ```
     db2 "export to /exported/lineitem.del.00001 of del modified by coldel|
     select * from lineitem where current node = nodenumber(l_orderkey)"
     ```

     Output Files Greater than 2 GB:

     It is important to consider the size of the output file. Even though AIX itself may support files greater than 2 GB in size, the export utility cannot yet write files greater than 2 GB.

     If there is more than 2 GB of data per node, then you can add another WHERE clause to the select to reduce the amount of data selected. However, this is potentially hazardous. You risk making a mistake and not exporting all the data. You will also have to scan the table many times.

     Another possibility is to export into a named pipe and have the Load utility read from this pipe, thus circumventing the 2 GB limit entirely. Be aware that:

     - This would do away with the requirement to store the output file on disk or other media. In a multi-partition database, load requires a header (as generated by db2split) to identify the partition (node) where the data must be loaded. The above Export command will not generate the header data that Load requires.
     - The export utility always creates a new output file so even if we create the header data in the output file (named pipe), export will not append the table's data after the header data.

A possible solution to this problem is to write a program in C that first reads the header data from a file (which has been created previously), writes the header data to a named pipe, and then performs a select via a cursor that reads all the rows for a particular node and appends the row data to the named pipe. This was tested and does work. However, the best solution in terms of performance and simplicity to the 2 GB limit problem was found to be the insert subselect method.

- Once the export process has been completed you can drop the table. Keep in mind that you have the backup in case of problems.

3. Disconnect all connections and then drop the database:

   ```
   db2 drop database TPCD
   ```

4. Remove the filesystems used by the database. In our case:

   On node 0:

   ```
   /MIG/inst1/NODE0000/SQL00001/SQLS0000
   /MIG/inst1/NODE0000/SQL00001/SQLS0001
   /MIG/inst1/NODE0000/SQL00001/SQLS0002
   ```

   On node 1:

   ```
   /MIG/inst1/NODE0001/SQL00001/SQLS0000
   /MIG/inst1/NODE0001/SQL00001/SQLS0001
   /MIG/inst1/NODE0001/SQL00001/SQLS0002
   ```

   As root, do the following steps for each file system. We will take the first file system as an example:

   - Unmount the file system:

     ```
     umount /MIG/inst1/NODE0000/SQL00001/SQLS0000
     ```

   - Remove the file system:

     ```
     rmfs /MIG/inst1/NODE0000/SQL00001/SQLS0000
     ```

5. Create file systems for the system catalogs and temp table space:

   - Use SMIT to create the file system /MIG. You should plan for up to 10 MB of space for the system catalogs. The file system is only needed on node 0:

     ```
     smit jfs
     ```

   - Mount the file system:

     ```
     mount /MIG
     ```

   - Give read and write permission to the instance owner:

     ```
     chown inst1 /MIG
     chmod 755 /MIG
     ```

   - Perform a similar process to create file systems for the temp table space. We created one file system per disk on each node. We decided to allow for the size of the largest table, spread over the file systems and nodes

6. Create the database:

   ```
   create database TPCD on /MIG numsegs 1
   ```

   This, by default, will create a temporary (SMS) table space, TEMPSPACE1 defined on this single file system. It is recommended that you create a new temporary (SMS) table space which uses the file systems defined in the above step. Then you should drop TEMPSPACE1.

7. Create logical volumes (LVs) in AIX for the DMS raw containers. In our example, we created 3 LVs per node as we had 3 disks per node. The names of the LVs (such as dblv1, dblv2, dblv3) should be the same for all nodes. As root, do the following steps for each LV:

   - Create a LV using SMIT or mklv. For example, to create a LV called dblv1 in volume group vg1 of size 400 MB (100 4 MB phyical partitions) on hdisk2:

     ```
     mklv -ydblv1 vg1 100 hdisk2
     ```

   - Give universal read and write permissions to the special device file for the LV. This file is called /dev/r<name of LV>:

     ```
     chmod 666 /dev/rdblv1
     ```

8. Once this process is completed for all the LVs, create a new DMS table space (DMS1) with one raw container per disk.

   ```
   create tablespace DMS1 in nodegroup IBMDEFAULTGROUP managed by database
    using (device '/dev/rdblv1' 100000,
           device '/dev/rdblv2' 100000,
           device '/dev/rdblv3' 100000)
   ```

9. Recreate each table in the newly created table space. For example, for LINEITEM:

   ```
   create table LINEITEM             ( l_orderkey   integer not null,
     l_partkey       integer not null, l_suppkey    integer not null,
     l_linenumber    integer not null, l_quantity     float not null,
     l_extendedprice   float not null, l_discount     float not null,
     l_tax             float not null, l_returnflag char(1) not null,
     l_linestatus    char(1) not null, l_shipdate      date not null,
     l_commitdate      date not null, l_receiptdate   date not null,
     l_shipinstruct char(25) not null, l_shipmode  char(10) not null,
     l_comment   varchar(44) not null)
   in DMS1
   ```

10. Before loading the data back into the tables you will need to prepend the header data to each exported datafile. The simplest way to create the header data is to run db2split with the correct configuration file for each table but supplying an empty data file. For example, for the LINEITEM table, run:

    ```
    db2split -c lineitem.cfg
    ```

    specifying in the lineitem.cfg file:

    ```
    infile=emptyfile
    ```

    Two files are produced, lineitem.00000 and lineitem.00001. These files contain the header data for partitions 0 and 1 respectively. To prepend these files to their respective exported datafiles, run:

    On Node 0:

    ```
    cat lineitem.00000 lineitem.del.00000 > lineitem.load.00000
    rm lineitem.del.00000
    ```

    On Node 1:

    ```
    cat lineitem.00001 lineitem.del.00001 > lineitem.load.00001
    rm lineitem.del,00001
    ```

11. For each table, run the LOAD command to reload the its data. For example, for LINEITEM:

On Node 0:

```
db2 load from lineitem.load.00000 of del modified by coldel| fastparse
replace into lineitem
```

On Node 1:

```
db2 load from lineitem.load.00001 of del modified by coldel| fastparse
replace into lineitem
```

Note: Since we are certain of the data's validity, we can use the new *fastparse* Load option to improve the performance of the load process.

12. Recreate the indexes on LINEITEM.
13. Do runstats on LINEITEM.
14. Once this process has been completed for all the tables that you wish to move to DMS table spaces, perform another database backup.

7.3.2 SMS to DMS: Insert Subselect

Summary: If disk space is limited, this method enables you to move a table at a time from SMS to DMS table spaces while requiring minimal free disk space. It is recommended for tables that are already well reorganized with respect to their preferred indexes.

Description:

- For each table, create a temporary table with the same description but defined in a DMS tablespace that uses containers spread over the existing file systems used by the SMS directory containers.
- Using insert subselect, copy the data from the original table to the temporary version.
- Verify that the insert has worked, then drop the original table and rename the temporary version to the original name.

The diagram shown in Figure 185 on page 340 illustrates the steps used in the insert subselect method. One of the file systems used by the database is taken as an example; a similar process will take place across the other file systems.

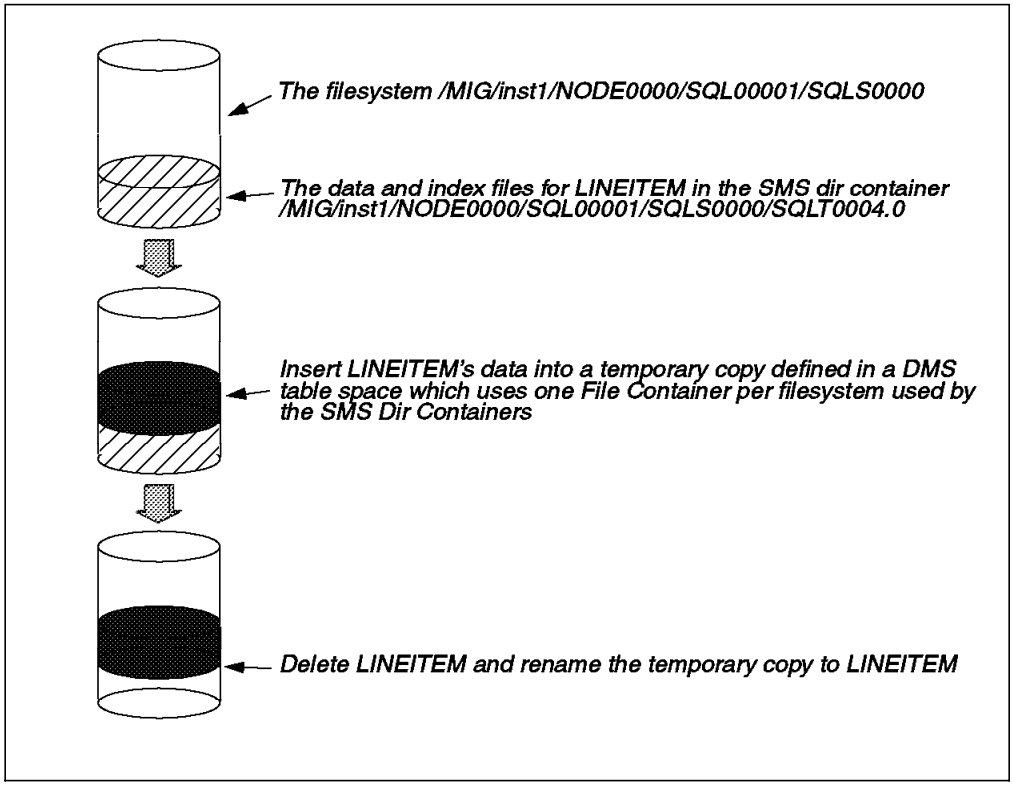


Figure 185. Insert Subselect Method

Process:

1. Backup the database in case of problems.
2. Table by table, perform the following steps: (for example, using LINEITEM)
   - Drop indexes defined on lineitem.

     Note: This will free up a significant amount of space when dealing with the larger tables. For example the indexes on lineitem take up 20% of the space that lineitem data uses.
   - Create a new DMS table space (DMS1) with one file container in each of the filesystems used by the SMS directory containers for the table.

```
create tablespace DMS1 in nodegroup IBMDEFAULTGROUP managed by database
 using (file '/MIG/inst1/NODE0000/SQL00001/SQLS0000/dms1' 100000,
        file '/MIG/inst1/NODE0000/SQL00001/SQLS0001/dms2' 100000,
        file '/MIG/inst1/NODE0000/SQL00001/SQLS0002/dms3' 100000)
on node (0)
 using (file '/MIG/inst1/NODE0001/SQL00001/SQLS0000/dms4' 100000,
        file '/MIG/inst1/NODE0001/SQL00001/SQLS0001/dms5' 100000,
        file '/MIG/inst1/NODE0001/SQL00001/SQLS0002/dms6' 100000)
on node (1)
```

   - Create a new table, LINETEMP in this DMS tablespace (DMS1). Also use the NOT LOGGED INITIALLY clause to avoid any logging. In the same unit of work, do INSERT INTO LINETEMP SELECT * FROM LINEITEM:

```
update command options using c off;

create table LINETEMP             ( l_orderkey   integer not null,
  l_partkey       integer not null, l_suppkey    integer not null,
  l_linenumber    integer not null, l_quantity     float not null,
  l_extendedprice   float not null, l_discount     float not null,
  l_tax             float not null, l_returnflag char(1) not null,
  l_linestatus    char(1) not null, l_shipdate      date not null,
  l_commitdate      date not null, l_receiptdate   date not null,
  l_shipinstruct char(25) not null, l_shipmode  char(10) not null,
  l_comment   varchar(44) not null)
in DMS1
NOT LOGGED INITIALLY;

insert into linetemp
select * from lineitem;

commit work;
```

- At this point, the contents of LINETEMP should be verified, either with a simple ″select count(*) from linetemp″ or a select statement which selects data and can be checked against the original LINEITEM table.
- Drop the original table and rename the new table (LINETEMP) to the original (LINEITEM).

```
drop table lineitem;
rename table linetemp to lineitem;
```

- Recreate the indexes on LINEITEM.
- Do runstats on LINEITEM.

3. Once this process has been completed for all the tables that you wish to move to DMS table spaces, perform another database backup.

Note:

- The speed of insert subselect when inserting into DMS tablespaces is greatly improved compared to DB2/PE. This is due to the fact that:
  - DMS file containers are pre-allocated. DB2/PE or UDB with SMS tablespaces must repeatedly make expensive calls to AIX to increase the size of the .DAT file. Even when using the MULTIPAGE_ALLOC option for this operation, DMS is considerably faster.
  - Insert subselect is disk intensive, so improvements in the efficiency of writing the output data make a large difference in the elapsed time. (This is in contrast to LOAD or IMPORT which are very CPU intensive).
- The maximum space required is the size of the largest table, spread over the number of filesystems that were defined as segment directories in DB2/PE. This is roughly equivalent to the amount of space which would have been necessary to REORG the table in DB2/PE. Also, by dropping the indexes before moving the data, a significant amount of space will be freed.

DMS Raw Containers: This example uses DMS file containers which make use of the disk space already allocated to the SMS directory containers and so does not need any additional disk space. To move to DMS raw containers requires that you have enough unallocated disk space to define as raw containers and store all the tables you wish to move to raw containers. If mirrored disks are being used, the mirroring may be turned off during the migration to free up the necessary disk space.

7.3.3 SMS to DMS: Redirected Restore

Summary: This method is really an additional step after completing the *insert subselect* method above. It allows for a transition from DMS file containers to DMS raw containers using backup and redirected restore. It is recommended to keep both the system catalog table space (SYSCATSPACE) and any temp table spaces (such as TEMPSPACE1) as SMS table spaces.

Note: The process for this method only uses the disk space that was previously allocated to the file systems used as segment directories in DB2/PE. If you have a lot of free disk space available (enough to allocate to new logical volumes to hold a second copy of the database) then you should consider the *export reload* and *insert subselec*t methods. These methods are more direct ways of moving data to DMS raw containers.

Description:

1. Backup the database.
2. Reduce the size of the file systems allocated to the SMS table spaces.
3. Create logical volumes using the disk space freed by the above step.
4. Perform a redirected restore so that the table spaces holding user tables are defined to use raw containers. These raw containers use the logical volumes created in the above step.

The diagram shown in Figure 186 illustrates the steps used in the redirected restore method. One of the disks used by the database is taken as an example; a similar process will take place across the disks.

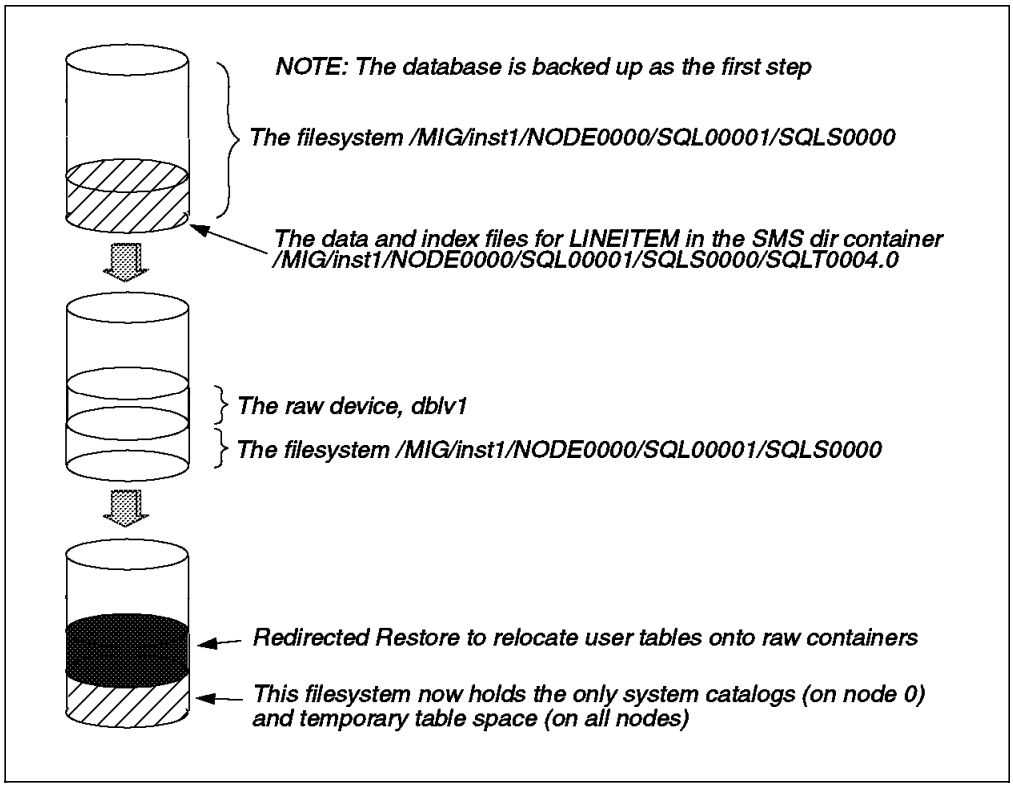


Figure 186. Redirected Restore Method

Process:

1. You have to set archive log mode (logretain or userexit). If it is not set, the restore redirect fails. You'll need it for the restore redirect later. Set the archive log mode on all nodes as following command:

   ```
   db2_all db2 "update db cfg for TPCD using logretain on"
   ```

2. Disconnect all applications and then take a full database backup. In this example we backup to disk; most users have a backup system in place use ADSM. Note that the catalog node must finish before other nodes can start.

   On Node 0: `db2 "backup db TPCD to /BK"`

   When this is complete:

   On Node 1: `db2 "backup db TPCD to /BK"`

3. Stop the database manager on all nodes:

   ```
   db2stop
   ```

4. Given that we will keep the SYSCATSPACE and TEMPSPACE1 table space as SMS, to free up space we will reduce the size of the file systems used by these table spaces. in our example, SYSCATSPACE and TEMPSPACE1 use these file systems:

   On node 0:

   ```
   /MIG/inst1/NODE0000/SQL00001/SQLS0000
   /MIG/inst1/NODE0000/SQL00001/SQLS0001
   /MIG/inst1/NODE0000/SQL00001/SQLS0002
   ```

   On node 1:

   ```
   /MIG/inst1/NODE0001/SQL00001/SQLS0000
   /MIG/inst1/NODE0001/SQL00001/SQLS0001
   /MIG/inst1/NODE0001/SQL00001/SQLS0002
   ```

   As root, do the following steps for each file system. We will take the first file system as an example:

   - Unmount the file system:

     ```
     umount /MIG/inst1/NODE0000/SQL00001/SQLS0000
     ```

   - Remove the file system:

     ```
     rmfs /MIG/inst1/NODE0000/SQL00001/SQLS0000
     ```

   - Use SMIT to recreate the file system at a smaller size. You should plan for up to 10 MB of space for the system catalogs (only node 0). For the amount used by TEMPSPACE1, we decided to allow for the size of the largest table, spread over the file systems and nodes.

     ```
     smit jfs
     ```

   - Mount the file system:

     ```
     mount /MIG/inst1/NODE0000/SQL00001/SQLS0000
     ```

   - Give read and write permission to the instance owner:

     ```
     chown inst1 /MIG/inst1/NODE0000/SQL00001/SQLS0000
     chmod 755 /MIG/inst1/NODE0000/SQL00001/SQLS0000
     ```

5. Create logical volumes (LVs) in AIX for the DMS raw containers. In our example, we created 3 LVs per node as we had 3 disks per node. The names of the LVs (such as dblv1, dblv2, dblv3) should be the same for all nodes. As root, do the following steps for each LV:

- Create a LV using SMIT or mklv. For example, to create a LV called *dblv1* in volume group *vg1* of size 400 MB (100 4 MB phyical partitions) on *hdisk2*:

  ```
  mklv -ydblv1 vg1 100 hdisk2
  ```

- Give universal read and write permissions to the special device file for the LV. This file is called */dev/r<name of LV>*:

  ```
  chmod 666 /dev/rdblv1
  ```

6. Now that the disk space has been reallocated we are ready to perform the redirected restore. There are three commands required to perform this operation:

   - Restore with the redirect option. This puts the table space into a ″ready for containers to be redefined″ state.
   - Set table space containers. This redefines the containers for a table space.
   - Restore with the continue option. This resumes the restore with the new containers.

   These steps must all execute in one CLP session. The restore redirect will fail archival logging is not enabled (using logretain or userexit). You will need to get the table space ID for the table space whose containers are to be redefined (via list tablespaces). In this example, the table space ID for DMS1 is 3.

   ```
   restore db TPCD tablespace (DMS1) from /BK to /MIG redirect;

   set tablespace containers for 3 using (
   device '/dev/rdblv1' 100000,
   device '/dev/rdblv2' 100000,
   device '/dev/rdblv3' 100000 );

   restore db TPCD continue;
   ```

   You should see messages similar to this:

   ```
   SQL1277N  Restore has detected that one or more table space containers are
   inaccessible, or has set their state to 'storage must be defined'.
   DB20000I  The RESTORE DATABASE command completed successfully.

   DB20000I  The SET TABLESPACE CONTAINERS command completed successfully.

   SQL2563W  The restore process has completed successfully, but one or more
   table spaces from the backup were not restored.
   ```

   Note: You need to run these commands at each node.

7. After the successful completion of the Redirected Restore, the affected table space will be left in rollforward pending state. You should run this command on only the catalog node:

   ```
   db2 rollforward db TPCD to end of logs and stop
   ```

7.4 Post-Migration

There are several optional activities you may wish to undertake following database migration:

- **Unique index conversion to DB2 Universal Database Version 5 semantics**

  Version 5 of DB2 supports deferred unique constraint checking until end of statement. This can result in correct processing of multiple row updates, that in previous releases of DB2, returned an error because the updates temporarily created duplicate values in the transient state. Deferred unique constraint checking will guarantee that updates, that result in a table with only unique keys (for example, key = key + 1) will succeed regardless of the order of the data.

  Note: This change only applies for unique indexes that are created in DB2 UDB V5.

  All unique indexes, in a migrated database, do not automatically migrate to Version 5 semantics during database migration because of the following reasons:

  – Converting unique indexes is a very time-consuming operation. You may have applications that depend on the previous version's unique index semantics. You may wish to manage the staged conversion of unique indexes on your own schedule, when needed.

  – All existing applications will continue to work even if the unique indexes are not converted to Version 5 semantics. You have to convert unique indexes to Version 5 semantics only if support for deferred uniqueness checking is required.

  To convert unique indexes, you need to perform the following steps:

  1. Log on with a user ID that has SYSADM authority.
  2. Issue the `db2start` command.
  3. Run the `db2uiddl` command against your migrated database. The syntax of this command is as follows:

     ```
     db2uiddl -d <database-name> [<-u table-schema>] [<-t table-name>]
     [<-f filename>] [<-h help>]
     ```

     where:

  – `<database-name>` is the name of the database to be queried.

  – `<-u table-schema>` is optional, and can be used to specify the schema (creator user ID) of the tables that are to be processed. The default action is to process tables created by all user IDs.

  – `<-t table-name>` is optional, and can be used to specify the name of a table that is to be processed. The default action is to process all tables.

  – `<-f filename>` is optional, and can be used to specify the name of a file to which output is to be written. The default action is to write output to standard output.

  – `<-h help>` can be used to display help information. When this option is specified, all other options are ignored, and only the help information is displayed.

    Note: This tool was not designed to handle certain types of names. If a specific table name or table schema is a delimited identifier containing

lower case characters, special characters, or blanks, it is preferable to request processing of all tables or schemas. The resulting output can be edited.

The db2uiddl command searches the database catalog tables and generates all the CREATE UNIQUE INDEX statements for user tables in an output file.

1. Review the output generated from the db2uiddl command, and make changes, if needed. Comments in the output will flag any situations that require your attention.
2. Execute the file as a DB2 Command Line Processor command file, using a command similar to the following:

   ```
   db2 -tvf filename
   ```

   where filename is output file from db2uiddl.

DB2 interprets the recreation of an already-existing unique index to signal that the index is ready to be converted to Version 5 semantics.

- Update Statistics

  When database migration is completed, the old statistics, used to optimize query performance, are retained in the catalogs. However, Version 5 of DB2 has statistics that are modified or do not exist in the previous version. To take advantage of these, you may wish to issue the runstats command on tables, particularly those tables that are critical to the performance of your SQL queries.

  Refer to the *DB2 UDB V5 Command Reference* for the syntax of the runstats command. For details on the statistics, refer to the *DB2 UDB V5 Administration Guide*.

- **Rebind Packages**

  During database migration, all existing packages are invalidated during catalog migration. After the migration, each package is rebuilt when it is used for the first time by the Version 5 database manager.

  However, for better performance, we recommend that you run the db2rbind command to rebuild all packages stored in the database, after database migration is complete. The following gives an example using sample as our database:

  ```
  db2rbind sample -l /tmp/db2rbind.log
  ```

  Refer to the *DB2 UDB V5 Command Reference* for more information about this command.

- **Update database and database manager configuration**

  Some of the database configuration parameters are changed to Version 5 defaults or to other values during database migration. The same is true for database manager configuration parameters that may have changed, during instance migration, to Version 5 defaults or to other values. For better performance, you may wish to tune your database and database manager configuration parameters to take advantage of Version 5 enhancements. Refer to the *DB2 UDB V5 Command Reference* for the syntax of updating database and database manager configuration.

  The following database manager configuration parameters are changed to Version 5 defaults:

– Database configuration release level

  This is changed to 0x0800.

– Sort Heap threshold (SHEAPTHRES)

  If the migrating database configuration file has this parameter at a value that is less than the Version 5 Default, the parameter is reset to its Version 5 default value of 10000 x 4 K pages.

– Backup buffer (BACKBUFSZ)

  If the migrating database configuration file has this parameter at a value that is less than the Version 5 default, the parameter is reset to its Version 5 default value of 1024 x 4 K pages.

– Restore Buffer (RESTBUFSZ)

  If the migrating database configuration file has this parameter at a value that is less than the Version 5 default, the parameter is reset to the default value of 1024 x 4 K pages.

– Number of concurrent databases running against the database manager (NUMDB)

  If the migrating database configuration file has this parameter at a value that is less than the Version 5 default, the parameter is reset to its Version 5 default value, as follows:

Type of Database	Number
Database server with local and remote clients	8
Database server with local clients	3

– Transaction Manager database name (TM_DATABASE)

  If the migrating database configuration file has this parameter set to NULL, the parameter is reset to 1ST_CONNECT (1ST_CONN).

– Query heap size (QUERY_HEAP_SZ)

  If the migrating database configuration file has this parameter at a value less than the Application Support Layer heap size (ASLEAPSZ) of the same file, the parameter is reset to ASLEAPSZ + 1. The default value is 1000 x 4 K pages.

– Maximum number of idle agents (MAX_IDLEAGENTS)

  If the migrating database configuration file has this parameter greater than the Maximum Simultaneous agents (MAXAGENTS) of the same file, the parameter is reset to MAXAGENTS -1. See the NUM_INITAGENTS and MAXAGENTS parameters for further information.

– TCP/IP Service Name (SVCENAME)

  If the service name in the TCP/IP services file is changed, the SVCENAME parameter is correspondingly changed. For example, from db2c to db2cDB2.

The following database configuration parameters are changed to Version 5 defaults:

– Database configuration release level

  This is changed to 0x0800.

– Database release level

  This is changed to 0x0800.

– Application control heap size (app_ctl_heap_sz)

  If the migrating database configuration file has this parameter at a value which is less than the Version 5 Default, the parameter is reset to its Version 5 default value, as follows:

Type of Database	Number (4K pages)
Database server with local and remote clients	128
Database server with local clients	64
Partitioned Database server with local and remote clients	256

– Lock list (LOCKLIST)

  For Version 1 DB2 databases, the LOCKLIST parameter will be first adjusted to LOCKLIST * 32/ 25. If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to its Version 5 default value, as follows:

Type of Database	Number (4 K pages)
Database server with local and remote clients	50
Database server with local clients	25

– Database heap (DBHEAP)

  For Version 1 DB2 databases, the database heap size will first be adjusted to DBHEAP * 16. If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to its Version 5 default value, as follows:

Type of Database	Number (4 K pages)
Database server with local and remote clients	600
Database server with local clients	300

– Default log space (LOGFILSIZ)

  The database migration process will attempt to increase the LOGFILSIZ value, if the log file related parameters have a total logfilsiz that is less than the default LOGFILSIZ value of 250 x 4 K pages.

– Application Heap size (APPLHEAPSZ)

For Version 1 DB2 databases, application heap size will be first adjusted to APPLEHEAPSZ * 16. If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to its Version 5 default value, as follows:

Type of Database	Number (4 K pages)
Database server with local and remote clients	128
Database server with local clients	64

- Sort Heap (SORTHEAP)

  For Version 1 DB2 databases, the value of the SORTHEAP parameter will be adjusted to SORTHEAP * 16.

- Statement Heap Size (STMTHEAP)

  If the migrating database configuration file has this parameter at a value which is less than the Version 5 default, the parameter is reset to the default value of 2048 x 4 K pages.

- Recovery Range and Soft Checkpoint Interval (SOFTMAX)

  If the migrating database configuration file has this parameter at a value that is less than the Version 5 default, the parameter is reset to the default value of 100.

- Package Cache Size (PCKCACHESZ)

  This is always reset to the Version 5 default, of -1, which means the value is 8 * MAXAPPLS unless 8 * MAXAPPLS is less than 32, in which case the default of -1 will set PCKCACHESZ to 32.

- Migrate Explain Tables.

  If you are not using explain tables in DB2 Version 2, skip this task.

  Version 5 of DB2 has added several new columns to the explain tables. These columns provide for the capture of:

  - Data for new SQL features added in Version 5.
  - More detailed access plan information.

  While the explain function in Version 5 will continue to work with explain tables created for Version 2, the new Version 5 data will not be captured in them.

  For better performance of SQL statements, we recommend that the Version 2 explain tables be dropped and new explain tables be created; see the SQL Reference and the Administration Guide for details on creating new explain tables. If, however, there are Version 2 explain tables that you need for ongoing comparison, you can use the EXPLMIG.DLL script to migrate them.

  To migrate the explain tables in a database that has been migrated to Version 5, connect to the database and run the following command from the sqllibmisc directory:

  ```
  db2 -tf sqllibmiscEXPLMIG.DLL
  ```

The explain tables belonging to the user ID that is used to connect to the database will be migrated. To migrate explain tables belonging to another user, connect to the database with that user ID and run the command.

7.4.1 Other Changes Introduced in DB2 UDB V5

During a database migration to DB2 UDB V5, the database path structure is modified and DB2's use of environment variables is modified. The DB2 Profiles Registry is introduced. The system catalog tables and views are changed. This section explains some of the changes that have occurred as a result of migrating your database system to DB2 UDB V5.

- **The DB2 Profiles Registry**

  Registry values, environment variables and configuration parameters control your database environment. In DB2 UDB V5, almost all of the environment variables have been moved to the DB2 Profiles Registry. The DB2 Profiles Registry is particularly advantageous for OS/2 and Windows NT DB2 systems since every time a variable is changed, the system does not need to be rebooted for the parameters to take effect. The DB2 Profiles Registry will also support applications that have no environmental settings. It centralizes control of environment variables on a global machine-wide level as well as supporting multiple environment profiles (one per instance).

  The DB2 Profiles Registry is defined in three distinct levels: the *system* or *node* registry, the *global* registry and the *instance* registry. This table displays the location of these flat files on the UNIX platform:

PROFILE REGISTRY	AIX	HP, SUN
System/Node Registry	/var/db2/v5/profiles.reg	/var/opt/db2/v5/profiles.reg
Global Registry	/var/db2/v5/default.env	/var/opt/db2/v5/default.opt
Instance Registry	$HOME/sqllib/profile.env	$HOME/sqllib/profile.env

- Profiles Registry after Migration

  During the migration process, only the files that hold the system and global registries are created. The file that holds the instance registry is not created, but a DB2 administrator can create this file to override and customize the environment for that instance. The values in the instance registry override the values in the global registry.

System Registry	Global Registry
List of DB2 UDB V5 Instances	(Empty)

The registries after migrating from DB2/PE V1 to DB2 UDB V5 are as follows:

- Customizing the Profiles Registry

  The db2set command is used to update the profile registry. It displays, sets, or removes DB2 profile variables.

To list all instance profiles at UDB V5, including the Database Administration Server, run the following command:

```
db2set -l
```

To list all the possible environment variables that can be configured and stored in the profiles registry at any of the profile registry levels (system, global or instance), run the following command:

```
db2set -lr
```

This produces the following list of variables:

Customizable Variables within the DB2 Profiles Registry		
DB2ACCOUNT	DB2LPORT	DB2SORCVBUF
DB2ADMINSERVER	DB2NBADAPTERS	DB2SORT
DB2ADMINSTART	DB2NBCHECKUPTIME	DB2SOSNDBUF
DB2BQTIME	DB2NBINTRLISTENS	DB2SYSTEM
DB2BQTRY	DB2NBRECVBUFFSIZE	DB2THREADIF
DB2CHKPTR	DB2NBRECVNCBS	DB2TIMEOUT
DB2CLIENTADPT	DB2NBRESOURCES	DB2UPMPR
DB2CLIENTCOMM	DB2NBSENDNCBS	DB2YIELD
DB2CLIINIPATH	DB2NBSESSIONS	DB2_AVOID_PREFETCH
DB2CODEPAGE	DB2NBXTRANCBS	DB2_FORCE_TRUNCATION
DB2COMM	DB2NODE	DB2_GRP_LOOKUP
DB2COUNTRY	DB2NTNOCACHE	DB2_INDEX_FREE
DB2DBDFT	DB2NTPRICLASS	DB2_RR_TO_RS
DB2DBMSADDR	DB2NTWORKSET	DB2_SHARE_MMU
DB2DEFPREP	DB2OPTIONS	DB2NPIPE
DB2DIRPATHNAME	DB2PATH	DB2_MMAP_READ
DB2DMNBCKCTLR	DB2PGCHK	DB2_MMAP_WRITE
DB2INCLUDE	DB2PRIORITIES	DB2AUTOSTART
DB2INSTDEF	DB2RAWLOGS	DB2DISCOVERYTIME
DB2INSTPROF	DB2REMOTEPREG	DB2NBDISCOVERRCVBUFS
DB2IQTIME	DB2ROUTE	DB2ENVLIST
DB2LOADREC	DB2RQTIME	DB2LIBPATH
DB2LOCK_TO_RB	DB2SERVICETPINSTANCE	DB2_LOADSORT_STACKSZ

- Database and Node Directories and the db2profile file

  Migration should successfully copy over the database and node directory entries for any remote databases. You should be able to connect to your remote databases without recataloging them. You may experience problems connecting to your database if you have not exported the environment variable DB2COMM to the communications protocol that you are using. During migration, a new DB2 UDB V5 instance is created, along with a new $HOME/sqllib/db2profile file. During the Instance migration, the DB2COMM environment variable is unset. You can find a copy of your previously customized db2profile file in your instance owner's $HOME/sqllib_v1 directory. If you had set any environment variables within the db2profile file (such as DB2COMM), then it is recommended that you set these variables in the DB2 Profiles Registry.

- Environment Variables

  During migration from DB2/PE V1 to DB2 UDB V5, some of the environment variables used in the previous version of DB2 are not set in DB2 UDB V5. The following lists the variables that are affected.

Environment Variables no longer present	Environment Variables set in DB2 UDB V5
DB2BQTIME, DB2BQTRY, DB2CHKPTR, DB2COMM, DB2DBDFT, DB2GROUPS, DB2IQTIME, DB2RQTIME,	DB2DIR=/usr/lpp/db2_05_00 INSTHOME=/home/inst1

- Authentication

  All local databases now have the same authentication type as the instance where they reside; the authentication type in the database directory is ignored by DB2 UDB V5 servers. Authentication exists at the instance for DB2 V2 and DB2 UDB V5; for DB2/PE V1, it exists at the database level.

- View Definitions

  If a view which was defined in DB2 V1 involves ″SELECT *″, the view may be unusable after migration. If the view is unusable, attempts to use it, directly or indirectly, will result in SQLCODE -158. The view must be dropped and recreated in order to avoid this error. If fewer than the current number of columns in the SELECT * table is desired, the recreated view must specify the needed columns.

Appendix A. db2split.cfg

```
;*********************************************************************
; Source File Name = db2split.cfg
;
; (C) COPYRIGHT International Business Machines Corp. 1996
; All Rights Reserved
; Licensed Materials - Property of IBM
;
; US Government Users Restricted Rights - Use, duplication or
; disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
;
; Function = db2split configuration file
;
;*********************************************************************
;
; ----------------------------
; Conventions used in this file
; ----------------------------
; -  the sign ';' starts a comment line
; -  empty lines will be ignored
; -  maximum length of a line is 255 bytes
; -  all valid entries are in the format like:
;       <keyword>=<value>
; -  all keywords are pre-defined and case insensitive
; -  the order of keyworks is not important
; -  keywords not specified in this file will either be undefined
;    or take a default value, depending on the keywords
; ------------------------------------------------------------------
;
; ------------------------------------------------------------
; Release level of this program.
; -  It can be "V5.0" or "v5.0" for Universal Database DB2, else
;    it is assumed to be DB2V1PE
; -  it has impact on the name of split files. Check the OutFile
;    parameter for detail.
;
Release=V5.0
; ------------------------------------------------------------
;
; ------------------------------------------------
; Name of input data file which will be split,
; make sure it exists and is readable.
; If not specified, use default: stdin
;
InFile=/u/username/testdata
; ------------------------------------------------
;
; ------------------------------------------------------------
; the type of input data file
; -  it can take the following values:
;    o  ASC (positional ASC file)
;    o  DEL (delimited ASC file)
;    o  BIN (binary numeric data file, all numeric columns are in
;       binary format instead of character string format)
;    o  PACK (packeddecimal data file, all decimal columns are in
;       packeddecimal format instead of character string format)
```

```
;    o  or combinations of ASC/BIN/PACK. For Example,
;
;          FileType=BIN
;          FileType=PACK
;
;       then all numeric columns in the data file are in binary
;       format, and all decimal columns are in packeddecimal format
;
; -  case insensitive
; -  no default value
; -  if not specified, file type is determined by cDelimiter
;    parameter (see follows)
;
FileType=ASC
; ------------------------------------------------------------
;
; ------------------------------------------------------------------
; Turn on or off NEWLINE checking
; -  it can be YES, or NO (case insensitive)
; -  only meaningful when the input data file is a fixed length (with
;    RecLen parameter specified) ASC file
; -  else, this flag is ignored
; -  by default, NO
; -  if YES, always assume records are delimited by line-feed
;    character, and the utility will check if each record is ended
;    with a line-feed character or not, also the actual record length
;    matches the specified RecLen or not
; -  else, turn off the checking
; -  it is useful, when combined with RecLen parameter, to detect
;    abnormal records (too long or too short) in a positional ASC
;    data file
;
NEWLINE=NO
; ------------------------------------------------------------------
;
; ------------------------------------------------------------
; Record length for input data file
; -  if specified, and a positional ASC data file, it is the exact
;    length of record (including line-feed if any) in the data file
;    minus one (1). The maximum record length is 32K in this case.
; -  if not specified, and a positional ASC data file, all records
;    in the data file are assumed to be delimited by a line-feed
;    character. The implied maximum record length is 32K as well.
; -  if a DEL (delimited) data file, this parameter will be ignored,
;    and there is no limitation for the maximum record length.
; -  if a BINARY data file, this parameter has to be specified,
;    and it has to be the exact length of record in the data file
;    minus (1)
; -  "minus (1)" here is just for backward compatibility
;
;RecLen=125
; ------------------------------------------------------------
;
; ------------------------------------------------------------
; Nodes on which the table is to be created
; -  it is used by the db2split program to generate a partitioning
;    map if there is no partitioning map supplied
; -  in the format like,
;       (0,1,m-n,...999)   WHERE m<n
```

```
; -  specify only "Nodes" or "Mapfili" but not both
;
;Nodes=(0,1)
; ------------------------------------------------------------
;
; ------------------------------------------------------------
; Nodes for which output files are to be created.
; -  "OutPutNodes" must be a subset of "Nodes"
; -  in the format like,
;        (0,1,m-n,...999)   WHERE m<n
; -  if not specified, output files will be produced on all "Nodes"
; -  If "OutPutNodes" has only one member, the option "OutputType"
;    can be used to specify whether the output should be written
;    to a file or to stdout
; -  If "OutPutNodes" has more than one member, the option
;    "OutputType" will be ignored
;
OutputNodes=(0,1)
; ------------------------------------------------------------
;
; ------------------------------------------------------------
; Write output data to a file or to stdout
; -  only effective when there is only one node in "OutPutNodes"
; -  take value from two choices:
;       w (file)        or      s (stdout)
; -  by default, OutPutType will be "s"
;
;OutputType=w
; ------------------------------------------------------------
;
; ------------------------------------------------------------
; Name of the partitioning map file.
; -  it is also used to get the list of nodes to partition the data
;    if "Nodes" is not specified
; -  specify only "Nodes" or "Mapfili" but not both
;
Mapfili=db2split.map
; ------------------------------------------------------------
;
; ----------------------------------------
; Name of the output partitioning map file
; -  mandatory for ANALYZE runtype
; -  optional for PARTITION runtype
;
Mapfilo=/u/username/MyOutMap
; ----------------------------------------
;
; ----------------------------------------
; Name of the distribution file
; -  if not specified, use default: DISTFILE
;
DistFile=/u/username/MyDistFile
; ----------------------------------------
;
; ------------------------------------------------------------
; Name of the file used to log db2split process,
; -  in the format like,
;       <logfilename>[,w|a]
; -  option "w" causes db2split over-write <logfilename>, while
```

```
;    option "a" causes db2split append <logfilename>
; -  default option: "w" (case insensitive)
; -  If not specified, logging will be wrote to standard error
;
LogFile=/u/username/MyLog,a
; ------------------------------------------------------------
;
; ------------------------------------------------------------
; Prefix for output data file
; -  if Release (parameter) is anything other than "V5.0" or
;    "v5.0", db2split appends a five-digit suffix (00000..00999)
;    to the end of the prefix to form output file names
; -  else db2split appends a three-digit suffix (000..999) to
;    the end of the prefix to form output file names
;
OutFile=/u/username/MyTestDataOut
; ------------------------------------------------------------
;
; ----------------------------------------------------------
; Column delimiter
; -  the column delimiter can be any character except
;    binary zero, line feed character, space, and carriage
;    return character
; -  Plus, if the codepage of data file is a DBCS/Mixed/EUC
;    codepage, it has to be less than '\x40'
; -  More, if the codepage of data file is an EBCDIC Mixed
;    codepage, it can not be the shift-in (SI), and shift-out
;    (SO) characters.
; -  the column delimiter can be specified in character
;    format, or in hexdecimal format. For example,
;    o  CDelimiter=,
;    o  CDelimiter=0X2c
;    o  CDelimiter=x2C
;    they all represent the same meaning, an ASC comma.
; -  if FILTYPE parameter was not specified in this file,
;    it determines the type of input data file
;    o  if it is not defined (commented out), the input file
;       will be a positional ASC data file
;    o  otherwise, DEL input data file
; -  if FILETYPE is specified and it is a DEL file, then the
;    default column delimiter is
;    o  0x2c (ASC comma) for data file in ASC codepages
;       or EBCDIC MBCS codepages
;    o  0x6b (EBCDIC comma) for data file in EBCDIC SBCS
;       codepages
; -  if FILETYPE is specified and it is not a DEL file, then
;    column delimiter is ignored
; -  CDelimiter, SDelimiter, and DecPt have to be mutually
;    exclusive
;
CDelimiter=,
; ----------------------------------------------------------
;
; ----------------------------------------------------------
; String delimiter
; -  The string delimiter can be any character except
;    binary zero, line feed character, space, carriage
;    return character, and period due to conflicts with
;    periods in time and time stamp values
```

```
; -  Plus, if the codepage of data file is a DBCS/Mixed/EUC
;    codepage, it has to be less than '\x40'
; -  More, if the codepage of data file is an EBCDIC Mixed
;    codepage, it can not be the shift-in (SI), and shift-out
;    (SO) characters.
; ------------------------------------------------------
; Decimal Point
; -  The decimal point can be any character except
;    binary zero, line feed character, space, and carriage
;    return character
; -  Plus, if the codepage of data file is a DBCS/Mixed/EUC
;    codepage, it has to be less than '\x40'
; -  More, if the codepage of data file is an EBCDIC Mixed
;    codepage, it can not be the shift-in (SI), and shift-out
;    (SO) characters.
; -  the decimal point can be specified in character
;    format, or in hexdecimal format. For example,
;    o  DecPt=.
;    o  DecPt=0X2e
;    o  DecPt=x2E
;    they all represent the same meaning, an ASC period
; -  if FILETYPE is specified and it is a DEL file, then the
;    default decimal point is
;    o  0x2e (ASC period) for data file in ASC
;       codepages or EBCDIC MBCS codepages
;    o  0x7f (EBCDIC period) for data file in EBCDIC
;       SBCS codepages
; -  if FILETYPE is specified and it is not a DEL file, then
;    decimal point delimiter is ignored
; -  CDelimiter, SDelimiter, and DecPt have to be mutually
;    exclusive
;
;DecPt=.
; ------------------------------------------------------
;
; ------------------------------------------------------
; Running type
; -  can be either ANALYZE or PARTITION
; -  ANALYZE: generate a cutomized optimal partitioning map
; -  PARTITION: split the data
;
RunType=PARTITION
; ------------------------------------------------------
;
; ---------------------------------------------------------------------
; Checking level
; -  can be either CHECK or NOCHECK
; -  CHECK: program will check truncation of each record at Output
;          it also check the record is empty or not at input if the
;          record is less than 32k bytes long
; -  NOCHECK: Program will not check for truncation of record at I/O
; -  by default, CHECK
;
;Check_Level=NOCHECK
; ---------------------------------------------------------------------
;
; ------------------------------------------------------------
; Partitioning key
; -  the order of partitioning keys is IMPORTANT
```

```
; -  at least one partitioning key has to be specified, and it must
;    be identical to the partitioning key definition for the table
; -  it includes 6 fields:
;    1. column name (for logging purpose)
;    2. column number of the partitioning key in each record
;       o  starts from column 1
;       o  only valid for delimited data
;    3. offset of the partitioning key with respect to
;       the beginning of the record
;       o  begins at position 1
;       o  only valid for non-delimited data
;    4. Length of partitioning key
;       o  mandatory for non-delimited data
;       o  for delimited data, if the partitioning key is CHARACTER
;          (or CHAR), VARCHAR, FOR_BIT_CHAR, or FOR_BIT_VARCHAR data type,
;          it must be specified as well. And it must be consistent with
;          the corresponding column length in the database table
;       o  excluding string delimiters if any
;    5. Null indicator
;       o  N    Null data is allowed
;       o  NN   Not Null.
;       o  NNWD Not Null with default, no difference from NN
;    6. Type of data conversion for hashing into partition index:
;       o  SMALLINT
--
;       o  INTEGER
;       o  float                (only valid for binary data file)
;       o  double               (only valid for binary data file)
;       o  CHARACTER or CHAR
;       o  VARCHAR
;       o  FOR_BIT_CHAR
;       o  FOR_BIT_VARCHAR
;       o  DECIMAL(x,y)   (x: precision, y: scale)
;       o  DATE                 in YYYY-MM-DD format
;       o  TIME                 in HH.MM.SS format
;       o  TIMESTAMP in YYYY-MM-DD-HH.MM.SS.XXXXXX format
;
Partition=age,2,,,N,Integer
; ------------------------------------------------------------------
;
; ----------------------------------
; Trace hashing values
; -  it is used to dump hashing values
; -  the number of records to be traced
;
Trace=100
; ----------------------------------
;
; ------------------------------------------------------------
; Generating header switch
; -  can be YES/NO (case insensitive)
; -  by default, it is YES and a header will be generated at the
;    beginning of output data file
; -  if NO, the header will not be generated
;
;header=yes
; ------------------------------------------------------------
;
; ------------------------------------------------------------
```

```
; codepage of database
; -  the codepage number of database, where the table is defined
; -  it must be a DBM supported codepage
; -  if not given, there are two possible situations:
;    o  codepage for the input data file is specified (below),
;
Trace=100
; ------------------------------------
;
; ------------------------------------------------------------
; Generating header switch
; -  can be YES/NO (case insensitive)
; -  by default, it is YES and a header will be generated at the
;    beginning of output data file
; -  if NO, the header will not be generated
;
;header=yes
; ------------------------------------------------------------
;
; ------------------------------------------------------------
; codepage of database
; -  the codepage number of database, where the table is defined
; -  it must be a DBM supported codepage
; -  if not given, there are two possible situations:
;    o  codepage for the input data file is specified (below),
;       it will be assumed the same as that codepage;
;    o  if not, it will be assumed the same codepage as the
;       application, and no codepage conversion will be done.+

;DB_CODEPAGE=819
; ------------------------------------------------------------
;
; ------------------------------------------------------------
; codepage of input data
; -  the codepage number of input data
; -  it must be a DBM convertable codepage
; -  if not given, there are two possible situations:
;    o  codepage for database is specified (above),
;       it will be assumed the same as that codepage;
;    o  if not, it will be assumed the same codepage as the
;       application, and no codepage conversion will be done.
;DATA_CODEPAGE=819
```

Appendix B. Special Notices

This publication is intended to help the system or database administrator to migrate database systems to DB2 Universal Database. The information in this publication is not intended as the specification of any programming interfaces that are provided by DB2 Universal Database. See the PUBLICATIONS section of the IBM Programming Announcement for DB2 Universal Database Version 5.0 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX®	Client Access
DB2®	DB2®/Universal Server
DRDA®	IBM®
MVS/ESA	NetView®
Net Data	OS/2®
OS/390	OS/400®
SP	SP2®

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

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

Appendix C. Related Publications

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

C.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 365.

- *DB2 Version 2 Planning Guide for Database Administrators,* SG24-2523

C.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

C.3 Other Publications

These publications are also relevant as further information sources:

- *IBM DB2 Universal Database* for Windows NT Quick Beginnings Version 5 S10J-8149
- *IBM DB2 Universal Database for OS/2 Quick Beginnings Version 5 S10J-8147*
- *IBM DB2 Universal Database for UNIX Quick Beginnings Version 5 S10J-8148*
- *IBM DB2 Universal Database Administration Guide Version 5 S10J-8157*
- *IBM DB2 Universal Database Command Reference Version 5 S10J-8166*
- *IBM DB2 Universal Database SQL Reference Version 5 S10J-8165*

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at http://www.redbooks.ibm.com.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

  For a list of product area specialists in the ITSO: type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Web Site on the World Wide Web**

  http://w3.itso.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.htm). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com
- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.htm). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

List of Abbreviations

ADSM	Adstar Distributed Storage Manager
BLOB	Binary Large Object
CLOB	Character Large Object
CAE	Client Application Enabler
CLP	Command Line Processor
DBLOB	Double Byte Binary Large Object
DB2/CS	Database 2 Common Server
DB2/PE	Database 2 Parallel Edition
DDCS	Distributed Database Connection Services
DEL	Delimited ASCII
DPropR	Data Propagator Relational
DRDA	Distributed Relational Database Architecture
DMS	Database Managed Space
EE	Enterprise Edition
EEE	Enterprise - Extended Edition
ITSO	International Technical Support Organization
IXF	Integrated Exchange Format
SDK	Software Developer's Kit
SMS	System Managed Space
UDB	Universal Database

Index

E

F

G

I

J

L

M

N

V

W

ITSO Redbook Evaluation

Migrating to DB2 Universal Database Version 5
SG24-2006-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

_____

Printed in U.S.A.