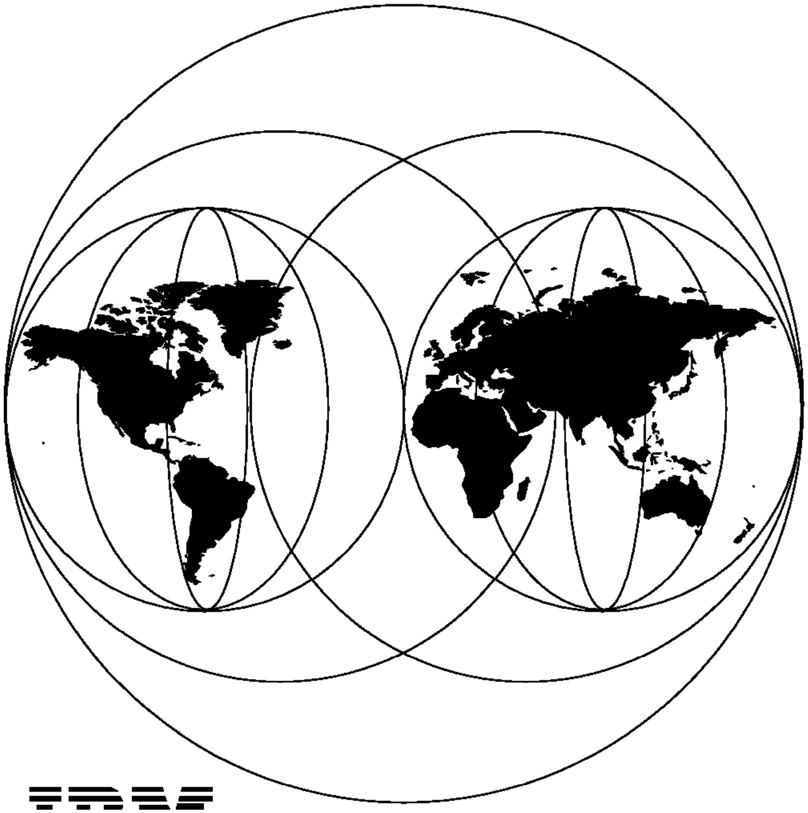
International Technical Support Organization SG24-2523-00

DB2 Version 2 Planning Guide for Database Administrators

January 1996



International Technical Support Organization
Austin Center

International Technical Support Organization

SG24-2523-00

DB2 Version 2 Planning Guide for Database Administrators

January 1996

Take Note!

Before using this information and the product it supports, be sure to read the general information under "Special Notices" on page xiii.

First Edition (January 1996)

This edition applies to DB2 for AIX Version 2.1 of Program Number 41H2128 for use with the AIX Operating System and DB2 for OS/2 Version 2.1 of Program 41H2114 for use with the OS/2 Operating System.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader′s feedback appears facing Chapter 1. If the form has been removed, 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.

Abstract

This document is unique in its detailed coverage of DB2 for the AIX and OS/2 platforms (DB2 for AIX and DB2 for OS/2). It focuses on those aspects that a database administrator must consider for planning a database in DB2 Version 2. The document has more AIX examples than OS/2, though DB2 for OS/2 database administrators will find it useful. It also provides information about migrating from a Version 1 database either OS/2 or AIX.

This document was written for database administrators and anyone wanting detailed information on DB2 Version 2. Some knowledge of DB2, Version 1, either for AIX or OS/2 is assumed.

(226 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help database administrators or anyone planning a DB2 Version 2 environment. It also provides detailed information on some of the new features in Version 2 that many people will find useful. The information in this publication is not intended as the specification of any programming interfaces that are provided by DB2 for AIX Version 2.1 or DB2 for OS/2 Version 2.1. See the PUBLICATIONS section of the IBM Programming Announcement for DB2 for AIX Version 2.1 or DB2 for OS/2 Version 2.1 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.

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.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM

The following terms are trademarks of other companies:

Windows is a trademark of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other trademarks are trademarks of their respective companies.

Preface

This document is intended to help database administrators plan their DB2 Version 2 environment. Specifically, it covers the AIX and OS/2 platforms. DB2 is supported on many platforms, such as HP, Sun or NT. These are not discussed. There may be platform dependendcies that need to be considered.

This document is intended for database administrators and anyone wanting detailed information about DB2 Version 2. This document focuses on administrative aspects such as placement, movement and backup/restore of data. It also looks at some of the user interfaces such as the Database Director, which is new in Version 2. It does not cover every aspect that a database administrator might be involved in, such as how to use the Visual Explain facility. Note that not every new feature of Version 2 is covered in this document.

How This Document is Organized

The document is organized as follows:

- Chapter 1, “Product Overview”

  This chapter describes the product packaging in Version 2. It also gives a connectivity summary.

- Chapter 2, “Customer Scenario”

  This chapter looks at a sample customer environment and explores some of the considerations that a customer might have in migrating their environment from DB2/6000 Version 1 to DB2 for AIX Version 2.

- Chapter 3, “Instances and Users”

  This chapter describes the instance, users and security.

- Chapter 4, “Data Placement”

  This chapter describes the placement of data. A detailed look at tablespaces and planning the storage is covered.

- Chapter 5, “Data Movement”

  This chapter describes the movement of data. In particular, it focuses on the load utility.

- Chapter 6, “Logging”

  This chapter discusses logging in DB2.

- Chapter 7, “Backup and Restore”

  This chapter describes planning a backup/restore strategy for your environment.

- Chapter 8, “Data Access”

  This chapter looks at the user interfaces a database administrator may choose from. Covered are the Command Line Processor, SQL Query Products and the Database Directory

- Appendix A, “Database Migration”

  This appendix describes the migration from Version 1 for both AIX and OS/2.

Related Publications

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

- *IBM DATABASE 2 Command Reference - for common servers Version 2* , S20H-4645-00
- *IBM DATABASE 2 SQL Reference - for common servers Version 2* , S20H-4665-00
- *IBM DATABASE 2 Administration Guide - for common servers Version 2*, S20H-4580-00
- *IBM DATABASE 2 Information and Concepts Guide - for common servers Version2*, S20H-4664-00
- *IBM DATABASE 2 for AIX Planning Guide Version 2*, S20H-4758-00
- *IBM DATABASE 2 DDCS User′s Guide*, S20H-4793-00
- *IBM DATABASE 2 Software Developer′s Kit for AIX*, S20H-4780-00
- *IBM DATABASE 2 Installing and Using AIX Clients*, S20H-4666-00
- *IBM DATABASE 2 Installing and Using DB2 Clients for Windows*, S20H-4789-00

International Technical Support Organization Publications

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

> *International Technical Support Organization Bibliography of Redbooks,* GG24-3070.

To get a catalog of ITSO technical publications (known as “redbooks”), VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order ITSO Redbooks

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-445-9629. Almost all major credit cards are accepted. Outside the USA, customers should contact their local IBM office.

Customers may order hardcopy ITSO books individually or in customized sets, called GBOFs, which relate to specific functions of interest. IBM employees and customers may also order ITSO books in online format on CD-ROM collections, which contain redbooks on a variety of products.

Acknowledgments

This project was designed and managed by:

Calene Janacek
International Technical Support Organization, Austin Center.

This publication is the result of a residency conducted at the International Technical Support Organization, Austin Center. The authors of this document are:

William McAuliffe
IBM Ireland

Veronique Gillet
IBM France

Grant Hutchison
IBM Canada

Calene Janacek
IBM Austin

Giuseppe Ornaghi
IBM Italy

Dale McInnis
IBM Canada

Gerhard Mueller
IBM Germany

Miguel Robles
IBM Spain

Melanie Stopfer
IBM US Education and Training

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

The document had many reviewers whose time and assistance was greatly appreciated:

Marcus Brewer, Editor
ITSO Austin

Dorothy Chen
IBM Canada

Al Comeau
IBM Canada

Grant Hutchison
IBM Canada

Dale McInnis
IBM Canada

Ella Polan
IBM Canada

Steve Schmidt
IBM Canada

My thanks for time and support is also extended to

Tim Sennitt
ITSO Boca Raton

Chapter 1. Product Overview

DB2 Version 2 introduces a common server code on both the Intel and RISC platforms. Version 2 provides equivalent function to both platforms, while using operating-system services and optimization to fully leverage each platform environment. Application function compatibility with Version 1 is maintained in Version 2.

The DB2 for OS/2 and DB2 for AIX products are now collectively known as DB2 common server because they operate on similar platforms and share the same code base. The packaging of DB2 products for both OS/2 and AIX is consistent in Version 2. DB2 common server also shares the same basic architecture as DB2 for MVS and uses many of the same key algorithms. However, the internal components of DB2 have been optimized to exploit each operating system and platform.

The DB2 common server Version 2 products that are discussed in this publication are the following:

- IBM DATABASE 2 for AIX Version 2.1 Single-User (DB2 for AIX)
- IBM DATABASE 2 for OS/2 Version 2.1 Single-User (DB2 for OS/2)
- IBM DATABASE 2 for AIX Version 2.1 Server (DB2 for AIX)
- IBM DATABASE 2 for OS/2 Version 2.1 Server (DB2 for OS/2)
- IBM Distributed Database Connection Services for OS/2 Version 2.3 Single-User (DDCS for OS/2)
- IBM Distributed Database Connection Services for OS/2 Version 2.3 Multi-User Gateway (DDCS for OS/2)
- IBM Distributed Database Connection Services for AIX Version 2.3 Multi-User Gateway (DDCS for AIX)
- IBM DATABASE 2 Software Developer′s Kit for AIX Version 2.1 (DB2 SDK for AIX)
- IBM DATABASE 2 Software Developer′s Kit for OS/2 Version 2.1 (DB2 SDK for OS/2)
- IBM DATABASE 2 Software Developer′s Kit for Windows Version 2.1 (DB2 SDK for Windows)

Throughout the remainder of this book, unless otherwise noted, DB2 will refer to the DB2 common server.

This chapter is an introduction to DB2 Version 2. Included are descriptions of the following;

- DB2 Version 2 Products
- DB2 Version 2 Components
- Packaging of Products and Components

1.1 DB2 Version 2 Engine

The DB2 database engine is available in two versions: Single-User and Server. The database engine in both of these versions is identical. The engine is a full-function relational database management system that includes the optimizer, SQL support and tools to help manage the data. The difference between these products is whether remote clients can access the databases or not and whether or not the application development environment is an integrated feature.

1.1.1 Base Engine Functions

The following are the base engine functions of DB2:

- Full-function relational database management system.
- Cost-based optimizer which supports complex queries.
- Data integrity through declarative referential integrity, forward recovery and multilevel concurrency control.
- Flexible management of very large databases.
- Ability for an application to read or update tables in more than one database from within a single unit of work with full data integrity. This capability is provided through Distributed Unit of Work (DUOW) functionality, also known as two-phase commit.

For a full discussion on the DB2 base engine functions, see the *Information and Concepts Guide*.

The major functional engine enhancements in DB2 Version 2 over Version 1 were in the following areas:

- Extended SQL Capabilities
- Enhanced SQL Optimizer
- Database Performance
- Systems Management Support
- Compatibility with DB2 for MVS
- Integrity and Data Protection
- Object and Relational Capabilities
- National Language Support

These enhancements are applicable to both the Single-User and Server versions of DB2. For a full description of these enhancements, see the *DB2 for OS/2 (or AIX) Planning Guide*.

1.1.2 DB2 Single-User

The Single-User version of DB2 contains all the features of the base engine, as well as the following:

- Application Development Environment

  The application development environment of the DB2 SDK is included with DB2 Single-User. This allows you to create applications on the same platform where you have local databases.

DB2 Single-User is an ideal environment for those who develop applications or perform database administration tasks and need to have local databases to test their applications.

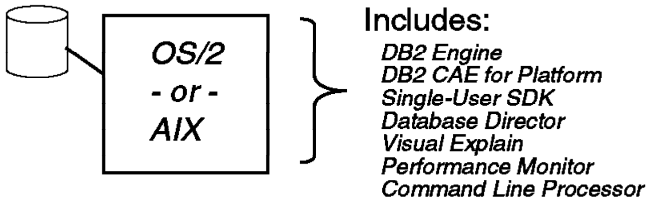


Figure 1. DB2 Single-User

Figure 1 shows the packaging for the DB2 Single-User product.

DB2 Single-User contains the DB2 database engine, application development tools, administration tools, an application-development Command Line Processor, Visual Explain, and the Client Application Enabler (CAE) for the platform on which it is installed. The DB2 for OS/2 Single-User package also contains the CAEs for DOS and Windows.

1.1.3 DB2 Server

DB2 Server supports all the base engine functions plus provides support for remote as well as local clients who want database access. Remote database clients can communicate with the server using any of the supported communication protocols (see *DB2 for AIX (or OS/2) Planning Guide*). Remote clients are available for a varied set of operating environments, such as DOS, Windows, OS/2, Macintosh, and UNIX. DB2 Server provides a true client/server database, supporting autonomous clients that do not need to know the physical location of the database. The processing of applications is split between the database server and client machines.

To provide the application development environment with DB2 Server, DB2 SDK must be installed on the same machine.

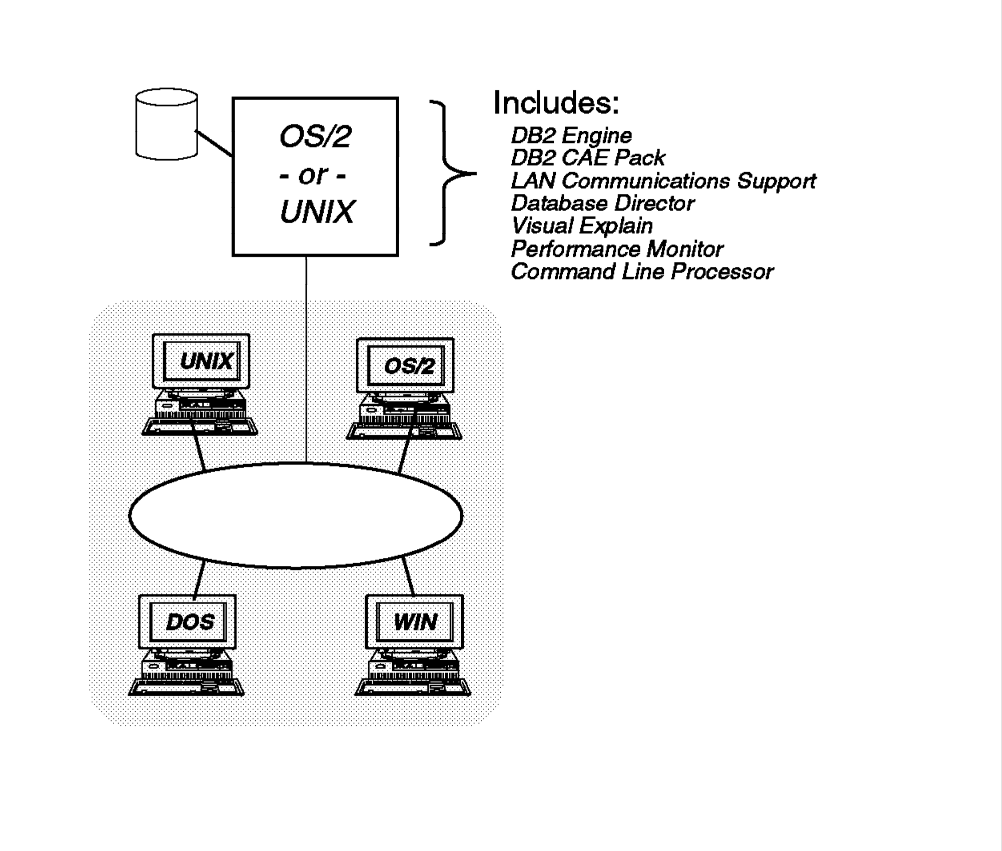


Figure 2. DB2 Server Engine

DB2 Server contains the DB2 database engine, communications support for clients, administration tools, the Command Line Processor, and the Client Pack (run-time code for a variety of client workstations).

DB2 Server also now has DRDA Application Server capability. This function allows DB2 for MVS, DB2 for VM and DB2 or OS/400 applications (or any other application that implements DRDA application requestor functionality) to access data located in DB2 databases. Thus, existing database applications running on the MVS, VM and OS/400 platforms may be able to access data stored in DB2 databases. The Distributed Unit of Work (DUOW) is subject to the availability of an external transaction manager.

1.2 DB2 Version 2 Components

DB2 Version 2 components are packaged and bundled with different DB2 products. Components are discrete elements of function within DB2 products which are labelled and described separately. They have no product cost, and customers may choose whether or not they wish to install them. The components may be installed on multiple workstations without additional product cost.

The components of DB2 Version 2 are as follows:

Command Line Processor (CLP)

Communications Support

Client Application Enabler (CAE)

- Database Director
- Visual Explain
- Performance Monitor

1.2.1 Command Line Processor (CLP)

The Command Line Processor (CLP) is a character-based application that provides the ability to enter SQL statements, environment, utility, and configuration commands. Using the CLP, you can create a database, connect to a database and select rows from a table, for example.

1.2.2 Communications Support

In Version 1 of DB2 for AIX, there was a separate product called Client Support/6000 which, when installed with DB2 acting as a database server, allowed databases to be accessed and shared by remote AIX, DOS, Windows, and OS/2 clients. Communications Support now replaces Client Support as an installable client communications support feature of DB2 Server and DB2 Multi-User Gateway.

1.2.3 Client Application Enablers (CAE)

The Client Application Enablers allow application programs running on DOS, Windows, OS/2, AIX, HP-UX, and Solaris client workstations to access data stored in any of the DB2 relational databases. The CAEs provide run-time access to applications that support Microsoft′s Open Database Connectivity (ODBC) interface and to those developed with the DB2 Software Developer′s Kit (DB2 SDK).

A set of all the supported DB2 Client Application Enabler products is included with the DB2 Server and DDCS Multi-User Gateway products. This is known as a *Client Pack*. Any number of these clients can be separately installed on the client workstations on your network. Client support for additional workstation operating systems, as they are developed, will be available at no charge to customers of DB2 Version 2 products.

The following is a list of the Version 2 enhancements found in the Client Application Enabler:

- DB2 Client Set-Up
- Remote Administration Capabilities
- Directory Caching
- DRDA Stored Procedures

For a full discussion on these enhancements, consult the *DB2 for OS/2 (or AIX) Planning Guide*.

There are several graphical database administration components designed to help database administrators manage and administer DB2 databases.

There are several DB2 components which, when installed on the same workstation as the database server, allow you to administer your databases locally. Alternatively, you can set up a dedicated database administrator′s system by installing these tools on a client workstation. This allows you to administer databases remotely.

Following is a list of the tools and their capabilities:

- Database Director

  The Database Director is an easy-to-use graphical interface that displays database objects (databases, tables and packages) and their relationship to each other. Using the Director, you can select one or more database objects to perform the following tasks:

  – Configure databases and database manager instances

  – Manage the directories necessary for accessing local and remote databases

  – Back up and recover databases or tablespaces

- Visual Explain

  This tool graphically shows you the access plans for an SQL statement. It provides a visual representation of how transactions relate to each other, and you can assess if a database change needs to be made. You can also model the effect of any changes on the production environment before committing the changes. This allows you to assess the impact of environment changes on SQL.

- Performance Monitor

  This tool can help you monitor the performance of your DB2 system for tuning purposes. It uses the database system monitor APIs provided with the base engine. With the Performance Monitor, you can:

  – Define your own statistics, in addition to the defaults provided

  – Determine and analyze performance problems in the database manager or database applications

  – Tune SQL statements for better performance

  – Identify exception conditions based on thresholds you define

1.3 Other DB2 Version 2 Products

To offer a comprehensive database management solution, other products are needed to support the database function. First, a communications gateway facility is needed to allow access to other relational database management systems that implement the DRDA application server specification. Second, a collection of development tools is required to meet the needs of database application developers. The DB2 companion products which meet these needs are:

Distributed Database Connection Services (DDCS)

Software Developer′s Kit (SDK)

1.4 Distributed Database Connection Services (DDCS)

The Distributed Database Connection Services (DDCS) product addresses the need to access and update corporate data stored in a host relational database from applications running in a LAN-based environment. It can be implemented as a Single-User or Multi-User Gateway environment. This capability is generically referred to as the DRDA Application Requestor function.

The following enhancements were made to the DDCS products in Version 2:

- DRDA Bind Options
- Stored Procedure Support
- Compound SQL
- Pre-fetching of Data Pages
- SQLCODE Mapping
- Accounting String Support

See the *DB2 for OS/2 (or AIX) Planning Guide* for a summary of these enhancements, or see the *DDCS User's Guide* for additional information.

1.4.1 DDCS Single-User

The Single-User version of DDCS is available for the OS/2 platform only. It provides local client direct access to databases maintained by DRDA Application Servers. It is not necessary to have the database manager product installed on this workstation.

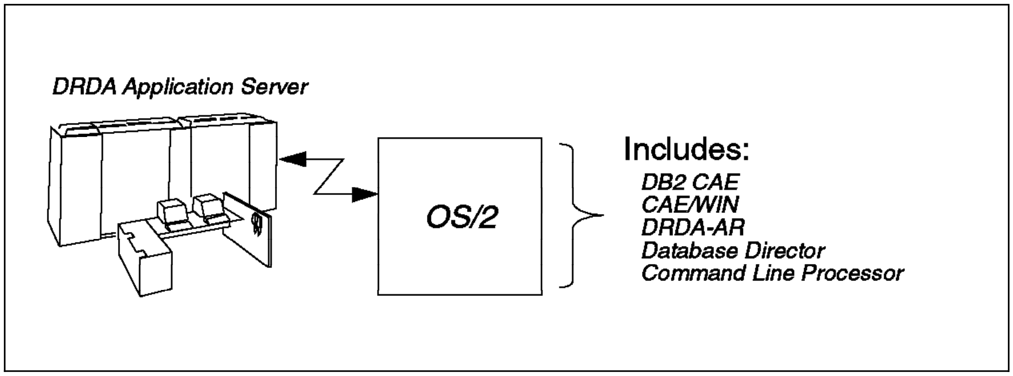


Figure 3. DDCS Single-User for OS/2

This package contains the DRDA Application Requestor, administration tools, a Command Line Processor, and the CAEs for OS/2, DOS and Windows.

1.4.2 DDCS Multi-User Gateway

The Multi-User Gateway version of DDCS provides access from the clients on your network to DRDA Application Servers. A DDCS gateway server can receive concurrent requests from multiple remote database clients. Each of these requests is rerouted to the appropriate host database for processing.

While it is possible to access the DB2 databases with DDCS, it is recommended that the native distributed support provided with the DB2 server functions be used rather than DDCS.

Version 2 has new installation considerations for DDCS Multi-User Gateway. When installing on the AIX platform, you must install the run-time portion of DB2 Server. It is not necessary to purchase DB2 Server unless you want database server functions. When installing DDCS Multi-User Gateway on the OS/2 platform, DB2 Server will install automatically as a pre-requisite.

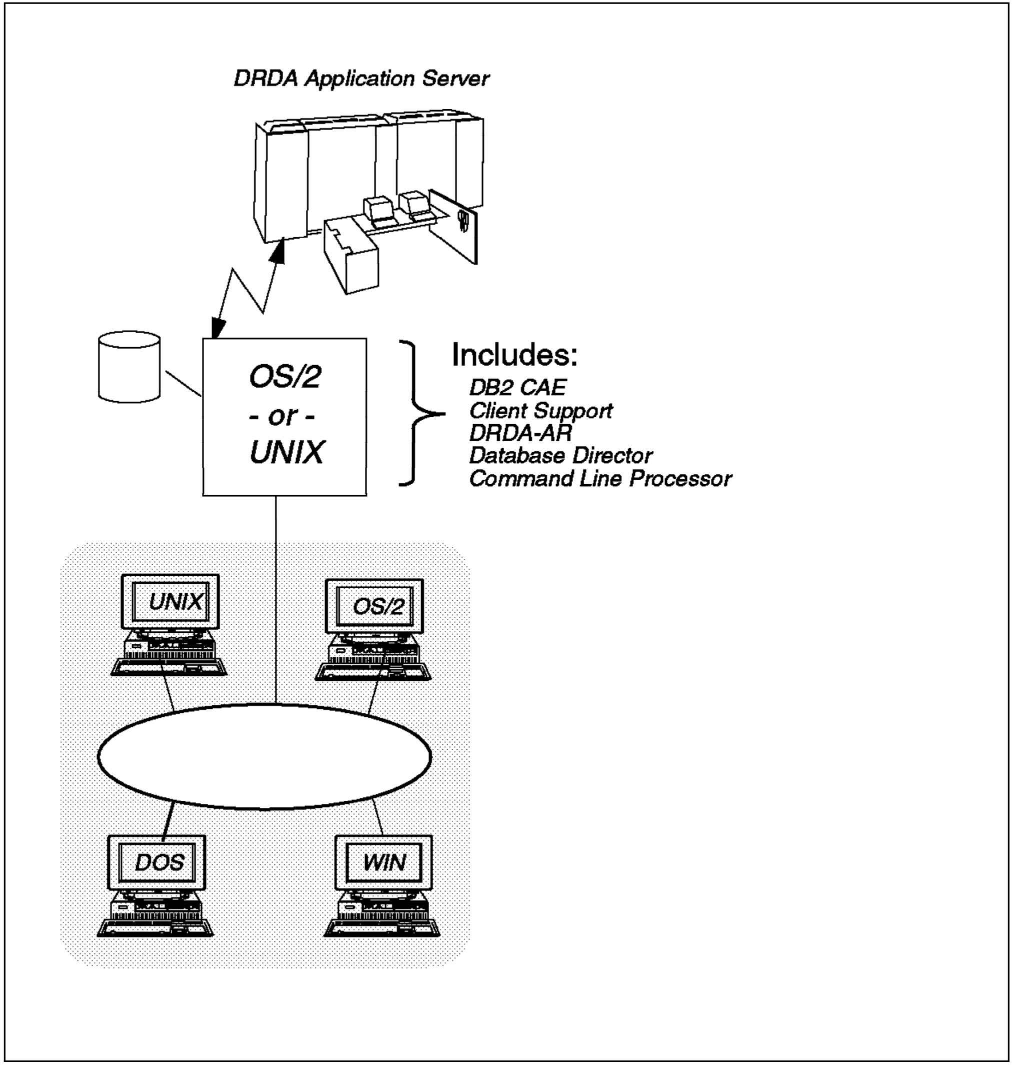


Figure 4. DDCS Multi-User Gateway

Figure 4 shows the DDCS Multi-User Gateway packaging. The gateway package contains the DRDA Application Requestor, Client Support, administration tools, Command Line Processor, and the Client Pack.

1.4.3 Software Developer′s Kit (SDK)

The DB2 Software Developer′s Kit is a collection of tools designed to meet the needs of database application developers. DB2 SDK includes all necessary development tools, except a compiler. It provides support for the creation of character-based, multimedia or object-oriented applications.

DB2 SDK is a separate product that can be installed on the server where DB2 is installed or on a remote client. There is a platform-specific version of the supported client environments. Applications developed with DB2 SDK will run on any client platform where the equivalent Client Application Enabler component is installed and can access all DB2 servers as well as any other application server that implements the DRDA protocol.

The DB2 SDK provides support to develop applications using the following interfaces:

- Embedded SQL
- Call-Level Interface (CLI) development environment (compatible with Microsoft′s ODBC on Windows)
- Application programming interfaces (APIs) to access database utilities
- A prototyping environment using the Command Line Processor′s interactive SQL

Programming libraries, header files, code samples, the Database Director, Visual Explain, and a complete set of documentation are provided for developing applications with embedded SQL and the DB2 CLI. Programming languages including COBOL, FORTRAN, REXX, C, and C++ are supported for application development, and pre-compilers for the supported languages are provided.

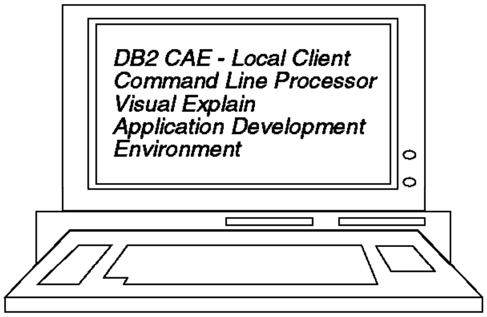


Figure 5. DB2 Software Developer′s Kit (DB2 SDK)

The developer′s kit package contains application development tools, Visual Explain, administration tools, an application development Command Line Processor, and the CAE for the platform on which it is installed.

The following enhancements have been made in Version 2 to the DB2 SDK products:

- DB2 Call-Level Interface extensions
- Programming language support
- Visual Explain
- Bind Options
- Flagger Utility

See the *DB2 for OS/2 (or AIX) Planning Guide* for a summary of these enhancements and the *DB2 SDK - Building Your Applications* publication for additional information.

1.5 DB2 Client/Server Environment

Figure 6 shows the DB2 client/server environment. It is comprised of clients on DOS, WINDOWS, OS/2, AIX, HP-UX, or Sun Solaris platforms. These clients can access data on DB2 for AIX, DB2 for OS/2, DB2 for HP-UX, or DB2 for Solaris database servers.

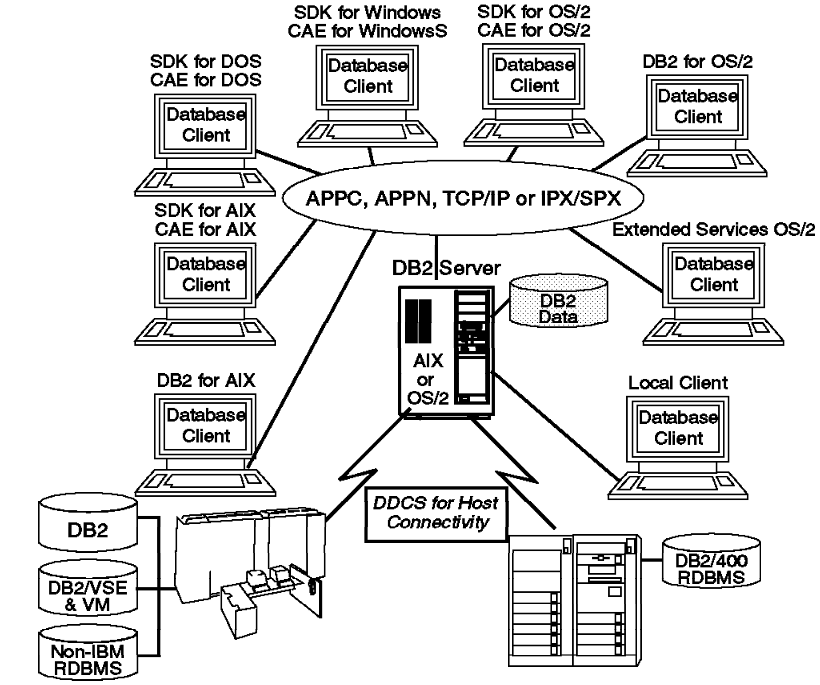


Figure 6. DB2 Client/Server Environment

Distributed Database Connection Services (DDCS) also has the capability of acting as a DB2 Gateway Server. It allows DB2 database clients transparent and concurrent access to data on any of the DRDA Application Servers (AS). The DRDA AS can be an IBM or non-IBM platform that is DRDA compliant. Therefore, this machine can be a DB2 Gateway Server and DRDA Application Requester (AR) at the same time.

DB2 or DDCS servers can communicate with clients using NetBIOS (DB2 and DDCS for OS/2 only), APPC, TCP/IP, or IPX/SPX as long as the client supports that same protocol. However, communication between DDCS and a DRDA AS is only via APPC (SNA LU6.2 protocol). The APPC or APPN (APPC with dynamic routing) support is provided by Communications Manager/2 on the OS/2 platform and by SNA Server/6000 on the AIX platform.

MPTN (Multiple Protocol Transport Network) allows the SNA LU6.2 protocol to flow on a TCP/IP network. ANYNET/6000 and ANYNET/MVS provide MPTN support.

1.5.1 DB2 Client/Server Communication Products and Protocols

Table 1 can be used as a reference to the protocols supported for a specific DB2 client/server product combination. DB2 Version 2 now supports clients connecting via TCP/IP to a DB2 for OS/2 database server. Also found in Version 2 of DB2 is native support by DB2 for AIX servers for clients that connect using the IPX/SPX protocol.

Table 1. Client/Server Interconnectivity

Client	Communication Protocols Supported	DB2 for OS/2 V2	DB2 for AIX V2	DB2 FOR HP-UX V1.2	DB2 for Solaris V1.2
DB2 Client Application Enabler for DOS V1.2	NetBIOS	Yes	No	No	No
	IPX/SPX	Yes	Yes	Yes•	Yes•
	TCP/IP	Yes	Yes	Yes	Yes
	APPC	No	No	No	No
DB2 Client Application Enabler for Windows V2.1	NetBIOS	Yes	No	No	No
	IPX/SPX	Yes	Yes	Yes•	Yes•
	TCP/IP	Yes	Yes	Yes	Yes
	APPC	No	No	No	No
DB2 Client Application Enabler for OS/2 V2.1	NetBIOS	Yes	No	No	No
	IPX/SPX	Yes	Yes	No	No
	TCP/IP	Yes	Yes	Yes	Yes
	APPC	Yes	Yes	No	No
DB2 Client Application Enabler for AIX V2.1	NetBIOS	No	No	No	No
	IPX/SPX	No	No	No	No
	TCP/IP	Yes	Yes	Yes	Yes
	APPC	Yes	Yes	Yes	Yes
DB2 Client Application Enabler for HP-UX V1.2	NetBIOS	No	No	No	No
	IPX/SPX	No	No	No	No
	TCP/IP	Yes	Yes	Yes	Yes
	APPC	Yes	Yes	No	No
DB2 Client Application Enabler for Sun Solaris V1.2	NetBIOS	No	No	No	No
	IPX/SPX	No	No	No	No
	TCP/IP	Yes	Yes	Yes	Yes
	APPC	Yes	Yes	No	No

Note: •Provided by the FireFox, Inc. NOV*IX for NetWare product.

1.5.2 Back-Level Interconnectivity

DB2 Version 2 also supports many back levels of DB2 products. Table 2 on page 12 shows the down-level support for DB2 products. The CAE and SDK products in Table 2 on page 12 are at the V1.2 level. The DB2/2 and DB2/6000 represent V1.x of those products. The DDCS products are at a V2.2 or earlier level.

Table 2. Back-Level Interconnectivity

Client	Protocol	ES V1.0 Server	DDCS/2 DB2/2	DDCS/6000 DB2/6000
OS/2 Extended Services 1.0	APPC/APPN	Yes	Yes	Yes●
	NetBIOS	Yes	Yes	No
	IPX/SPX	No	No	No
	TCP/IP	No	No	No
DB2/2	APPC/APPN	Yes	Yes	Yes●
	NetBIOS	Yes	Yes	No
	IPX/SPX	No	No	No
	TCP/IP	No	No	No
DB2 CAE/2 or DB2 SDK/2	APPC/APPN	Yes	Yes	Yes●
	NetBIOS●	Yes	Yes	No
	IPX/SPX●	No	Yes	No
	TCP/IP	No	No	Yes
DB2 CAE/DOS or DB2 SDK/DOS	APPC/APPN	No	No	No
	NetBIOS●	No	Yes	No
	IPX/SPX●	No	Yes	No●
	TCP/IP	No	No	Yes
DB2 CAE/6000 or DB2 SDK/6000	APPC/APPN	No	No	Yes●
	NetBIOS	No	No	No
	IPX/SPX	No	No	No
	TCP/IP	No	No	Yes

Note: ●Support provided with V1.2. ●NOV*IX NetWare Server can translate IPX/SPX to TCP/IP ●APPN support provided with SNA Server/6000 V2.1

1.6 Product Distribution

DB2 for OS/2 products are available on CD-ROM and/or 3.5 inch diskettes. The CD-ROM also contains softcopy publications in BookManager READ format.

DB2 for AIX products must be ordered on CD-ROM, but a supplementary 8mm tape option is also available for ease of installation. The charge for the tape option is only for the media. Both the CD-ROM and the tape contain product code as well as the Postscript and INF files for all publications.

The Client Pack CD-ROM, delivered with the DB2 server and the DDCS products, contains the most recent version of every client application enabler. The documentation included with Client Pack is as follows: an installation guide for each client in hardcopy, BookManager and INF formats. Also included are 3.5 inch diskettes for those client platforms for which diskette is the common install media.

1.6.1 Access Keys

Access keys are required for installation of all DB2, DDCS and SDK products. The keys are provided in the CD-ROM and diskette packages only. The pre-generated access key label is on a sheet of paper packaged with the product. These labels are non-transferable and should be kept in a secure place as the products cannot be used until the access keys are entered. Each access key label is unique (unique label serial number and the access key).

The client application enabler can be installed and used without entering an access key.

Chapter 2. Customer Scenario

This chapter describes a customer scenario which may be instructive for you when considering the implementation of DB2 Version 2 and/or upgrading from DB2 Version 1. Almost every chapter in this document looks at this customer environment and details how customers will address their environment in Version 2. Using this example, this document means to provide some insight into the concerns and priorities of the database administrator (DBA).

2.1 Background Information

A telephone company has implemented a comprehensive operational system for its telephone network service. It stores information about all aspects of the network inventory and configuration and automates all customer transactions, such as line requests and connections.

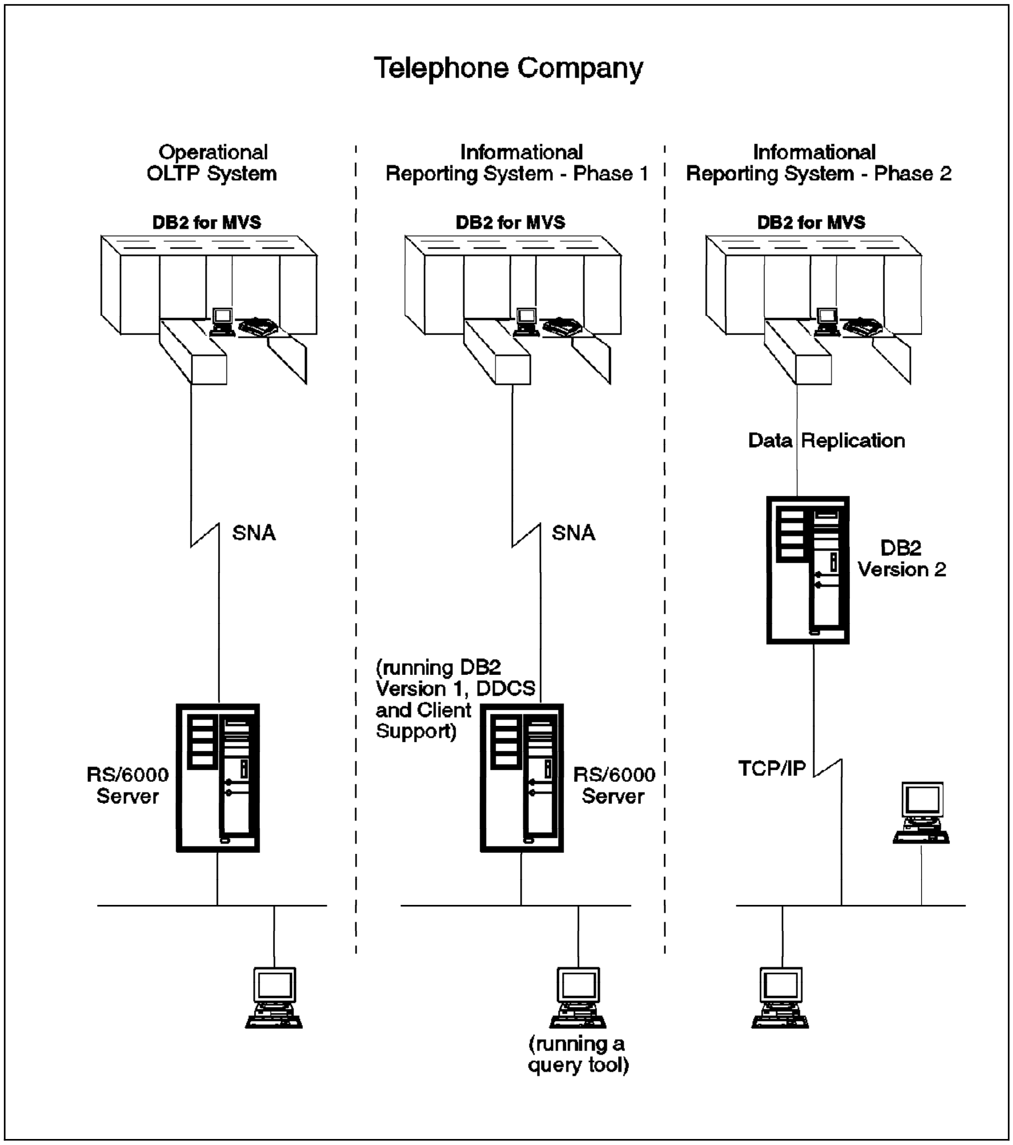


Figure 7. Telephone Company Environment

As Figure 7 shows, the operational online transaction system is executing on an MVS host using DB2 for MVS and is linked to 35 regional RS/6000 servers which, in turn, serve a PC user population in the company′s offices nationwide.

A second requirement subsequently emerged to facilitate a management information system based on the information stored in the DB2 for MVS database. About 300 users (out of 4000) wanted to generate ad hoc reports from the central database information from their own workstations. A distributed database environment was established using DB2 Version 1 (with DDCS) to connect the PC users to the MVS host. Standard SQL query requests are formulated by an end-user query product. For a number of reasons (possibly because the SQL queries were inefficiently constructed), the performance of this configuration was not satisfactory. Currently, users are restricted from issuing queries during peak hours.

2.1.1 Current Status

A solution was proposed to right-size the query server from MVS to AIX. Using DB2 Version 2, a development environment has been established where the DB2 for MVS data will be down loaded from the host onto an RS/6000 server. The database environment will ultimately consist of up to 50 tables containing up to 30 Gigabytes of information (currently 5 Gigabytes). No BLOBs or CLOBs will be used within the database.

2.1.2 Issues and Problems

- The logical design of the database was sourced from the DDL of the DB2 for MVS database. This has given the project team an effective starting point from which to develop the new solution.
- Designing the physical database is more challenging. The approach so far has been on a trial-and-error basis. Depending on performance, relocation of tables and tablespaces may be necessary.
- Backup/Restore policy is another issue that needs to be addressed. Due to the size of the database, the backup policy is one of the concerns of the project team. Frequency and scope of the backups are to be determined.
- On a timely basis (possibly four times per year), a complete reload of the database will be required. These loads will have to be performed in less than 48 hours. Recovery from an “unsuccessful” load must be guaranteed.
- Moving from an unsatisfactory performance system requires that performance should be carefully monitored during the first stages of production.

2.2 Summary

As stated in the Preface, the intention of this book is to guide the DBA through the issues and decisions to be made when planning and installing DB2 Version 2. Prompted by this real-life customer scenario, we will now step through the critical work tasks of a DBA as he/she attempts to solve the problems and issues raised. The chapters are sequenced to reflect the order in which the tasks would be undertaken.

We will look at the following items:

1. The physical design of the database and tablespaces and how they will be placed onto disk.
2. The movement of data into the database.
3. The backup/restore policy that has been determined.

The chapters that follow in this document contain a detailed discussion of subject areas for the DBA followed by (when appropriate) a plan of implementation that the customer has selected.

Chapter 3. Instances and Users

DB2 instances are the key to database administration and security in DB2 Version 2. In this chapter, we will review the concept of instances and the implementation differences between AIX and OS/2. In particular, we will look at how users access and use instances at different levels. The chapter is organized as follows:

- Instances in AIX and OS/2
- Instance Structure
- Creating an Instance
- Instances and Security
- User and Group Support
- Directories and Database Access
- Accessing an Instance

3.1 Instances in AIX and OS/2

A relational database presents data as a collection of tables, with each table consisting of data logically arranged in columns and rows. In addition to data, each database includes a set of system catalog tables which describe the logical and physical structure of the data, a configuration file which contains the parameter values associated with the database, and recovery logs which record ongoing transactions and transactions that can be archived.

The database manger manages the Relational Database Management System (RDBMS) by providing centralized control and independence of data. As well as defining the physical storage of data (in databases), a database manager provides for efficient access to, as well as integrity of, recovery, concurrency control, privacy, and security. In DB2, the database manager is an instance of the DB2 product and its databases.

DB2 is now implemented as instances under AIX and OS/2. Each instance is a unique database-manager environment, that is an environment containing a separate database-manager configuration with one or more databases. Many instances may be created, and run concurrently, on the same physical machine. This allows databases to service various users and applications, which have different requirements for security and performance, without impacting the various users and applications.

3.1.1 Multiple Instances

Multiple instances may be created on a single workstation. This means that you can create several instances, running concurrently, on the same physical machine. This provides flexibility in setting up environments.

Multiple instances provide the following:

- Separate test and production environments
- The ability to tune a database environment for a specific application
- Protection of sensitive information from others

DB2 for OS/2 only supports multiple instances with Version 2.1 or higher. DB2 for AIX Version 2 also supports multiple instances. However, only AIX allows multiple instances and different versions of the product to coexist on the same database server. Figure 8 on page 20 shows two instances on the same physical machine with links to DB2 Version 2.1 and Version 1.2.

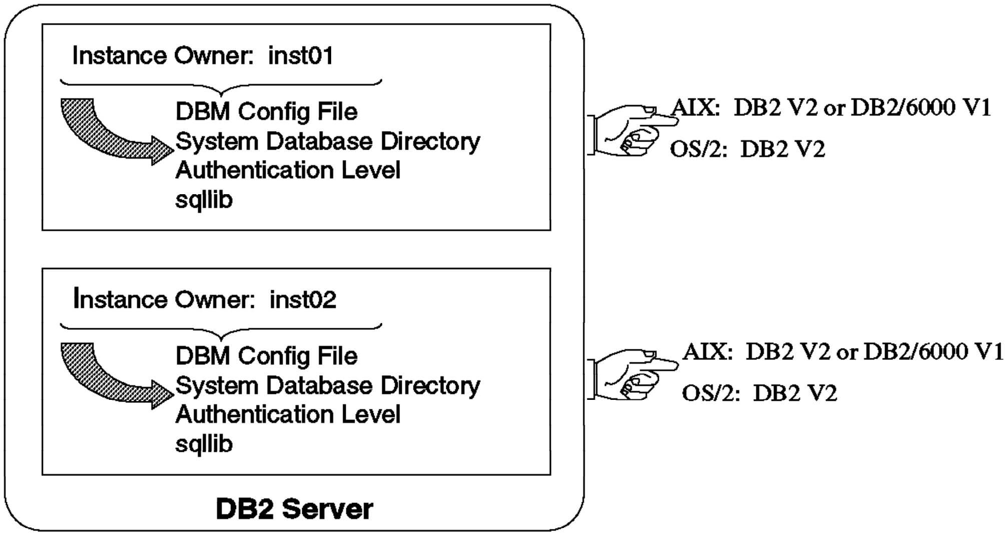


Figure 8. Instances in DB2.

Figure 8 shows two instances on the same server. For both AIX and OS/2, the following are characteristics of instances:

- Each instance has a unique instance owner associated with it. This instance owner is also the DB2 system administrator (SYSADM). For more information on authorizations within DB2, see 3.4.3, “Administrative Control” on page 30. The default instance for DB2 for OS/2 does not have an instance owner associated with it. However, it is recommended that you create an instance owner.
- Each instance has a unique database configuration (DBM) file. The name of the file is db2systm. In this file, you will find most of the parameters that affect the amount of system resources that are allocated to an instance.
- Authentication, specified at the instance level for Version 2 of DB2, is the mechanism that verifies the user′s identity. For more information on authentication, see 3.4.1, “Authentication at the Instance Level” on page 29.
- Every instance owner will have an sqllib directory associated with the instance.

3.2 Instance Structure

The directory structure for DB2 Version 2 is different between AIX (UNIX) and OS/2. These differences are due to the way in which products are installed under the different operating systems.

3.2.1.1 AIX Instance/Directory Structure

When a DB2 Version 2.1 product is installed on an AIX platform, the products will be placed in the /usr/lpp/db2_02_01 directory, while DB2/6000 Version 1 products were in the /usr/lpp/db2_01_01_0000 directory. This naming convention used by AIX makes it possible to install and use multiple versions of DB2 on the same machine.

The directory structure within the db2_02_01 directory is as follows:

/adm	System administrator executable files
/adsm	ADSTAR Distributed Storage Manager files
/bin	Binary executable files
/bnd	Bind files
/cfg	Default system configuration files
/dba	Database Director
/deinstl	Files to reject applied software
/doc/%L	Postscript and Online books for language %L
/function	User-defined functions
/include	C and FORTRAN include files
/include/cobol_mf	COBOL COPY files for Micro Focus COBOL
/include/cobol_a	COBOL COPY files for ANSI COBOL
/instance	Instance scripts
/lib	Libraries
/map	Map files for DDCS for AIX
/misc	Utilities and examples
/msg/%L	Message catalogs for language %L
/netls	iFOR/LS files
/Readme/%L	Readme files for language %L
/samples/c	C sample programs
/samples/cli	DB2 CLI examples
/samples/clp	Command line processor examples
/samples/cobol	COBOL sample programs
/samples/db2sampl	Sample database
/samples/fortran	FORTRAN sample programs
/samples/rexx	DB2 REXX sample programs

Unlike OS/2, AIX systems do not directly use these files within an instance. When an instance is created under AIX, links to these files and directories are created from the instance owner′s home directory. The exception to this is the individual configuration files for each instance.

By using symbolic links to the DB2 installed product, multiple instances can be maintained with minimal use of of additional disk space. When a database is created, the default location for that database is within the instance owner′s home directory. A subdirectory with the same name as the instance is created, and the databases will be stored within this directory.

With this structure, the only limitation is that each instance must have a unique name. As the instance name maps directly to an operating-system user, this implies that each instance must have a unique UserID as the instance owner.

When an instance is created, the following directory structure will be created under the instance owner′s home directory.

SYSADM executable files

sqllib/adm/db2start	Start the database manager

sqllib/adm/db2stop	Stop the database manager
sqllib/adm/db2trc	Trace database execution paths

Configuration files

sqllib/db2systm	Database manager configuration file
sqllib/cfg	Miscellaneous configuration files

Sample scripts to set default DB2 environment

sqllib/db2cshrc	(for C Shell users)
sqllib/db2profile	(for Bourne and Korn Shell users)

Directory for stored procedures and user-defined functions

sqllib/function

Links to the /usr/lpp/db2_02_01 subdirectories

sqllib/adsm	links to /usr/lpp/db2_02_01/adsm
sqllib/bin	links to /usr/lpp/db2_02_01/bin
sqllib/bnd	links to /usr/lpp/db2_02_01/bnd
sqllib/dba	links to /usr/lpp/db2_02_01/dba
sqllib/doc	links to /usr/lpp/db2_02_01/doc
sqllib/include	links to /usr/lpp/db2_02_01/include
sqllib/include/cobol_mf	links to /usr/lpp/db2_02_01/include/cobol_mf
sqllib/include/cobol_a	links to /usr/lpp/db2_02_01/include/cobol_a
sqllib/lib	links to /usr/lpp/db2_02_01/lib
sqllib/map	links to /usr/lpp/db2_02_01/map
sqllib/misc	links to /usr/lpp/db2_02_01/misc
sqllib/msg	links to /usr/lpp/db2_02_01/msg
sqllib/Readme	links to /usr/lpp/db2_02_01/Readme
sqllib/samples	links to /usr/lpp/db2_02_01/samples

Work directories

sqllib/tmp
sqllib/db2dump

The following considerations exist for DB2 for AIX:

1. Each instance must have a unique name.
2. Multiple versions of DB2 may be installed on the same database server.
3. Database names must be unique, even across instances.

3.2.1.2 OS/2 Instance/Directory Structure

When a DB2 product is installed on the OS/2 platform, it will be placed, by default, under a directory called X:\SQLLIB, where X is the target installation drive. The structure of the DB2 directories for OS/2 are as follows:

\sqllib	The SYSLEVEL files, base configuration file and instance directory file
\sqllib\bin	Executable files
\sqllib\bnd	Bind and list files
\sqllib\book	Online book (documentation) files
\sqllib\cfg	Default configuration files
\sqllib\db2	Default DB2 instance configuration files
\sqllib\db2\sqldbdir	Default DB2 instance system database directory files
\sqllib\db2\sqlgwdir	Default DB2 instance gateway directory files

\sqllib\db2\sqlnodir	Default DB2 instance node directory files
\sqllib\tmp	DB2 temporary files
\sqllib\dcslib	DDCS files
\sqllib\dll	Dynamic link libraries
\sqllib\function	System functions
\sqllib\function\unfenced	Unfenced stored procedures. Initially an empty directory
\sqllib\help	Help files
\sqllib\include	FORTRAN and C header files
\sqllib\include\cobol_a	IBM COBOL header files
\sqllib\include\cobol_mf	Micro Focus COBOL header files
\sqllib\install	Installation and maintenance utility
\sqllib\lib	Library files
\sqllib\map	DDCS map files
\sqllib\misc	Miscellaneous applications/utilities
\sqllib\msg\prime	Messages files
\sqllib\samples\c	Sample C files
\sqllib\samples\cli	Sample CLI files.
\sqllib\samples\clp	Sample CLP files
\sqllib\samples\cobol	Sample COBOL files
\sqllib\samples\db2sampl	Sample source files
\sqllib\samples\db2sampl\prime	Sample source files
\sqllib\samples\fortran	Sample FORTRAN files
\sqllib\samples\rexx	Sample REXX files and Windows support README
\sqllib\win	Windows support README file
\sqllib\win\bin	Windows client executable files
\sqllib\win\bnd	Bind files and list files
\sqllib\win\book	Online books
\sqllib\win\help	Help files
\sqllib\win\include	C and COBOL header files
\sqllib\win\lib	Library files
\sqllib\win\msg	Message files
\sqllib\win\samples\c	Sample C files
\sqllib\win\samples\cli	Sample CLI files
\sqllib\win\samples\clp	Sample CLP files
\sqllib\win\samples\cobol	Sample COBOL files

When a new instance is created, the following directories will be created for that instance.

- X:\SQLLIB\instance_name
- X:\SQLLIB\instance_name\SQLDBDIR
- X:\SQLLIB\instance_name\SQLGWDIR
- X:\SQLLIB\instance_name\SQLNODIR
- X:\SQLLIB\instance_name\TMP

Here, instance_name is the name of the instance created. Also, the default directory for databases that are created in any instance is X:\, where X is the drive on which DB2 was installed. This target drive may be modified in the database manager configuration or specified when the database is created.

The following considerations exist for DB2 for OS/2:

1. Each instance must have a unique name.

2. Multiple versions of DB2 may not be installed because the DLL file names will conflict.
3. Database names must be unique, even across instances.

3.3 Creating an Instance

The concept of an instance is the same in both AIX and OS/2. However, there are some differences in implementation. Before you can use DB2 for AIX, you must create an instance; when you install the DB2 for OS/2, a default instance is created for you.

There are certain recommended steps that are the same for creating instances in both AIX and OS/2:

1. Create a group for the instance owner.
2. Create the instance owner.
3. Customize the environment of the instance owner for DB2.

3.3.1 Instance Creation in AIX

Each instance in DB2 is represented by its user. This user must be a member of one primary group. First, create an instance owner group and user as root. Enter the license information that was provided with the product.

Then you can create the instance. The db2instance command found in DB2/6000 Version 1 is now called db2icrt in order to be consistent between AIX and OS/2. The db2icrt command is invoked as follows:

```
/usr/lpp/db2_02_01/instance/db2icrt instanceowner -a authentication
```

In Version 2, you can specify the authentication type of the instance at creation time. Valid authentications are SERVER, CLIENT or DCS. Note that this authentication level applies to all databases under this instance. If the authentication type is not specified, it will be determined when the first database is created in the instance (default authentication is SERVER). The files created during instance installation are located in INSTHOME/sqllib, where INSTHOME is the home directory of the instance owner. This directory contains symbolic links to directories in /usr/lpp/db2_02_01 and includes all files and sub-directories for use by the database manager. To prevent a potential loss of data if an instance is deleted, you should not create any files within this directory structure. Exceptions are UDF libraries and stored procedures.

All users belonging to the group defined for the instance owner will become SYSADM. This means that all users will have DB2 system administrator authority for a given instance. The SYSADM_GROUP parameter is set to this primary group owner and cannot be changed.

After creating an instance, you must set the DB2 environment for each user who needs access to this instance and its databases.

3.3.1.1 AIX Configuration

For this example, we used the UserID db2 and the primary group db2adm. The process of creating the users/groups in AIX is as follows:

As the root user

1. Create the instance owner′s primary group if it does not exist.

   ```
   # mkgroup db2adm
   ```

2. Create the instance owner′s UserID if it does not exist.

   ```
   # mkuser -a pgrp=db2adm groups=db2adm db2
   ```

3. Create the database instance.

   ```
   # /usr/lpp/db2_02_01/instance/db2icrt db2 -a server
   ```

   This will set the authentication type to server.

4. Enter the licensing information. This may be a nodelock file entry or an iFOR/LS license.

5. Modify user profiles to include the instance profile. Here, we assume the home directory of the instance owner is /home/db2.

   ```
   /home/db2/sqllib/db2profile
   ```

> **Note**
>
> Once the environment has been configured, you will be able to access online help if it has been installed. To start the online manuals, use the command db2help from an aixterm.

As the instance owner, db2

6. Start the database engine so that local users may access it.

   ```
   $ db2start
   ```

3.3.2 Instance Creation in OS/2

The installation of DB2 for OS/2 has been enhanced to provide the same look and feel as other software products from IBM. The installation process involves the loading of DB2 product files, setting the environmental variables in the CONFIG.SYS file and the automatic creation of one default instance called DB2.

There are some differences between the instance creation tool under AIX and OS/2. When you create an instance under AIX, you can specify the authentication level and parameters for fenced user-defined functions (UDFs). These options -a and -u are not valid under the db2icrt under OS/2. An error will be returned if they are specified. When you issue the db2icrt to create an individual instance, you can only provide the instance name. The authentication level is set to SERVER. Therefore, if you want to create an instance with authentication CLIENT or DCS, you must create it as SERVER, and then update the DBM authentication parameter.

3.3.2.1 OS/2 Configuration

The process of creating the users/groups in OS/2 is done via the User Profile Management (UPM) Service. When DB2 is installed on the OS/2 platform, a default instance, called DB2, is automatically created. For consistency with AIX (UNIX) platforms and ease of administration, it is recommended that you create your own instance with a different UPM-created UserID and GroupID. Alternatively, as in this example, create a UserID and administration group that will be the administrative user and group for the DB2 instance. In the following example, db2 will be the instance owner, and db2adm will be the administrative group.

The steps to do this are as follows:

1. Using the UPM administrator logon, create the group db2adm and the user db2. You will also need to add the user db2 to the group db2adm.

   > **UPM Note**
   >
   > If you have not used UPM before, the default administrator UserID is "USERID," and the default password is "PASSWORD." It is recommended that you change this immediately.

2. Enter the licensing information for the nodelock file. This file will be in one of the following locations.

   a. %DB2PATH%\NODELOCK

   b. %DB2INSTPROF%\NODELOCK

3. Set the DB2INSTANCE environment variable. This will usually be set up in CONFIG.SYS.

4. Check the database manager configuration.

   ```
   [C:\] db2 GET DATABASE MANAGER CONFIGURATION
   ```

   or

   ```
   [C:\] db2 GET DBM CFG
   ```

5. Make any desired changes. For example, to set up the system administration group to db2adm, you would use the command:

   ```
   [C:\] db2 UPDATE DBM CFG USING SYSADM_GROUP db2adm
   ```

6. Start the database manager

   ```
   [C:\] db2start
   ```

3.3.3 Client/Server Connectivity Configuration

A DB2 Version 2 database can support both local and remote clients concurrently. Some form of communication support is required on all remote clients to support the communication protocol they choose to use. The communications support components for the desired protocol(s) need to be installed. For a listing of what protocols are supported with what DB2 products, see Chapter 1, "Product Overview" on page 1. The following table summarizes the tasks needed by database servers and remote clients to communicate using DB2.

Table 3 (Page 1 of 2). Client/Server Connectivity Tasks

Task	Protocol			
	APPC	TCP/IP	NetBIOS	IPX/SPX
Install correct level of products to support protocols used	YES	YES	YES	YES
Apply required maintenance to products	YES	YES	YES	YES
Update environment variables on server	YES	YES	YES	YES
Update environment variables on client			YES	

Table 3 (Page 2 of 2). Client/Server Connectivity Tasks

Task	**Protocol**			
	APPC	**TCP/IP**	**NetBIOS**	**IPX/SPX**
Update communication profiles on server	YES			
Update communication profiles on client	YES			
Specify Local LU (Optional)	YES			
Update DBM on server	YES	YES	YES	YES
Update DBM on client			YES	
Catalog Node Directory on client	YES	YES	YES	YES
Catalog Database Directory on client	YES	YES	YES	YES
Register DB2 server on NetWare File Server •				YES
Update hosts file on client		YES		
Update services file on client		YES		
Update services file on server		YES		
Set up client and server for DCE Directory Services (Optional)	YES	YES		
Bind client packages on servers •	YES	YES	YES	YES

Note:

1. When using file server addressing only. Not required when using direct addressing on all clients.
2. Required only when client platform is different from server platform.

For more detail on the setup of client/server environment, there are other manuals available, such as *Installing and Using DB2 Clients for Windows* and *Installing and Using AIX Clients*.

3.3.3.1 DB2 Communication Environment Variables

DB2 Version 2 has a number of environment variables that need to be set to enable communication. For example, DB2COMM must be set to support remote clients. The setup differs for OS/2 and AIX.

- For a DB2 or DDCS for OS/2 server, this is done by editing CONFIG.SYS and adding the entry:

  ```
  SET DB2COMM=XXXXX
  ```

  where XXXX is the protocol the client will use, such as NETBIOS, APPC, TCPIP, or IPXSPX. More than one protocol may be defined by using a comma as a separator, such as:

  ```
  SET DB2COMM=TCPIP,APPC,IPXSPC,NETBIOS
  ```

- For a DB2 or DDCS for AIX server, this is done by editing the db2profile script and specifying:

  ```
  DB2COMM=XXXX
  export DB2COMM
  ```

  This script can be invoked directly or added to a user's .profile or .login file.

3.3.4 Other Administrative Commands

In the previous section, we discussed commands to create an instance in DB2. There are some additional commands in DB2 Version 2 to manipulate instances. In DB2 for AIX, these commands are located in the /usr/lpp/db2_02_01/instance directory. You must have root authority to use them. In DB2 for OS/2, they are located in \sqllib\xxxxxxx.

- db2ilist

  Returns a list of all instances that exist on the workstation.

- db2iupdt

  Updates the instance. The parameter -u is the name of the user under which fenced user defined functions will be run. The default is the user "nobody."

- db2idrop

  Removes an instance.

  Note that any cataloged databases will not be removed by issuing this command.

- db2imigrev (AIX only)

  If for some unexpected reason you must reverse an instance migration, it is possible by using the db2imigrev command. This will restore the INSTHOME/sqllib_v1 directory to INSTHOME/sqllib and create a copy of the Version 2 INSTHOME/sqllib to INSTHOME/sqllib_v2. To migrate back to Version 2 of DB2, execute db2imigr again. This restores Version 2 of DB2 from the copy.

3.4 Instances and Security

If you are moving or migrating to DB2 Version 2, it is a good idea to understand the authentication mechanism and security features before you start the move. This may save considerable time in reconfiguring or perhaps remove the risk of security violations.

There are two levels of security that control access in a database system. The first controls access to a database system, and the second controls access within the database system.

- Access to the database system is managed by operating- system functions. These functions allow the system to authenticate the user and to control access to objects such as files and programs. The concepts of users and groups are the basis of this mechanism. AIX makes sure, for example, that the user who is logging in is who he/she claims to be.
- Access within a database manager instance is controlled by DB2 itself. The concepts of administrative authorities and user privileges are the basis of this mechanism.

The database administrator must ensure that sensitive information are not accessed by those without a "need to know." A plan for controlling database access should be developed by defining your objectives for a database access-control scheme and specifying who shall have access to what objects and under what circumstances.

3.4.1 Authentication at the Instance Level

The first step in managing security is to verify the user′s identify. This is called authentication. Authentication DB2 Version 2 is set at the instance level. This means that all databases within an instance must have the same authentication level.

Authentication is the operation of verifying whether a user is who he/she claims to be, based on the username and password. Every time users try to connect to a local or remote database, they are authenticated. DB2 relies on the operating system to perform this authentication. In UNIX or AIX systems, the authentications allows the user or group to exercise control over objects such as files and programs. OS/2 provides this level of control via User Profile Management (UPM) Services.

3.4.2 Levels of Authentication

There are three authentication levels in DB2 Version 2. They are:

- SERVER Authentication

  This is the default setting. The user is authenticated at the server machine. The user name and password flow from client to server in ASCII format. The server will then authenticate them before processing any requests.

- CLIENT Authentication

  The server machine will assume that the client machine has already authenticated the user. The server will process the request using the supplied UserID without further authentication.

- DCS Authentication

  DCS authentication is similar to server authentication for DRDA connections. This will allow the host database to authenticate the user. If authentication is set to server when connecting to a host database, authentication will take place at the gateway machine.

Client authentication should only be used on secure networks. This is because the server does not determine which machine the client user is on. Given this, it is possible in an unsecure network for someone to introduce a machine with any UserID and be able to access the server using that UserID without the requirement of a password.

In DB2/2 Version 1, the only supported authentication level was server authentication. DB2/6000 Version 1.0 supported mixed authentication types within a single database, while DB2/6000 Version 1.2 reduced this to one authentication type for each database, but still permitted mixed authentication types within the instance. In DB2 Version 2, the authentication type is defined at the instance level, and it applies to all databases within the instance. Table 4 summarizes the authentication levels for different levels of DB2.

Table 4 (Page 1 of 2). Authentication Levels in DB2.

Product/Version	**Authentication Type**
DB2/2 Version 1.x	Server Only
DB2/6000 Version 1.0	Mixed authentication types allowed in one database
DB2/6000 Version 1.2	One authentication type/database, but can mix authentication types of database within instance.

Table 4 (Page 2 of 2). Authentication Levels in DB2.

Product/Version	Authentication Type
DB2 Version 2 (OS/2 and AIX)	One authentication type/instance

3.4.2.1 Setting Authentication Levels

In DB2 Version 2, the authentication level must be the same for databases contained in that instance. The setting of what the authentication level is for the instance can be done in one of the following three ways:

1. At instance creation, you can define the authentication for that instance and for all databases contained in that instance.
2. The authentication level can be determined when the first database is created in the instance.
3. If not done explicitly, as in the above steps, then the authentication will be set to SERVER at first database creation.

To change the authentication type after the instance has been created, the database manager configuration file must be changed. The instance must be stopped and restarted for the changes to take effect.

3.4.3 Administrative Control

After authentication, the second step is to control the ability to operate on objects in the database. This is called access control. Access control is a combination of privilege and authority. Privilege is the right to perform an operation on a database object. Authority defines a set of privileges. Authority levels are SYSADM, DBADM, SYSCTRL, SYSMAINT, and users.

Users are the lowest level of authority within the database. Users here are User_ID and Group_ID. These are the two types of authorizations, individual and group. An individual ID is an ID assigned to a particular person. A group ID consists of one or more individual IDs. If an individual ID is a member of a group, that individual ID automatically has all privileges granted to the group.

The group ID PUBLIC plays a particular role in the access control mechanism of the database. All users accessing a database are automatically members of the group PUBLIC. Granting a privilege to PUBLIC grants that privilege to all users of the database.

In DB2 Version 1, there were two levels of administrative control:

- SYSADM - Highest level of authority applies to an instance
- DBADM - Applies to a specific database

Each of these authority levels gives the user a defined set of privileges. A privilege can be defined as the right to perform an operation on a database object. There are a large number of privileges in the database environment; some of these include:

- CREATE
- SELECT
- INSERT
- UPDATE

- EXECUTE

To grant a privilege to a user or group, the SQL grant statement is used. To revoke a privilege, the SQL revoke statement is used.

DB2 Version 2 has added new authorization levels. Authorization levels within a database now include:

- System Administration Authority (SYSADM)
- System Control Authority (SYSCTRL)
- System Maintenance Authority (SYSMAINT)
- Database Administration Authority (DBADM)
- User Authority

The authority levels, SYSADM, SYSCTRL and SYSMAINT, are not granted in the same way as privileges. The authority levels are associated with groups through the database manager configuration. To grant a user the required authority level, you need to add that user to the associated group. These authorities are for the entire instance and are not limited to an individual database.

3.4.3.1 Hierarchy of Authorizations

Authorization levels in DB2 provide a hierarchy for administration capabilities. This can be seen in Figure 9, or a more complete list of the operations each authorization level provides can be found in Table 5 on page 33. At the top of this hierarchy is the System Administrator or SYSADM. SYSADM is able to perform any of the DB2 administration operations as well as select any information from any database that exists within the instance.

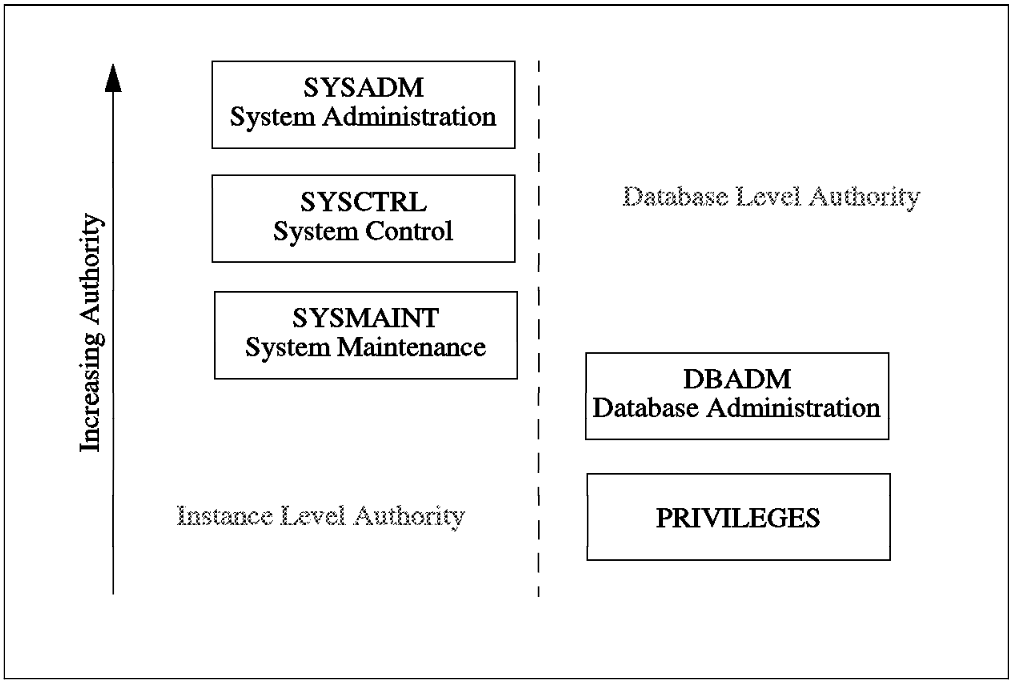


Figure 9. Authorization Heirarchy

System control (SYSCTRL) provides the ability to perform almost any administration command. The SYSCTRL user does not have authority to access user information or modify the database manager configuration.

System Maintenance (SYSMAINT) is restricted to maintenance operations, such as backup and restore of databases, update of database configuration and database monitoring.

A listing of the new authorities and privileges are as follows:

1. **SYSCTRL** - System Control

SYSCTRL offers almost complete control of database objects defined in that DB2 instance, but cannot access user data directly, unless granted the privilege to do so. A user with this authority, or higher authority, can perform the following functions:

- Update the database, node and Database Connection Services (DCS) directory
- Update database configuration parameters
- Create or drop a database
- Force applications
- Quiesce the DB2 instance or database
- Run the RESTORE DATABASE utility on a new database
- Create or drop a tablespace

2. **SYSMAINT** - System Maintenance

This authority allows the execution of maintenance activities, but cannot access user data. Only users with this level of authority or higher (SYSADM or SYSCTRL) can do the following tasks:

- Update database configuration files
- Backup databases and tablespace
- Restore to an existing database
- Restore tablespace
- Start and stop DB2 instance
- Run the database monitor
- Start and stop traces

At the database administration level, there is the DBADM authority. Database Administration authority (DBADM) is a special case. Although this is an authority and not a privilege, it may be granted in the same way as a privilege. However, it may not be granted to the special group PUBLIC. The creator of a database will automatically have DBADM authority on that database. DBADM authority is for a single database only. It is possible to hold DBADM authority for multiple databases. DBADM provides some system-level administration, such as loading tables, some tablespace operations and event monitoring. The DBADM has complete authority over the database, however. Using it, an administrator may query, drop or create any table and set the privileges for users within the database.

To specify authorization levels, three new parameters have been added to the database manager configuration file. These parameters are SYSADM_GROUP, SYSCTRL_GROUP and SYSMAINT_GROUP. For example, you are the system administrator on an instance and want to have another trusted person do some of the system control functions. Create a group on your system, INSTCRL, for example. Then update the database manager configuration file with following:

```
db2 update database manager configuration using SYSCTRL_GROUP INSTCTRL
```

This will give SYSCTRL to any user who is in the group INSTCTRL. Remember to stop and restart the database manager instance to read the configuration file with any changes.

There are some important considerations when running an AIX (UNIX) server where users may log into the server machine. Providing authority levels higher than user authority may require you to give the user write permission on certain directories. For instance, anyone with SYSCTRL authority will require write permission if they are going to be able to update the database, node or DCS directories. You should consider using access control lists to limit the write privilege to specific users or groups. Access control lists are created and maintained by the `acledit`, `aclget` and `aclput` commands.

Table 5 shows all the authorities and privileges that exist for DB2 Version 2.

Table 5 (Page 1 of 2). Database Authorities.

Function	SYSADM	SYSCTRL	SYSMAINT	DBADM
CATALOG/UNCATALOG DATABASE	YES			
CATALOG/UNCATALOG NODE	YES			
CATALOG/UNCATALOG DCS	YES			
UPDATE DBM CFG	YES			
GRANT/REVOKE DBADM	YES			
GRANT/REVOKE SYSCTRL	YES			
GRANT/REVOKE SYSMAINT	YES			
FORCE USERS	YES	YES		
CREATE/DROP DATABASE	YES	YES		
QUIESCE INSTANCE OR DATABASE	YES	YES		
CREATE/DROP/ALTER TABLESPACE	YES	YES		
RESTORE TO NEW DATABASE	YES	YES		
UPDATE DB CPG	YES	YES	YES	
BACKUP DATABASE OR TABLESPACE	YES	YES	YES	
RESTORE TO EXISTING DATABASE	YES	YES	YES	
PERFORM ROLL FORWARD RECOVERY	YES	YES	YES	
START/STOP DATABASE INSTANCE	YES	YES	YES	
RESTORE TABLESPACE	YES	YES	YES	
RUN TRACE	YES	YES	YES	
TAKE DBM OR DB SNAPSHOTS	YES	YES	YES	
QUERY TABLESPACE STATE	YES	YES	YES	YES

Table 5 (Page 2 of 2). Database Authorities.

Function	SYSADM	SYSCTRL	SYSMAINT	DBADM
UPDATE LOG HISTORY FILES	YES	YES	YES	YES
QUIESCE TABLESPACE	YES	YES	YES	YES
LOAD TABLES	YES	YES		YES
SET/UNSET CHECK PENDING STATUS	YES	YES		YES
READ LOG FILES	YES	YES		YES
CREATE/ACTIVATE/DROP EVENT MONITORS	YES	YES		YES
RUN LOAD UTILITY	YES	YES		YES

3.5 User and Group Support

Some operating systems will allow the same name for a user and a group. This may cause confusion for authorization and privilege checking within DB2.

In general, there are three possible scenarios.

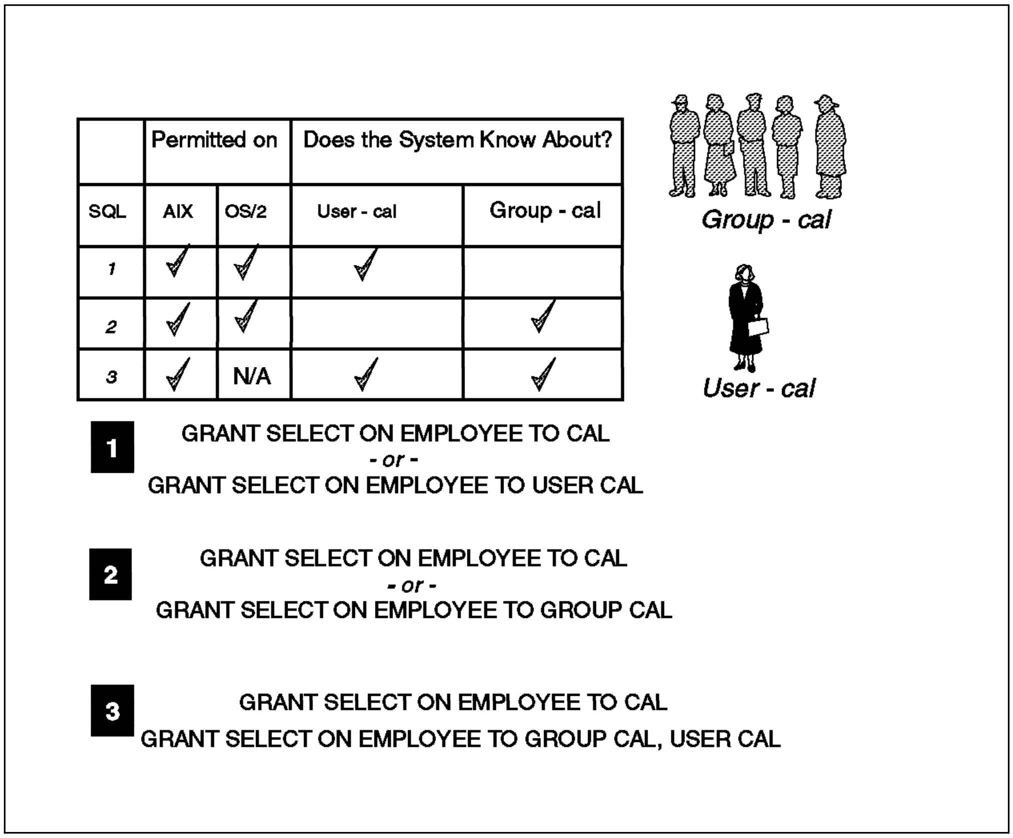
1. The name is unique to a user.
2. The name is unique to a group.
3. The name refers to both a user and a group.

In the first two scenarios, there is no problem since it should be clear as to whom the authority or privilege is being granted.

The third scenario is not possible on OS/2 platforms since identical user and group names are not permitted. However, it is possible for this to occur on AIX or UNIX platforms.

In DB2/6000 Version 1, the use of the environment variable, DB2GROUPS, was used to distinguish between group and user privileges. The DB2GROUPS environment variable is no longer required, or used, in DB2 Version 2.

Figure 10 on page 35 shows how user and group support is implemented in DB2 Version 2. Note how OS/2 does not allow a user and group to have identical names.



	Permitted on		Does the System Know About?	
SQL	AIX	OS/2	User - cal	Group - cal
1	✓	✓	✓	
2	✓	✓		✓
3	✓	N/A	✓	✓

1 GRANT SELECT ON EMPLOYEE TO CAL
- or -
GRANT SELECT ON EMPLOYEE TO USER CAL

2 GRANT SELECT ON EMPLOYEE TO CAL
- or -
GRANT SELECT ON EMPLOYEE TO GROUP CAL

3 GRANT SELECT ON EMPLOYEE TO CAL
GRANT SELECT ON EMPLOYEE TO GROUP CAL, USER CAL

Figure 10. Group and User Support in Version 2

In DB2 Version 2, changes have been made to the `grant` and `revoke` SQL statements to add an optional parameter in the TO/FROM authorization-name clause which is used to indicate if the privilege is intended for a user or group. As a result, if a user and a group have the same name in the previous database version, the authority and privilege to the group must be explicitly re-granted after migration. This condition check may result in a member of a group no longer having authority for database objects that he had in V1.x. If the user within the group still requires the authority, then you must explicitly grant the authority at the group level. To overcome this problem, you are able to specify either `to user` or `to group` in the `grant` command. Figure 10 illustrates three scenarios on both AIX and OS/2. They are as follows:

1. A user named cal exists on either AIX or OS/2. The select privilege may either be implicitly or explicitly given. The select permission will be granted only to the user.

2. A group named cal exists on either AIX or OS/2. The select may either be implicitly or explicitly given. Because no user named cal exists, the select permission will be granted only to the group.

3. Only AIX allows a user and group to have the same name. If the select does not specify the `to user` option explicitly, the select permission will be granted to the user cal. Otherwise, you may explicitly grant permission to the group by using the `to group` option.

Equivalently, the `revoke` command may include either `from user` or `from group` to indicate where the authority or privilege is to be removed. If the additional `to/from` clause is not included, then the default will be for user.

During migration, the authorization catalog tables are checked to determine if existing privileges are for users or groups, and the GRANTEETYPE (′U′ or ′G′) is determined. The following condition check occurs:

- If GRANTEE in a V1.x table is a USER or undefined, then GRANTEETYPE is set to U.
- If GRANTEE in a V1.x table is a GROUP, then GRANTEETYPE is set to G.
- If GRANTEE in a V1.x table is both a USER and a GROUP, then GRANTEETYPE is set to U.

3.6 Directories and Database Access

Access to both local and remote databases is done through the use of directories. The directories hide the requirement for a user to know where a database actually resides. Users are able to connect to local databases, remote databases and host or DRDA databases simply by specifying a database name. The directories that make this possible are:

- Node Directory
- Database Connect Services Directory
- Database Directory

Each database client maintains a node directory. The node directory contains entries for all instances that a client will access. The node directory is used to contain communications information about the network connection to the instance. If multiple instances exist on a remote machine, then each instance must be cataloged as a separate node before you will be able to access any information contained within the instance.

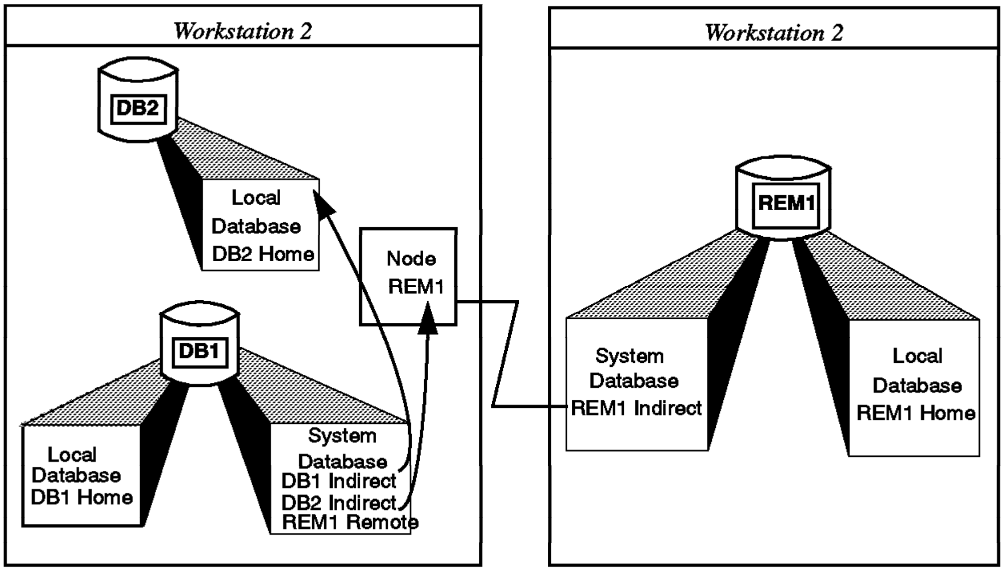


Figure 11. DB2 Directory Structure

DB2 Version 2 includes the capability to catalog other instances on the same machine as a local node. The implications of this are discussed in 3.7.2, “Local Access” on page 39.

The connection information for DRDA-connected databases, that is databases residing on a host machine, is different from the information for LAN-connected databases. Due to this, DB2 uses a different directory to catalog this type of database. The directory used is the Database Connection Services (DCS) directory. This directory will only exist if DDCS has been installed on your system. The DCS directory stores information used by the database manager to access databases on a host computer. For these type of databases, there is no node directory entry; this is because the DCS directory contains all the required information to connect to the host database.

The final type of directory used is the database directory. There are two database directories, the System Database directory and the Local Database directory.

The System Database directory resides in the sqldbdir subdirectory, which is located in the instance directory. This directory is used to access both local and remote databases. The directory contains the database name, alias, type, and node where the database resides. If the database is local, the directory where the database resides is included. The System Database directory also contains authentication information.

The Local Database directory resides in every subdirectory that contains a database. It is used to access local databases in that subdirectory. Each entry in the directory contains the database name, alias, type, and storage information about the database.

Each time a user connects to a database, these directories are used to locate the database. To help reduce the amount of time it will take to connect to a database, DB2 Version 2 has included the option to cache the directory information in memory. This is set by the database manager configuration parameter DIR_CACHE. The default value is yes, which stores the directory information in memory. The performance benefits of this are best seen on a gateway machine that is processing a large number of requests or where the application issues multiple connection requests.

3.6.1 DCE Directory Services

Distributed Computing Environment (DCE) Cell Directory Services (CDS) is supported in DB2 Version 2. CDS provides a mechanism for centralizing database directories. The CDS server maintains the database directories, and the participating clients will reference the CDS server to resolve directory entries. This centralization of database directories means that if a catalog update needs to be done, then it only need be done on the CDS server and not on the individual clients.

3.7 Accessing an Instance

This section discusses the ways in which a database user (other than the instance owner) accesses instances and objects belonging to instances, such as a database. Accessing an instance can be divided into two sections:

- Remote access
- Local access

3.7.1 Remote Access

Remote access is accomplished by making entries into the directories found in DB2. For more information on directories, see 3.6, "Directories and Database Access" on page 36. The administrator of the client machine must configure the system to communicate with the DB2 server. This will depend on the operating system, either AIX or OS/2, and the communication protocol that the DB2 server is using. The supported protocols are:

- NetBIOS - OS/2 only
- TCP/IP - AIX or OS/2
- APPC - AIX or OS/2
- IPX/SPX - AIX or OS/2

The client user wanting access to a remote DB2 server will do the following:

1. Catalog the remote node
2. Catalog the DB2 server

The cataloging of the remote database server will depend upon the communication protocol being used. For example, to catalog a DB2 server using TCP/IP, enter:

```
db2 catalog tcpip node nodename remote hostname server service-name
```

where:

- `nodename` is a unique name for the server node. This unique name may be the same as the TCP/IP hostname. The name will be used to catalog the remote database.
- `hostname` is the TCP/IP hostname of the server.
- `service-name` is the service name entered in the /etc/services file on the client. The port number on the client must match the port number that the DB2 server is using for that instance.

To catalog the remote database at the client, enter:

```
db2 catalog database database-name as alias at node nodename
authentication server
```

where:

- `database-name` is the name of the database on the DB2 server machine.
- `alias` is the name by which the database will be known at the client.
- `nodename` is the unique name used in the catalog node command.
- `authentication server` is the type of authentication used. This must match the authentication as defined on the DB2 database server. Valid choices are server, client and dcs.

Once the client has cataloged the remote server node and database, it is possible to either connect to a database or attach to a remote instance.

3.7.1.1 Remote Administration

DB2 Version 2 provides the capability for remote client administration of a server node. By attaching to an instance, the client is able to perform functions such as the following:

- Create/drop databases
- Get/update/reset database manager configuration
- Get/update/reset database configuration
- Monitor a database
- Backup/restore rollforward database
- Load data
- Force applications

To attach to another instance, you would use the command:

```
ATTACH [TO nodename] [USER username [USING password]]
```

If the attach command is specified without any arguments, the current attachment status is returned. Also, if you attach to an instance while already attached to a different instance, the current instance will be detached before the new attachment is attempted.

Database connections are independent of instance attachments. A single application may maintain several database connections at the same time it has an instance attachment, but may only maintain a single instance attachment at any one time.

When attached to another database, you will still be using the directory services of the local instance, which is determined by the DB2INSTANCE variable. Because of this, you will not be able to perform catalog changes, list, update or reset commands on databases, nodes or DCS directories.

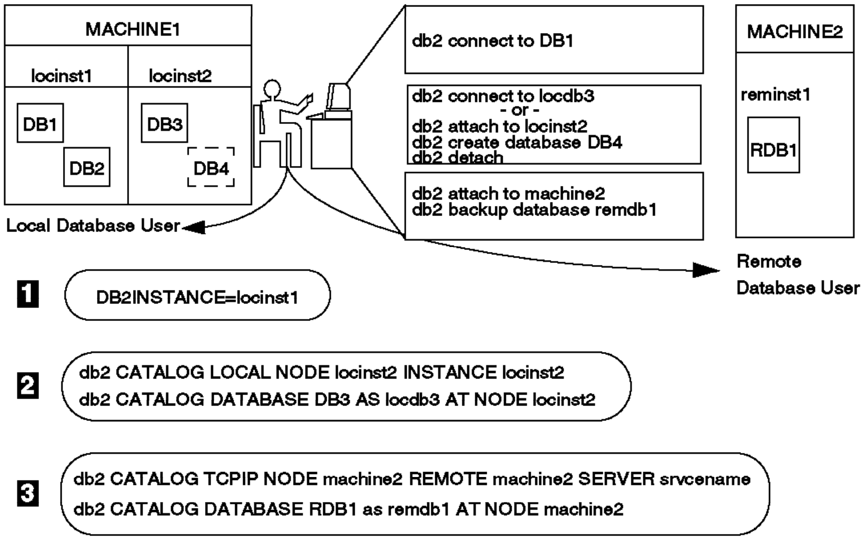
If you create a new database, the database will be created at the attached instance, and a catalog entry will be created in the local instance directories which refer to the remote database. To drop the database at the remote instance, the local directory must contain an entry for the database.

3.7.2 Local Access

If the database user is local, there are several ways to access a local instance:

- Set the client environment so that the local user has defined the DB2INSTANCE and DB2PATH variables.
- Catalog a local instance

The following scenario illustrates a user accessing two local instances and one remote instance.



```
db2 connect to DB1
```

```
db2 connect to locdb3
        - or -
db2 attach to locinst2
db2 create database DB4
db2 detach
```

```
db2 attach to machine2
db2 backup database remdb1
```

```
DB2INSTANCE=locinst1
```

```
db2 CATALOG LOCAL NODE locinst2 INSTANCE locinst2
db2 CATALOG DATABASE DB3 AS locdb3 AT NODE locinst2
```

```
db2 CATALOG TCPIP NODE machine2 REMOTE machine2 SERVER srvcename
db2 CATALOG DATABASE RDB1 as remdb1 AT NODE machine2
```

Figure 12. Example - Local and Remote User in DB2

Figure 12 is explained as follows:

1. The local user has set the DB2INSTANCE and DB2PATH variables to allow access to the locinst1 instance. In AIX, this user would edit his/her .profile to include the following entry:

   ```
   . /home/locinst1/sqllib/db2profile
   ```

2. The local user also wants access to another instance on the same server machine. The local user will catalog the other instance as a local node and catalog the local database. The catalog node command is as follows;

   ```
   db2 catalog local node locinst2 instance locinst2
   ```

   where:

   - `local` identifies that this is a local instance.
   - `locinst2` is a name that is to particular to local client user and will be referred to in the `catalog database` command.
   - `locinst2` is the local instance name as it is defined on the server.

   The catalog database command is:

   ```
   db2 catalog database db3 as locdb3 at node locinst2
   ```

   where:

   - `db3` is the actual name of the database as it exists on the server.
   - `locdb3` is the alias that the local client user will use to refer to the database.
   - `locinst2` is the reference name given to the local instance by the client user in the `catalog node` command.

3. The local user has also cataloged a remote node, reminst1. For more information about the cataloging of a remote node, see 3.7.1, “Remote Access” on page 38.

3.7.3 Accessing the DB2 Server

The authentication level of the instance will affect the client, either local or remote, when connecting to a database or attaching to an instance. The other consideration is the type of client machine attempting access.

3.7.3.1 AIX Client

When an AIX client connects to a DB2 server, there are two forms of the connect statement to consider. One specifies the user name and password, and one does not. When connecting to a DB2 server with authentication server, the following behavior results from the connect statement:

```
connect to <database> user <userid> using <password>
```

The user name/password combination is sent to the DB2 server for authentication. This information is checked against what is contained in the /etc/passwd file on the AIX DB2 server or in UPM on the OS/2 DB2 server.

The following format of the connect statement is not allowed from an AIX client with authentication server on the DB2 server:

```
connect to <database>
```

Authentication client means that the AIX client user is validated at the local workstation. When using client authentication, there are two forms of the connect statement that are valid:

```
connect to <database> user <userid> using <password>
connect to <database>
```

3.7.3.2 OS/2 Client

When an OS/2 client connects to a database server, there are two forms of the connect command to consider with an authentication type of server.

```
connect to <database> user <userid> using <password>
```

If the OS/2 client is accessing an AIX DB2 server, the UserID/password is sent to the AIX server for authentication there. UPM on the OS/2 client is not involved in this process. This means that the UserID/password will flow as typed in on the command line. The AIX DB2 server is case sensitive. It will check this information against what is stored in the /etc/passwd file. There must be an exact match.

The other form of the connect statement to consider is:

```
connect to <database>
```

Here, the node name that is stored in the client database directory is retrieved and sent to UPM. Since UPM is involved in this form of the connect statement, there are three cases to consider:

1. Remote logon

   If the client user on the OS/2 workstation has already done a remote logon to the database server through UPM, then the user ID/password is retrieved from UPM and sent to the database server.

2. Local logon

   If the client user is logged on locally at the OS/2 workstation, then UPM retrieves the local UserID/password. This information, in turn, gets passed to the database server for authentication.

3. Not logged on

   If a logon has not occurred on the OS/2 workstation, the user is then prompted for a remote logon using the node information that was passed to UPM by the CAE client code. The UserID/password also gets validated at the database server.

If the OS/2 client is accessing an AIX DB2 server, the password in this type of connect statement is converted to uppercase. This leads to the requirement that while the UserID created on AIX for OS/2 users can be defined in lowercase, the password must be stored on the AIX server in uppercase. UPM will fold the password into upper case. To match on the AIX server, the password in /etc/passwd must also be in uppercase.

When the DB2 server has authentication client and the client is an OS/2 client, two forms of the connect statement are valid:

```
connect to <database> user <userid> using <password>
connect to <database>
```

The UserID and password are passed to UPM for validation that this is a valid local user. This information is checked in the NET.ACC file on the OS/2 workstation.

3.7.3.3 DOS/Windows Client

With the authentication type of server on the DB2 instance, the DOS/Windows client supports three types of connect syntax. One type specifies a UserID (login name) and password; one specifies the UserID only, and one specifies neither UserID nor password.

They are as follows:

```
connect to <database> user <userid> using <password>
```

The UserID and password are sent to the database server unencrypted for authentication.

```
connect to <database> user <userid>
```

With an authentication type server and no password specified on the connect statement, the password will be extracted from the DOS environment variable, DB2PASSWORD. If this variable has not been set, an error will be returned with an SQLCODE of -1403.

```
connect to <database>
```

If the authentication type is server and neither UserID nor password is specified on the connect statement, the UserID will be extracted from the DOS environment variable, DB2USERID. The password will be extracted from DB2PASSWORD. If either of these are not set, an SQLCODE of -1403 is returned. Otherwise, the information stored in these variables is sent to the server for authentication.

For client authentication, there are two forms of the connect statement that are valid:

```
connect to <database> user <userid> using <password>
connect to <database>
```

When using client authentication, the UserID specified with the connect statement is passed to the database server to determine database privileges. For an AIX database server, if the user is not a valid AIX user, the user will get the default privileges associated with the AIX group, PUBLIC.

3.7.3.4 Binding the Database Utilities

When a client wants access to a database server where the operating system is different from the client machine, another step must be taken. The database utilities must be bound on the server for the client to use. The database utilities that a remote client must bind to the DB2 server include:

- Ad hoc SQL functions in the Command Line Processor (CLP)
- DB2 Call-Level Interface
- The binder program
- Import and export functions
- The `reorg` command

If you create a new database on the server, packages for the database utilities must also be created in the database′s system catalog tables. All of the client packages are contained in a db2ubind.lst file. These packages may be created by executing the `bind` command from the client workstation. Before executing this command, you must first connect to the server database, and then execute the `bind` command. For example, to bind all the DB2 CAE/DOS packages, execute the following commands:

```
connect to <database>
bind <path>@db2ubind.lst blocking all
```

where

- `<database>` is the name of the database as it is cataloged on the client.
- `<path>` is the full pathname of the db2ubind.list file, such as \sqllib\bnd or C:\sqllib\bnd.
- `@` indicates to the `bind` command that what follows is a file containing a list of bind files.
- `db2ubind.lst` contains the list of bind files.
- `blocking all` allows for row blocking to be performed.

The `bind` command must be run separately for each database that you wish to access. If you have different types of clients coexisting on your network, you must bind the utilities from each type of client. The utilities only have to be bound once from each type of client.

Chapter 4. Data Placement

This chapter discusses the storage model for Version 2. We will begin by reviewing the storage model for Version 1 on both OS/2 and AIX. The storage model for DB2/2 in Version 1 created the database files on a single disk drive with the exception of log files. Log files for OS/2 in Version 1 could be placed on a different drive. However, the size of the database was limited to the size of the hard drive. For OS/2, the physical limitations were as follows:

- Maximum table size determined by maximum file size
- Maximum database size determined by size of the drive

Log files for AIX in Version 1 could be also be placed on a different physical volume from the database. DB2/6000 Version 1 based its storage model on tables that were segmented into multiple directories. Each table in DB2/6000 was given a corresponding file name, for example, SQL00001.DAT. These AIX files were stored in directories. Thus, you had the ability to spread the contents of one table, for example, SQL00001.DAT, over multiple directories. These file systems could be mounted over these AIX directories, if your database needed to be greater than the 2GB size limitation imposed by an AIX file or file system.

4.1 Storage Concepts in DB2 Version 2

The enhancements in the storage model for DB2 on OS/2 and AIX allow for user-controlled location of database objects onto different media. It further provides the capability to store data onto these physical devices directly, no longer requiring that a file system be used to store data as in Version 1. These enhancements lead to:

- Improved performance
- More flexibility in database configuration
- More portability
- More granularity for database administration

The storage model in DB2 Version 2 has led to new or different definitions of storage items than was found in Version 1. The concepts this section will discuss are:

- Container
- Tablespace
- Extent
- Database
- Table Objects

4.1.1 Container

A container is a generic term used to describe the allocation of physical space. Figure 13 on page 46 shows that a container can be any of the following:

- File
- Directory

- Device

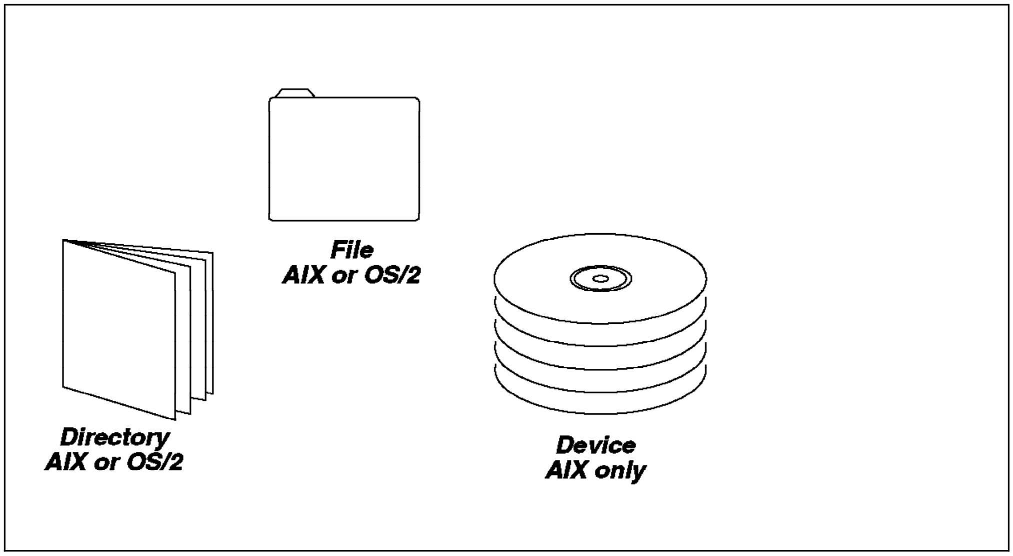


Figure 13. What Does a Container Look Like?

The type of container depends on the type of tablespace and the platform. For example, in AIX, a device container is a logical volume.

4.1.2 Tablespace

The storage model in DB2 Version 2 is based upon a logical layer between the database and its tables called the tablespace. The concept of a tablespace is not a new one to the DB2 family. Tablespaces are also implemented in DB2 for MVS.

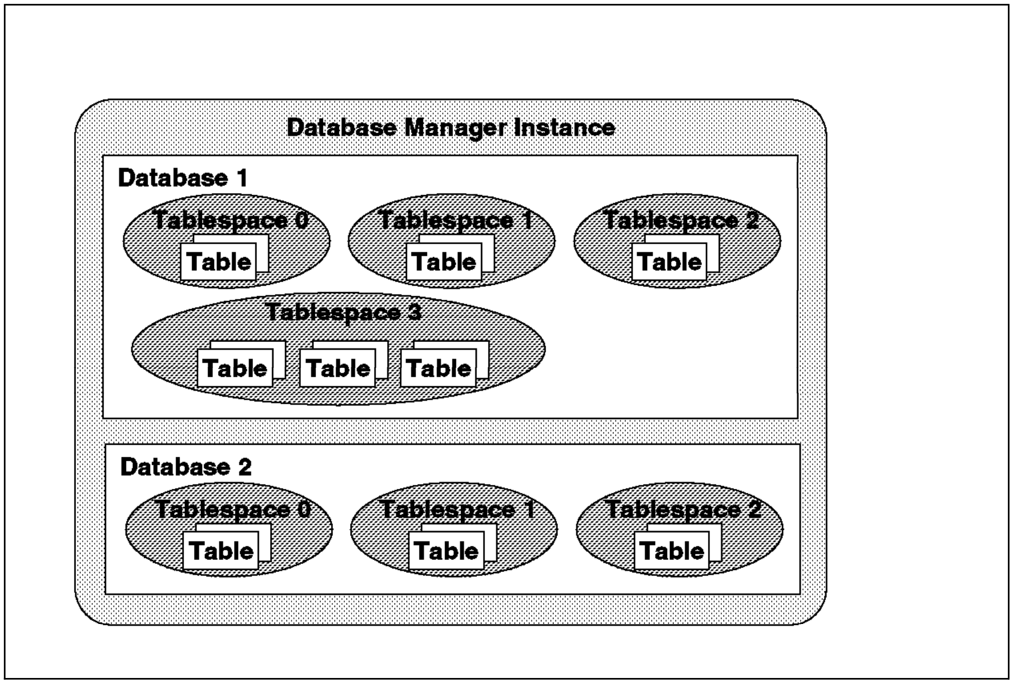


Figure 14. Database Manager Instance, Database, Tablespace and Tables

Figure 14 shows the relationship between tables, tablespaces and databases found in one database manager instance on a server. The instance is created.

A database, Database 1, is created within the instance. A tablespace may be created with defaults or explicitly with the `create tablespace` command.

Tables may be created in tablespaces as shown. A tablespace can contain more than one table. For example, Tablespace 3 in Database 1, has three tables created in it.

There are two ways to create tablespaces:

1. `create database` command
2. `create tablespace` command

4.1.2.1 Database, Tablespaces and Table Objects

In DB2, a database is made up of a grouping of different tables of data. A table holds different types of objects or parts. There are basically five types of storage objects that make up a table. They are:

- DAT - contains the column data
- LF - contains the LONG VARCHAR columns defined in the table
- LOB - this is the data that is stored in BLOBs (Binary Large Objects) or CLOBs or DBLOBs
- LOBA - this the allocation control type of information about LOB data
- INX - if the table has an index defined on it, it is contained here.

Each of these storage objects that belong to a table can exist in only one tablespace. These storage objects (table parts) are basically a set of functionally related pages. The type of tablespace determines how these objects are stored.

4.1.2.2 Tablepsaces and Containers

There is a one-to-many relationship between a tablespace and containers. Multiple containers may be defined for a tablespace, however, a container can only be assigned to one tablespace. Figure 15 shows this one-to-many relationship.

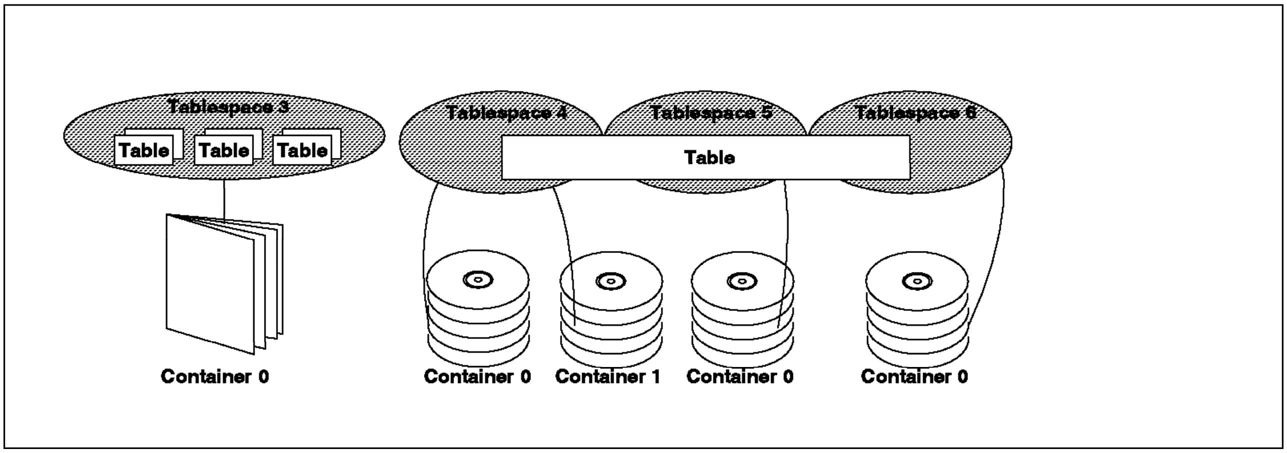


Figure 15. Tablespace and Container, One-to-Many Relationship

A container is the allocation of storage to a tablespace. The physical space may or may not exist at the time the tablespace is created depending on the tablespace type. Figure 15 shows tablespace 3 with only one container (a directory) assigned to it. A tablespace may have one or more containers assigned to it. However, a tablespace cannot share containers. This is shown in tablespace 4 which has two containers, Container 0 and Container 1 assigned to it. Container IDs are assigned in numerical order to a tablespace, starting from 0.

Figure 15 on page 47 shows another important concept about tablespaces. Depending on the type of tablespace, it is possible for a table to span tablespaces. The restriction is that the data must be of the same type in the tablespace. Tablespace 4, 5 and 6 contain only one table but span multiple tablespaces. Note that the same one-to-many relationship still exists between tablespaces and containers, even if the table spans multiple tablespaces.

4.1.3 Extent

An extent is an allocation of space, within a container of a tablespace. The extent is allocated to a single database object. This allocation consists of multiple pages.

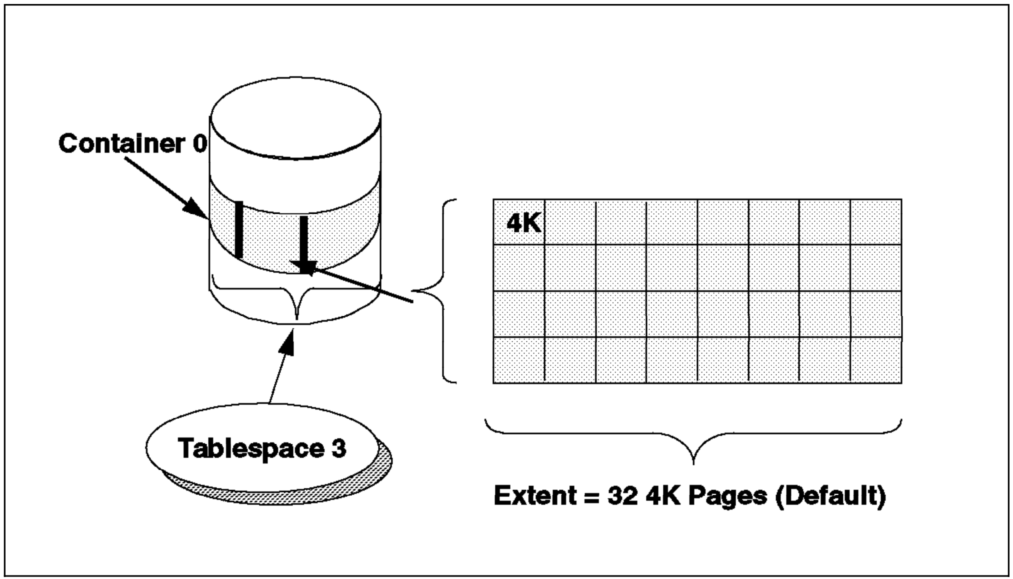


Figure 16. Extent, Container and Tablespace in DB2 Version 2

Figure 16 shows the relationship between an extent, a container and a tablespace. The tablespace is initialized when it is created. As part of this initialization, an allocation size is set for the tablespace. This parameter is called the extent size. The tablespace page size is 4096 bytes (4K). The default extent size is 32 4K pages.

4.1.4 Summary

The following concepts will be discussed in this chapter as they relate to data placement:

Container Directory or file in AIX and OS/2, logical volume in AIX only. Containers cannot be shared among tablespaces.

Tablespace The logical layer between the physical tables and the database. Containers are assigned to tablespaces.

Extent An allocation of space within the container of a tablespace. It is the number of pages written into by the database manager in a container before writing to another physical allocation of the same size.

Table A set of rows and columns. A table may be assigned to one tablespace. Although, several tables can share the same tablespace. Depending on the type of tablespace, a table may span multiple tablespaces.

Table Objects Also called table parts or storage objects. This is the data in the table and any related parts, such as the index of the table.

Database A collection of tablespaces that hold tables.

4.2 SMS and DMS Tablespaces

DB2 Version 2 supports two kinds of tablespaces:

- System Managed Space or Storage (SMS) Tablespace
- Database Managed Space or Storage (DMS) Tablespace

System Managed Storage or System Managed Space (SMS) tablespaces are a generalization of the storage model found in Version 1. SMS tablespaces are called System Managed because as in Version 1, the database manager uses the operating system′s file system. For OS/2, the file system type is the High Performance File System (HPFS) or File Allocated Tables (FAT). For AIX, the file system is the Journaled File System (JFS).

Database Managed Storage (DMS) tablespaces are characterized by tablespaces that are assigned pre-allocated storage. This storage can be a device, such as a logical volume in AIX, or a file in OS/2 and AIX. The database manager is responsible for the management of this space. The use of the logical volume in AIX in a DMS tablespace is often referred to as raw I/O.

4.2.1 Container Options with SMS and DMS Tablespaces

The following table lists the container options for SMS and DMS tablespaces for AIX and OS/2:

Table 6. Container Options for SMS and DMS Tablespaces

Operating System	File	Directory	Device
OS/2	DMS	SMS	Not Applicable
AIX	DMS	SMS	DMS

4.2.1.1 Using Directories as Containers

SMS tablespaces only use directories as containers. Table parts (storage objects are built by creating files within these directories. Directories are created to be used as a container for an SMS tablespace when the tablespace is created. The name of the directories is specified when the tablespace is created. In the following example, four directories were specified as containers when the tablespace was created. The names specified were /db2data/mydir1, /db2data/mydir2, /db2data/mydir3 and /db2data/mydir4. The instance owner, inst01 in the example, must have read and write permission in the /db2data filesystem. In AIX, after the filesystem (or directory) is created, the system administrator must change ownership of it so that the instance owner may write into it. If inst01 is the instance owner whose primary group is db2adm, the AIX system administrator should issue the following command:

```
$ chown inst01.db2adm /db2data
```

To check the result, you can issue the following command:

```
$ ls -la /db2data
drwxr-s---   2 inst01 db2adm    512 Aug 23 11:56 mydir1
drwxr-s---   2 inst01 db2adm    512 Aug 23 11:56 mydir2
drwxr-s---   2 inst01 db2adm    512 Aug 23 11:56 mydir3
drwxr-s---   2 inst01 db2adm    512 Aug 23 11:56 mydir4

$
```

4.2.1.2 Using Files as Containers

Files can be used as containers only for DMS tablespaces. These files do not have to be previously created.

A file is created as a container for a DMS tablespace in one of the following:

- The DMS tablespace is created either at create database or create tablespace time
- A container is added to a DMS tablespace

In both cases, the containers are specified by the name of the file and the number of 4 KB pages assigned to each container. See the syntax of the `create tablespace` and the `alter tablespace` statements in 4.5.7, “Creating Tablespaces” on page 72 and 4.6, “Managing Tablespaces” on page 77. The following screen shows the file created when the container was created. The name specified for the file was /db2data/contts01 and the container was given 200 4 KB pages. Note that space is allocated when the container is created. The owner and the permissions of the file are not specified when the container is created. The instance owner will be the owner of the file. Group permissions will be associated with the instance owner′s primary group.

```
$ ls -la /db2data/contts01
-rw-------   1 inst01   db2adm    819200 Aug 23 11:25 /db2data/contts01

$
```

4.2.1.3 Using Devices as Containers

When planning to use a device in AIX as a container for a DMS tablespace, the device has to be previously created by the AIX system administrator. Devices used as containers are usually logical volumes on the AIX platform. When a logical volume is created, the AIX system administrator provides the following information, either by accepting the default values or by entering specific values:

- Name of the logical volume
- Size of the logical volume. The size is specified as number of logical partitions. The size of each partition is 4 MB.
- Physical volume or volumes where the partitions should be allocated.
- Desired position in the physical volume of these partitions, in the edge, middle or center of the physical volume.
- Type of logical volume. The logical volume has to be a raw logical volume (as opposed to JFS) to be used as a container.

The result of the creation of a logical volume is a special file in the /dev directory. The following screen shows the file created when a logical volume is created with name volts1.

```
$ mklv -y′volts1′ rootvg 4
$ ls -la /dev/volts1
brw-rw----   1 root     system    20,  9 Aug 23 08:39 /dev/volts1
$
```

The container created has two devices, the block special device and the character special device. The character special device will automatically have an “r” in front of the name you have given it at creation.

```
$ ls -la /dev/rvolts1
crw-rw----   1 root     system    20,  9 Aug 23 08:42 /dev/rvolts1
$
```

The instance owner must have read and write permissions on only the character special device part. The AIX system administrator must grant read and write (r/w) permissions to the instance owner on the device. The AIX chown command will change permissions.

```
$ ls -la /dev/rvolts1
crw-rw----   1 root     system    20,  9 Aug 23 08:48 /dev/rvolts1

$ chown inst01.db2adm /dev/rvolts1
$ ls -la /dev/rvolts1
crw-rw----   1 inst01   db2adm   20,   9 Aug 23 08:52 /dev/rvolts1
```

One of the main differences between using the file, directory or device for containers is that the device must exist before it can be used. The device (logical volume in AIX) can be used for DMS tablespaces in any of the following:

- When the DMS tablespace is created, either when creating the database or creating a tablespace
- When adding a container to an existing DMS tablespace

In both cases, the containers must exist before using. The container is specified by the name of the special file and the number of 4 KB pages assigned to each container. See the syntax of the `create tablespace` and the `alter tablespace` statements on 4.5.7, “Creating Tablespaces” on page 72 and 4.6.3, “Alter Tablespace” on page 80 for examples.

4.3 System Managed Storage (SMS) Tablespaces

System Managed Storage is based on the storage model found in DB2 Version 1. Containers in SMS tablespaces have some of the following characteristics:

- The container in an SMS tablespace does not pre-allocate its storage. There will be some space used during tablespace creation for overhead.
- Containers cannot be added to a SMS tablespace after the tablespace is created.
- The total number of containers in a SMS tablespace is determined either at `create database` or at `create tablespace` time.

4.3.1.1 Default Tablespaces

When creating a database, three tablespaces must either be specifically created or defaults will be used. These tablespaces will hold the following database objects:

- System catalogs
- Temporary space
- User data

The tablespace that holds the system catalogs cannot be dropped or changed after the `create database` command is issued. There is only one tablespace for the system catalogs. The tablespaces that holds temporary data or user may be changed after creation of database or other tablespaces. There may be multiple tablespaces for temporary data or user data depending on your database environment. There must be at least one temporary tablespace for use by the database at all times.

If you do not explicitly specify tablespaces when creating a database, three default SMS tablespaces are created.

Adding Tablespaces

You can add tablespaces after the database is created by using the `create tablespace` command.

The default SMS tablespaces are based on the storage model found in Version 1 of DB2. Figure 17 on page 53 shows the directory structure of the database and SMS tablespaces when the `create database` command is entered and default values are selected..

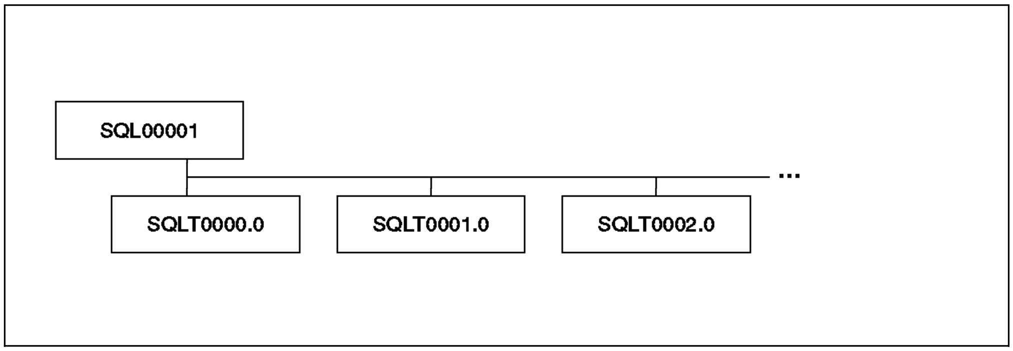


Figure 17. Default SMS Tablespaces

Figure 17 shows DMS tablespaces that are created if default values are selected at `create database` time. Not all the database structures are represented here. Figure 20 on page 62 shows the other default objects that are created when a database is created.

In AIX, if you accept the default directory for database placement, it is determined by the Default Database Path (DFTDBPATH) parameter of the database manager configuration. When the instance is created, this path is set to the instance owner′s home directory. The value of this parameter can be changed with the `update database manager` command. Figure 17 shows this as the "database directory." The default placement is a subdirectory with the same name as the instance owner under the home directory of the instance owner. For example, if the instance owner is db2, the default placement of the three SMS tablespaces is /home/db2/db2.

In OS/2 the default placement of a database is under the drive and directory specified by the Default Database Path (DFTDBPATH) parameter of the database manager configuration. By default, DFTDBPATH is the root directory of the the drive where the DB2 code is installed.

Regardless of platform, the first database will be placed in the SQL00001 directory. SQL00001 is a subdirectory of the database directory as shown in dflt.. SQL00001 consists of three containers assigned to three default SMS tablespaces:

1. SQLT0000.0 is the container that holds the system catalogs. The system catalog tablespace contain all the system catalog tables for the database and cannot be dropped. This tablespace is called SYSCATSPACE.
2. SQLT0001.0 is the container that holds temporary tables that are created and removed during normal processing. This tablespace is called TEMPSPACE1.
3. SQLT0002.0 is named USERSPACE1. This is the default SMS tablespace that holds all user defined tables.

4.4 Database Managed Storage (DMS) Tablespaces

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

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

- If the container is a file, it is created when the tablespace is created and dropped when the tablespace is dropped.
- If the container is a logical volume in AIX, the container must exist before creating the tablespace. After dropping the tablespace, the logical volume still exists and must be removed.
- Storage space is pre-allocated to a container when a container is created.
- Containers can be added to a tablespace using the `alter tablespace` statement.

One of the biggest differences and advantages to using DMS tablespaces over SMS tablespaces is the ability to span a table over multiple tablespaces. When creating a table, you can decide to place certain objects of the table in different tablespaces. DMS allows you to store Large Object Data (LF and LOBs) and indexes in different tablespaces. A table can then split up across three different tablespaces. The tablespaces used to store the table are selected when the table is created.

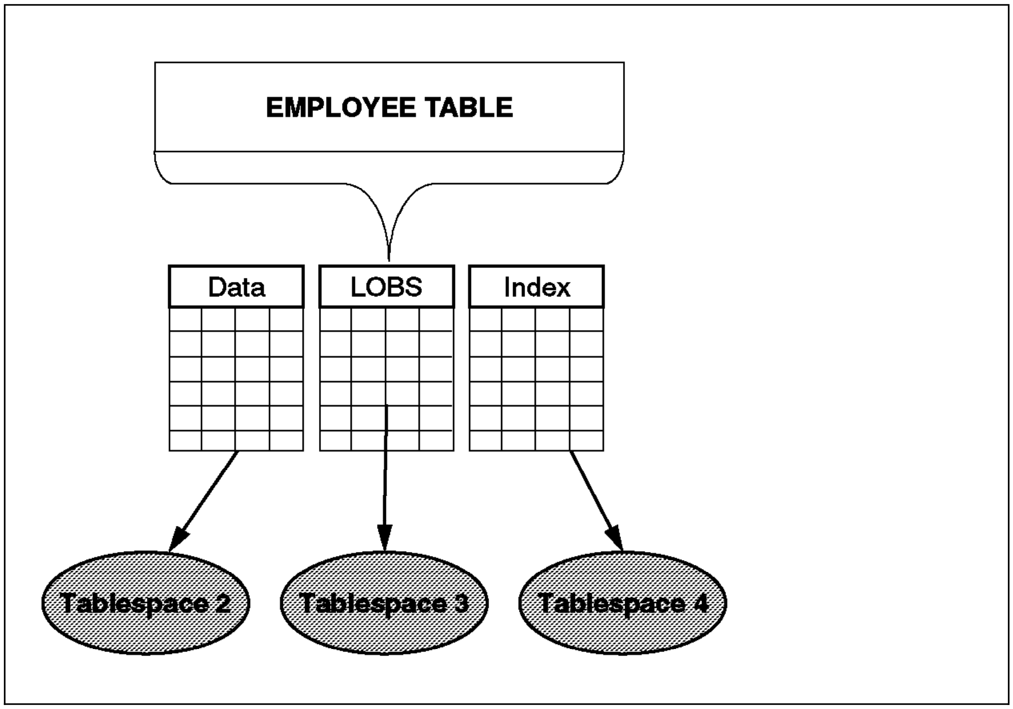


Figure 18. Tables in DMS Tablespaces

Figure 18 shows the EMPLOYEE table being split among three DMS tablespaces. The regular table data is placed in Tablespace 2. LOBS have been placed in Tablespace 3 and indexes in Tablespace 4. In DMS tablespaces, the tablepsace is defined by a set of operating system filenames. These file names may be device names. The storage objects or table parts are built by allocating extents from the tablespace.

4.4.1 SMS and DMS Tablespace Considerations

When choosing between SMS and DMS tablespaces, there are many factors to consider. Consider the operational characteristics of the three tablespace types:

- Catalog tablespace.

  Only one catalog tablespace can exist in a database. Although a choice is possible, SMS is recommended unless you will have very large system catalogs. (Here, very large is defined as a file that exceeds the size limitations imposed by the operating system.) From an administration point, SMS will be easier to maintain. Also, it is not possible to add containers to an SMS tablespace should you exceed size limits for the filesystem/directory.

  When the system catalog tablespace (SYSCATSPACE) is created the database manager creates a set of tables, indexes and views. The space needed for the SYSCATSPACE is approximately 200 extents. Should you select DMS for the system catalogs and use the default extent size, you must understand the amount of pre-allocated storage that would be required. The default extent size is 32 pages, each page of 4 KB. If SYSCATSPACE were to be a DMS tablespace pre-allocated space would be allocated with the `create database` command. This means that the necessary space for the system catalogs to be created as a DMS tablespace would be approximately 200 x 32 x 4K.

- Temporary tablespace.

  Temporary tablespaces are used by the database manager to create temporary tables. SMS offers the best choice for the temporary tablespace. The size of the temporary tablespace can peak to a high value while sorting large tables. Its peak size is difficult to predict, and, if using DMS, you would have to allocate a large amount of space that you would not be able to recover.

  There is no performance advantage in having more than one temporary tablespace. If there are multiple temporary tablespaces, they are accessed in a round-robin manner at run-time. It would be very difficult to predict which temporary tablespace (if multiple ones existed) would be used. The temporary tablespace should be at least large enough to support the largest temporary table that might be generated during run-time.

- User tablespaces.

  The choice between user SMS and DMS tablespaces is not an obvious choice. It depends on many factors, such as

  – Size of database

  – Physical resources such as amount and type of disk

  – Performance

  – Type of operations on the data, such as frequent joins

  – Data types such as BLOBs, CLOBs or DBLOBs

  You will have to weigh the advantages and disadvantages of SMS and DMS tablespaces. You could even decide that a mix of tablespaces, some of them SMS tablespaces and some of them DMS tablespaces, is the best alternative for your database.

4.4.1.1 Advantages of SMS User Tablespaces

There are two possible advantages to using SMS tablespaces for user data:

- Size of database and total disk in system

  Storage space is not pre-allocated for SMS tablespaces. Storage needs of tablespaces are provided by the operating system. When a tablespace is dropped, the space used by the tablespace is immediately recovered. If disk space is constrained in your system and your database is small, you might want to consider using SMS user tablespaces. The definition of small is debatable. For AIX, small would be 2 gigabytes or less. However, even a database that is 2 GB might want to take advantage of the benefits of DMS tablespaces.

- Ease of administration
- SMS tablespace administration is easier than DMS tablespace administration. You do not have to pre-allocate space, so there is no need to create containers before using each tablespace. Also, if your database was small, several hundred MB or less, you could do an offline backup of the database when backing up the entire operating system. For example, in AIX, performing a mksysb on an AIX system containing a small database would be easy to administer. The database in this situation must not be in use.

4.4.1.2 Advantages of DMS Tablespaces

This storage model has important benefits when compared to the SMS storage model. Some of the advantages are the following: Table objects can be accurately placed in the physical devices. Performance may increase when the operating system layer is avoided during I/O operations.

- Placement of database objects

  Tables may be split across multiple DMS tablespaces, allowing the separation of table parts.

- Maximizing system resources

  By splitting data among tablespaces, you can direct table parts to specialized disks.

- Flexibility

  By controlling the placement of database objects, you can control not only access, but administrative tasks like backup and restore. You can place less frequently accessed items like BLOBs that may store images that once created are neither accessed nor frequently updated.

- Performance

  Space allocation is cheaper in DMS tablespaces. If using devices for DMS tablespaces, you avoid the filesystem overhead.

- Scalability

  DMS tablespaces allow you to add containers to the tablespace online. Rebalancing of the data is done immediately and automatically when a container is added.

Ask the questions found in the table that follows to help you determine whether your user tablespaces should be SMS or DMS:

Table 7. Characteristics of SMS and DMS User Tablespaces

Characteristics	**SMS Tablespace**	**DMS Tablespace**
Can Increase Number of Containers in Tablespace	No	Yes
Can Store Index Data in Separate Tablespace	No	Yes
Can Store Long Data in Separate Tablespace	No	Yes
One Table can Span Several Tablespaces	No	Yes
Space allocated only when needed	Yes	No
Table Objects can be Directed to Different Types of Disk	Yes	Yes

4.5 Planning Your Tablespace Environment

When the logical design of your database is done, you have to deal with the physical design of your database. This design will depend on the resources, mainly disk space or disk drives you have available. The objective of planning your user tablespace environment is to determine the type of tablespace and the placement of the user tables in the tablespaces.

To efficiently plan your environment, you will need to gather information about the size of your tables and indexes. The list of tasks that you will need to accomplish is the following:

1. Logical design of tablespaces
2. Creating the database
3. Sizing of tables and tablespaces
4. Determination of container characteristics
5. Creation of containers and tablespaces
6. Creation of tables and indexes

To illustrate the process of planning a tablepsace environment, we will use one example throughout this section. It will consist of a database, dss, comprised of eight tables.

4.5.1 Logical Design of Tablespaces

There are two approaches when grouping tables into tablespaces:

- Create one tablespace for each table.

  This policy allows you to make individual backups of each table, as backups can only be performed at tablespace or database level. You can recover the space allocated to the container after dropping the table and the tablespace.

- Group related tables into tablespaces.

Tables that are related through referential contraints or triggers can be grouped together. Backup and restore can be made at tablespace level, reducing the time needed to restore the database to a consistent level.

Unrelated tables can be also grouped together depending on the size of these tables. A group of small tables may be grouped into a tablespace to use a different extent size.

Once you have grouped tables into tablespaces, you may wish to further categorize each tablespace by performance characteristics: one for index data, one for LOB and Long Field Data, and one for regular table data. Thus, if you are going to define indexes or use LOBs and Long Field Data, you will have to create more tablespaces than you might otherwise expect.

In our example (see Figure 19 on page 59), we grouped our tables into two groups to facilitate our backup policy. We will have to create one tablespace for each group of tables. Additionally, we want to create a tablespace to store all the indexes of our tables and a tablespace to store LOBs. The six tablespaces that will be used by our database dss are:

- SYSCATSPACE. This is an SMS tablespace that will contain the system catalogs.
- TEMPSPACE1. This is an SMS tablespace that will be used for temporary tables that are built during processing.

  USERSPACE1 will also be created, but will be dropped. SYSCATSPACE and TEMPSPACE1 are created using default values during the create database command.
- ts01. This tablespace will store the regular data of the ORDERS, OPTIONS1 and OPTIONS2 tables.
- ts02. This tablespace will store the regular data of the rest of the tables (CUSTOMERS, LOCATIONS, SITEMAPS, EMPLOYEE, EMPDEPT)
- index01. This tablespace will store all the indexes of our tables.
- lobs01. This tablespace will store the LOB table objects of the ORDERS, SITEMAPS and LOCATIONS tables.

  Tablespaces ts01, ts02, index01 and lobs01 will all be DMS tablespaces. They will be created explicitly with the create tablespace command.

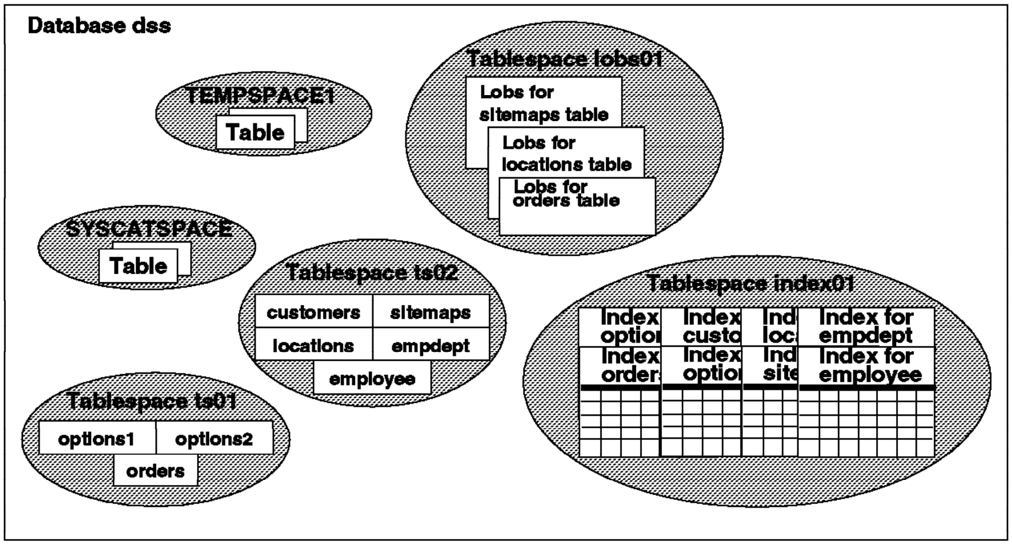


Figure 19. Example Database and Tablespace Environment

4.5.2 Creating a Database

The create database command has changed from DB2/6000 Version 1. You can create tablespaces when creating a database. Alternatively, you can use the create tablespace statement to create tablespaces. For our example, we will use the defaults for creating a database, with the exceptions that we will use a different path for the database, /db/dss. The system catalogs and temporary space will be SMS tablespaces and will be stored in subdirectories under /db/dss. Later, we will drop USERSPACE1 and create DMS tablespaces. The syntax for our example is the following:

```
db2 ″create database dss on /db/dss″
```

4.5.2.1 Create Database Syntax

The syntax of the create database command is:

```
►►──CREATE DATABASE──database-name──┬──────────────────────┬──┬──────────────────────────┬──────────►
                                     └─ON──┬─path (AIX)──┬─┘  └─ALIAS──database-alias─┘
                                           └─drive (OS2)─┘

►──┬───────────────────────────────────────────────┬──┬──────────────────────────────┬──────────►
   └─USING CODESET──codeset──TERRITORY──territory─┘  └─COLLATE USING──┬─SYSTEM───┬─┘
                                                                       └─IDENTITY─┘

►──┬───────────────────┬──┬─────────────────────────────────┬──────────►
   └─NUMSEGS──numsegs─┘  └─DFT_EXTENT_SZ──dft_extentsize─┘

►──┬───────────────────────────────────────┬──┬────────────────────────────────────┬──────►
   └─CATALOG TABLESPACE──┤ tblspace-dfn ├─┘  └─USER TABLESPACE──┤ tblspace-dfn ├─┘

►──┬─────────────────────────────────────────┬──┬─────────────────────────────┬──────────►◄
   └─TEMPORARY TABLESPACE──┤ tblspace-dfn ├─┘  └─WITH──"comment-string"─┘
```

tblspace-dfn:

```
├──MANAGED BY──────────────────────────────────────────────────────────────►

                       ┌─,──────────────────┐
►──┬─SYSTEM USING──(───▼─'container string'─┴──)──────────────────────────┬──────►
   │                     ┌─,──────────────────────────────────────────────────┐   │
   └─DATABASE USING──(───▼─┬─FILE───┬──'container string'──number-of-pages─┴──)─┘
                           └─DEVICE─┘

►──┬───────────────────────────────┬──┬────────────────────────────────┬──────►
   └─EXETENTSIZE──number-of-pages─┘  └─PREFETCHSIZE──number-of-pages─┘

►──┬────────────────────────────────────┬──┬────────────────────────────────────────┬──────┤
   └─OVERHEAD──number-of-milliseconds─┘  └─TRANSFERRATE──number-of-milliseconds─┘
```

The description of the command options for the `create database` command are the following:

ON Specifies a path or drive on which to create the database. The configuration file and the log files will be created in sub-directories of the specified path. For AIX, will use the home directory of the instance owner as default.

ALIAS Gives an alias name to the database. If not specified, it will use the same name as the database-name.

USING CODESET Specifies the code set to be used for data.

COLLATE USING Determines the type of collating sequence used. If SYSTEM, the collating sequence is based on the current country code. This is the default. If IDENTITY, strings will be compared byte by byte.

NUMSEGS Specifies the number of segment directories that will be used to store DAT, IDX and LF files. Only applies if any of the tablespaces are SMS tablespaces. Its default value is one.

DFT_EXTENT_SZ It is an integer that defines the size of the extent in 4KB pages. If not specified, will use 32 4 KB pages. This will be the default value when creating tablespaces unless the EXTENTSIZE parameter is used in the `create tablespace` command.

CATALOG TABLESPACE
Defines the catalog tablespace. If not specified, SYSCATSPACE will be created as a SMS tablespace.

USER TABLESPACE Defines the initial user tablespace, USERSPACE1. If not specified, it will be created as a SMS tablespace.

TEMPORARY TABLESPACE
Defines the initial temporary tablespace, TEMPSPACE1. If not specified, it will be created as an SMS tablespace.

See 4.5.7, "Creating Tablespaces" on page 72 for a description of the tablespace definition options.

4.5.2.2 The DFT_EXTENT_SZ Parameter

One of the new database parameters in DB2 Version 2 is DFT_EXTENT_SZ. It is set when you issue the `create database` command. It determines how many pages are written to a container before writing to another container. This is the extent size for tablespaces, both DMS and SMS. The default size for DFT_EXTENT_SZ is 32 4K pages. If you do not alter this value, all of your tablespaces within that database will have this default value. The range of values for DFT_EXTENT_SZ is between 2 and 256 pages.

You may still change the number of pages written before writing to another container (extent size) at the tablespace level. This can be done when creating the tablespace with the statement option EXTENTSIZE. Care should be taken to determine the correct value for EXTENTSIZE, as once it is set for a tablespace, it determines the extent size for a tablespace and cannot be altered. The extent size can have an impact on space utilization and performance.

4.5.2.3 The NUMSEGS Parameter

NUMSEGS and SEGPAGES were two parameters that were set with the `create database` command in DB2/6000 Version 1. They controlled the segmentation of databases in Version 1:

- NUMSEGS was the number of segment directories over which the database was split.
- SEGPAGES was the number of pages (4K) in a segment

SEGPAGES no longer exists in Version 2. NUMSEGS is kept mainly for backwards compatibility with Version 1. However, it does have significance when dealing with SMS tablespaces.

Because of restrictions in AIX Version 3.2.5, the maximum size of any one file or filesystem was 2 GB. The way DB2/6000 stored its data in one filesystem meant that without spreading the physical data over different filesystems, the maximum size of a database in Version 1 would have been restricted to 2 GB. So, to exist with AIX 3.2.5, DB2 allowed the database to be split across many filesystems such that the maximum size of the database was 2 GB x the number of file systems allocated to the database.

NUMSEGS in Version 2 is similar in concept. If you select SMS tablespaces, you are selecting NUMSEGS. That is the number of directories allocated to an SMS tablespace and is determined by the database parameter NUMSEGS in Version 2. Consider the example of selecting all the default tablespaces at `create database` time. (See Figure 17 on page 53.) There are 3 default SMS tablespaces created. Each one of these is a segment directory. NUMSEGS refers to the number of segment directories that the default SMS tablespace (USERSPACE1) will have. The default value for NUMSEGS is one. That means that the total size of USERSPACE1 if allowed to be in one segment directory is determined by the maximum size of any one filesystem in AIX. This is dependent on the level of AIX that you are using with DB2 Version 2. If using AIX 3.2.5 or lower, the maximum filesystem size is still 2 GB. If using AIX 4.1 or higher, the maximum filesystem size is 64 GB.

Alternatively, you may still use SMS tablespaces, but select a higher value for NUMSEGS. You may also create other SMS tablespaces with explicit directory paths, thus exceeding defaults. However, NUMSEGS is still set at the database level when issuing the `create database` command.

4.5.2.4 Default Database Files and Directories

When the `create database` command is issued, a number of database files and objects are created by default. The default database files and directories are placed in the database directory. The exact location of the database directory may be altered at database creation, but the name of the directory is assigned by the database manager. The first database directory will be SQL0001. The next database directory is SQL0002 and so on. This is independent of the type of tablespace that is created. Figure 20 shows the default objects associated with a database.

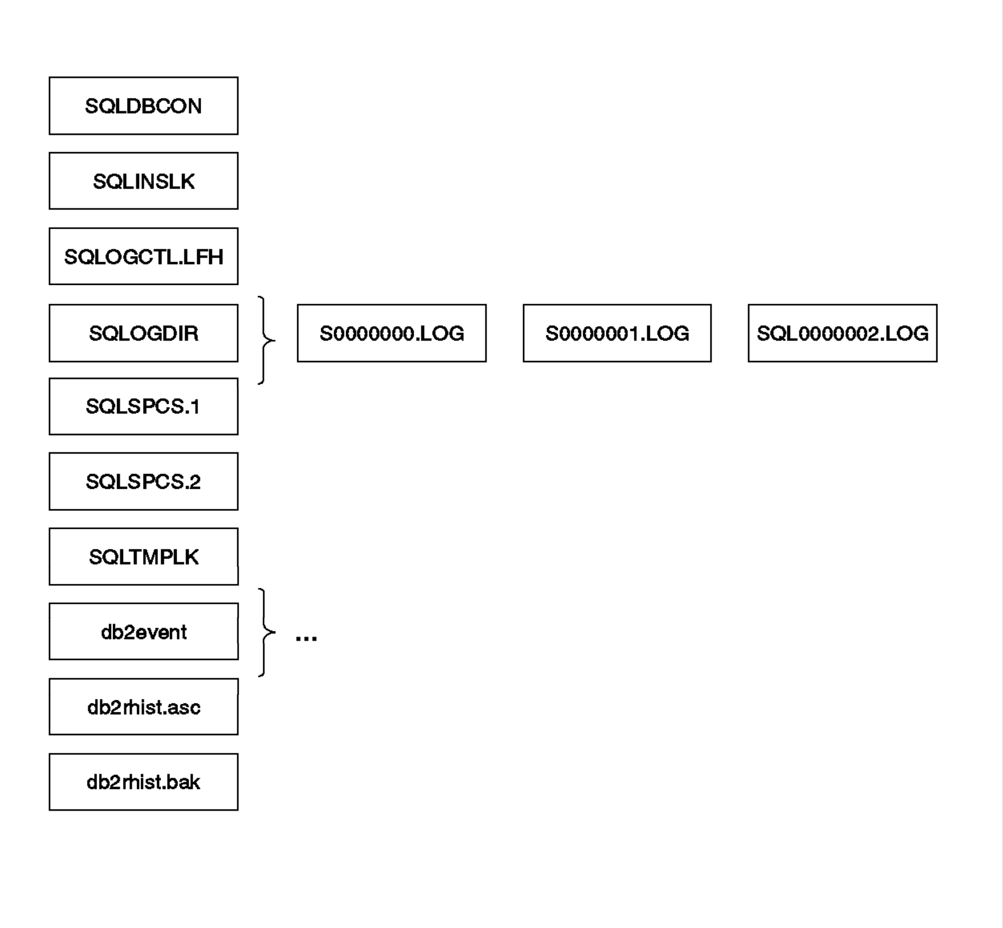


Figure 20. Default DB2 Database Objects

The following files and directories are created with every DB2 Version 2 database:

SQLDBCON File that stores the individual database parameters

SQLINSLK File that is used to ensure that a database is only used by one instance of the database manager

SQLTMPLK File that is used to ensure that a database is only used by one instance of the database manager

SQLOGCTL.LFH File that tracks and controls all the database logs

SQLOGDIR Directory that contains the three default logs. Circular logging is the default logging type. This parameter may be changed in the database configuration after creation. For more information on logging, see Chapter 6, “Logging” on page 131.

SQLSPCS.1 File that contains the definition and current state of all the tablespaces in the database

SQLSPCS.2 File that is a copy of SQLSPCS.1, used as a backup should SQLSPCS.1 fail

db2event Default directory that can be used for output files associated with an event monitor

db2rhist.asc File that is used for history of all backups and load operations

db2rhist.bak File that is used with db2rhist.asc. db2rhist.asc and db2rhist.bak are ASCII files that are merged together and viewed as one entity by the user. db2rhist.bak is called a shadowed history file.

4.5.3 Sizing of Tables and Tablespaces

When calculating the size of tables, the row size and structure of the table can be determined. However, other factors such as overhead for disk fragmentation, free space and variable length columns make it difficult to determine exact size.

An estimate of the size of user tables data can be obtained by:

```
(average_row_size + 8) * number_of_rows * 1.5
```

The calculations for the table row sizes and number of rows per table for the eight tables in our example for the dss database (see Figure 19 on page 59) are as follows:

Table 8 (Page 1 of 2). Sizes of Rows - dss Database

Table Name	Regular Data	BLOB	CLOB	Indexes	Number of Rows
ORDERS	344 Bytes	none	7 KBytes	46 bytes	150000
OPTIONS1	132 Bytes	none	none	32 bytes	750
OPTIONS2	132 Bytes	none	none	32 bytes	750

Table 8 (Page 2 of 2). Sizes of Rows - dss Database

Table Name	Regular Data	BLOB	CLOB	Indexes	Number of Rows
CUSTOMERS	186 Bytes	none	none	51 bytes	16000
LOCATIONS	144 Bytes	none	16 KB	12 bytes	3000
SITEMAPS	52 Bytes	1.5 MB	none	14 bytes	200
EMPLOYEE	144 Bytes	none	none	28 bytes	400
EMPDEPT	64 Bytes	none	none	12 bytes	64

User tables are stored using 4 KB pages. These 4096 bytes will limit the length of the row as a row cannot span multiple pages. Due to overhead and other factors, the actual size of rows is limited to 4005 bytes.

The pages do not contain Long Field Data or Large Object Data (LOBs). For columns containing these data types, only a descriptor is held in the table. Descriptor lengths are referenced in the following table:

Table 9. Size of Descriptors for Long Field Data and LOBs

Data Type	Bytes
LONG VARCHAR	20
LONG VARGRAPHIC	20
BLOB/CLOB/DBCLOB	72 to 280

The descriptor for BLOBs and CLOBs has a size between 72 and 280 bytes. The exact size is a function of the size of the object, whether the compact option is used or not and allowing for any fragmentation within the lob space. A recommendation for sizing is to either assume the worst case, or at take an average of the above information.

Long Field Data, as LONG VARCHAR or LONG VARGRAPHIC, are stored in a separate object. This object is divided in areas of 32 KB. Areas are broken into segments. The size of these segments is a multiple of 512 bytes. Each LONG VARCHAR/VARGRAPHIC value must fit into a single segment. Unused space in the Long Field Data object depends on the size of the data and on the consistency of the sizes of each entry. Unused space can add up to 50% of the size of Long Field Data.

Large Object Data are stored in two separate table objects: LOB Allocation Objects and LOB Data Objects.

- LOB Allocation Objects maintain information of allocation and free space for LOB Data Objects. This information is stored in 4 KB pages. The number of pages used depends on the space allocated for LOB Data Objects. Use one 4 KB page for every 64 GB of LOB Data Objects and use one 4 KB page for every increment of 8 MB of LOB Data.

- The size of LOB data objects is a multiple of 1024 bytes. To reduce the amount of redundant space, the COMPACT parameter on the `create table` and `alter table` can be used. This option reduces the amount of redundant space, allowing the LOB data to be placed into separate segments.

An estimate of the space needed for each index can be obtained by:

```
(average_index_key_size + 8) * number_of_rows * 2
```

If the index includes columns with different data types you should not use the maximum declared size of VARCHAR and VARGRAPHIC columns. Use the average size instead.

4.5.3.1 Minimum Space Requirements for Container in DMS Tablespace

The minimum space required for a DMS tablespace is determined by the extent size. By default, the extent size is 32 4 KB pages. Three extents are reserved for overhead and two more are the minimum required to store any user table data. Note that two extents are the minimum for each table in a DMS tablespace.

If the tablespace will contain indexes, Long Field Data or LOBs, two additional extents are required for each type.

The following table lists the minimum size for DMS tablespaces:

Table 10. Minimum Size of Tablespaces - DMS

Tablespace	**Size (Number of extents)**	**Size if using Default Extent Size (32 4 KB pages)**
Regular (only data)	5	160 4 KB pages
Regular (only indexes)	5	160 4 KB pages
Regular (data and indexes)	7	224 4 KB pages
Regular (data and LOBs)	7	224 4 KB pages
Regular (data, indexes and LOBs)	9	288 4 KB pages
Long	5	160 4 KB pages

The following table lists the maximum size for DMS tablespaces:

Table 11 (Page 1 of 2). Maximum size of Tablespaces - DMS

Tablespace	**Size**
Regular	64 GB

Table 11 (Page 2 of 2). Maximum size of Tablespaces - DMS

Tablespace	Size
Long	2 TB

4.5.4 Sizing Example for DMS Tablespace

To estimate the sizes of tables and tablespaces in our dss database example, we set the size of the extent to 8 4 KB pages for tablespaces ts01, ts02 and index01, and the size of the extent to 32 4 KB pages for the tablespace lobs01. Extent size is discussed in 4.5.6.1, "Extent Size in Tablespaces" on page 71.

The estimation of the size required by the regular table objects of our database example is the following (Extent size 8 4 KB pages):

Table 12 (Page 1 of 2). Sizing Regular Table Objects - Extent Size 8 4 KB Pages

Table Name	Regular Data	BLOB descriptor	CLOB descriptor	Number of Rows	Total	Number of extents
ORDERS	344 Bytes	0	72 Bytes	150000	((344+72) + 8) * 150000 * 1.5 = 95400 KB	2982
OPTIONS1	132 Bytes	0	0	750	(132 + 8) * 750 * 1.5 = 157.5 KB	5
OPTIONS2	132 Bytes	0	0	750	(132 + 8) * 750 * 1.5 = 157.5 KB	5
CUSTOMERS	186 Bytes	0	0	16000	(186 + 8) * 16000 * 1.5 = 4656 KB	146
LOCATIONS	144 Bytes	0	72 Bytes	3000	((144+72) + 8) * 3000 * 1.5 = 1008 KB	32
SITEMAPS	52 Bytes	72 Bytes	0	200	((52+72) + 8) * 200 * 1.5 = 39.6 KB	2
EMPLOYEE	144 Bytes	0	0	400	(144 + 8) * 400 * 1.5 = 91.2 KB	3

Table 12 (Page 2 of 2). Sizing Regular Table Objects - Extent Size 8 4 KB Pages

Table Name	Regular Data	BLOB descriptor	CLOB descriptor	Number of Rows	Total	Number of extents
EMPDEPT	64 Bytes	0	0	64	(64 + 8) * 64 * 1.5 = 7 KB	2 **1**

Note that even if the regular table object of the EMPDEPT table requires 7 KB, two extents **1** of 8 4KB pages are allocated to this object, as two extents is the minimum required for each table object.

4.5.4.1 Sizing Example for Indexes

The estimation of the size required by the index table objects of our database example is the following (Extent size 8 4 KB pages):

Table 13. Sizing Index Table Objects - Extent Size 8 4 KB Pages

Table Name	Index Size	Number of Rows	Total	Number of Extents
ORDERS	46 Bytes	150000	(46 + 8) * 150000 * 2 = 16200 KB	507
OPTIONS1	32 Bytes	750	(32 + 8) * 750 * 2 = 60 KB	2
OPTIONS2	32 Bytes	750	(32 + 8) * 750 * 2 = 60 KB	2
CUSTOMERS	51 Bytes	16000	(51 + 8) * 16000 * 2 = 1888 KB	59
LOCATIONS	12 Bytes	3000	(12 + 8) * 3000 * 2 = 120 KB	4
SITEMAPS	14 Bytes	200	(14 + 8) * 200 * 2 = 8.8 KB	2
EMPLOYEE	28 bytes	400	(28 + 8) * 400 * 2 = 28.8 KB	2
EMPDEPT	12 Bytes	64	(12 + 8) * 64 * 2 = 2.6 KB	2

4.5.4.2 Sizing Example for LOBs

The estimation of the size required by the LOB table objects of our database example is the following (Extent size 32 4 KB pages):

Table 14. Sizing LOB Table Objects - Extent Size 32 4 KB Pages

Table Name	LOB Data	LOB Segment	Number of Rows	Total	Number of Extents
ORDERS	7 KBytes	8 KBytes	150000	1200000 KB	9375
LOCATIONS	16 KBytes	16 KBytes	3000	48000 KB	375
SITEMAPS	1.5 MB	2048 KB	200	400000 KB	3125

Notice that LOB data has to fit into segments which are a multiple of 1024 bytes (1KB, 2KB, 8KB, 16 KB, and so on). If the COMPACT option is specified when the tables that contain LOB objects are created, an important amount of space could be saved, as each entry would use the most appropriate segment. A BLOB entry whose size is 900 KB, would use a 1 MB segment. For our database example, we assume that the COMPACT option is not specified and thus, all entries of the same table will use segments of the same size.

Allocation and free space information is stored in 4 KB allocation pages separated from the actual data. The number of these 4 KB pages depends on the amount of data, including unused space, allocated for the large object data. The overhead is calculated as follows: one 4 KB page for every 64 GB plus one 4 KB pages for every 8 MB.

If the character data is less than 4 KB in length, use data type CHAR, GRAPHIC, VARCHAR or VARGRAPHIC instead of BLOB, CLOB or DBCLOB.

4.5.4.3 Sizing Example - Summary

The space needed for a DMS tablespace will be obtained by adding together the space required for each table or table object that the tablespace will contain. Since data is written to containers in terms of extents, you should calculate the number of extents required for each table. To the number obtained you will need to add three extents that are required for overhead.

```
 sum(Number_of_extents_table_object) + 3
```

The number of extents for the DMS tablespaces of our database example is obtained by adding together the extents of the table objects that are going to be stored in the tablespace.

Table 15. Sizing DMS Tablespaces Example

Tablespace	Extents - Table Objects	Extents - Overhead	Total Extents	Extent Size
ts01	2982 + 5 + 5	3	2995	8 Pages of 4 KB
ts02	144 + 32 + 2 + 3 + 2	3	188	8 Pages of 4 KB
index01	507 + 2 + 2 + 59 + 4 + 2 + 2 + 2	3	583	8 Pages of 4 KB
lobs01	9375 + 375 + 3125	3	12878	32 Pages of 4 KB

4.5.5 Determining Containers

The next task is to choose the containers that will belong to each tablespace and to determine the size of each container. The number of containers that you will use and the placement of each container is highly dependent on your physical environment.

When choosing containers, you must consider on which physical drives will these containers be placed. In AIX, this will depend on the disks specified when the logical volumes were created, or the file system created, if any, when using files as containers. For OS/2, it will depend on the partitions or hard disks available.

The number of containers should be selected based on the number of physical disks associated with the database server. Consideration should also be given to the type of database operation being performed on database objects. For example, if performing a join on two separate tables, it would be advisable to place those tables, if possible, on separate disks. With the addition of the concept of tablespaces in Version 2, these tables may be in separate tablespaces.

Once the tablespace is created, it is not possible to change the type of tablespace, either DMS or SMS. Also the extent size may not be changed after creating the tablespace. However, there is an option that allows you to change the container definition when performing a restore operation. For more information, see 8.3.6, "Restoring a Database" on page 200.

For our database example we have four drives available. We decided to create containers for the ts01 and ts02 tablespaces on two drives. We will create one container for our index tablespace (index01), on the third drive. lobs01 tablespace will use one container on the fourth drive. By doing this, we will place LOBs, indexes and data on different drives. Using two different drives for our regular data will enable parallel I/0 operations. Containers for our database are shown in the following figure:

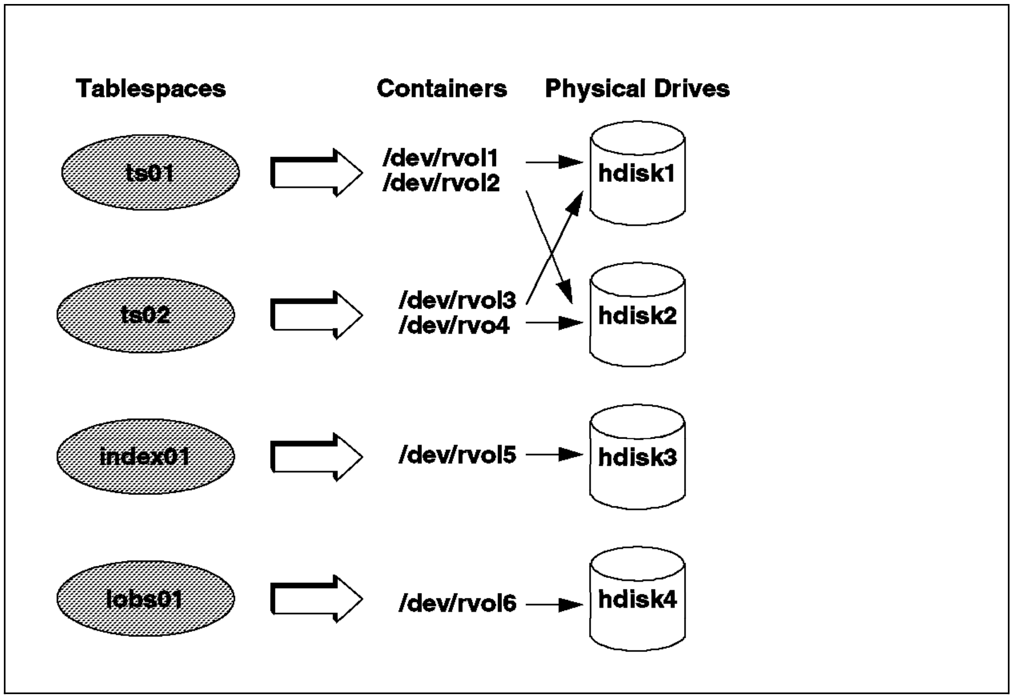


Figure 21. Defining Containers for User Tablespaces - AIX and DMS

If possible, make the containers the physical size in the operating system.

4.5.6 Writing to Containers

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

Figure 22 shows three containers, Container 0, Container 1 and Container 2 defined to Tablespace 4.

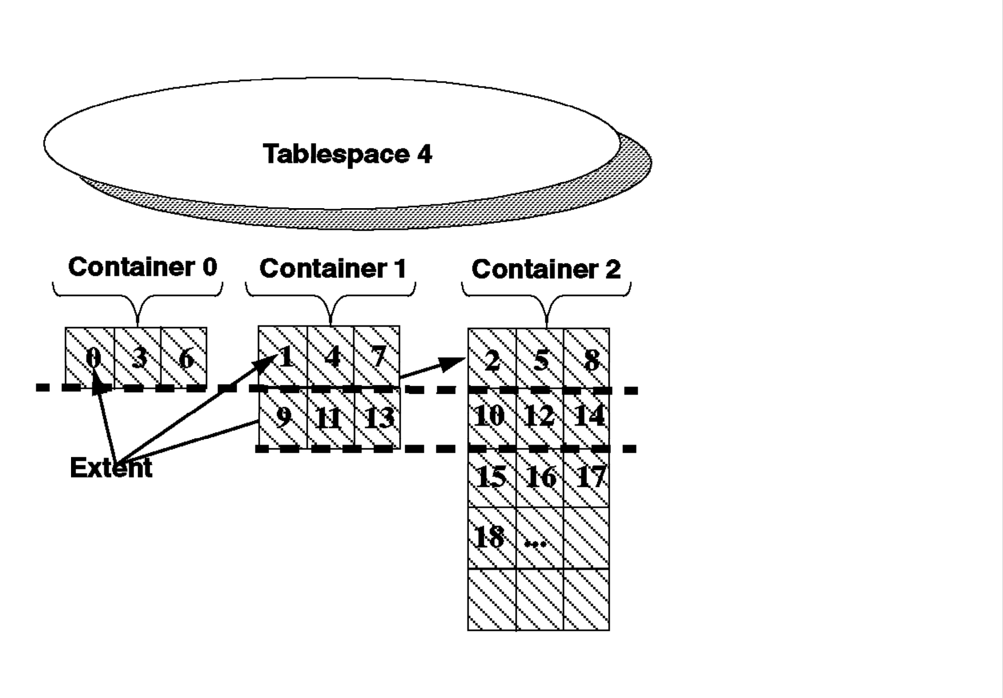


Figure 22. Writing to Containers

Tablespace 4 has three containers assigned to it, each with different sizes. However, the extent size for Tablespace 4 is the same for each container.

In this example, once the database manager has written to all the containers allocated to Tablespace 4, it will write back to the one it started writing to. This round-robin process of writing to the containers is designed to balance the load. In Figure 22, the database manager will write to the first extent, Extent 0 on Container 0 in Tablespace 4. Then writing will be in the second extent, Extent 1 in Container 1 in Tablespace 4. The third extent will be written to Extent 2 in Container 2 in Tablespace 3 and so on.

When the smallest container, Container 0, fills up, the database manager will continue to write to the other two containers. Writing will also be performed in a round-robin discipline, alternating containers. When containers 0 and 1 are filled, the database manager will have to do all the writing to Container 2. This is only true if the container is assigned to a DMS tablespace. If the container is assigned to an SMS tablespace, and the database manager wants to write to container 0 and is not able to, an error will be returned. When the containers no longer permit write access, the database manager will not be able to add any new pages. However, read or update access is still allowed.

4.5.6.1 Extent Size in Tablespaces

By default, the extent size to be used when a tablespace is created is 32 4 KB pages. You can change this default value at database level using the `update database` command, if the tablespace has not been created. Once, the extent size is set for the tablespace, it cannot be changed. You can also use a specific extent size for each tablespace. This can be accomplished when the tablespace is created using the EXTENTSIZE option. The range of EXTENTSIZE is from 2 4 KB pages to 256 4 KB pages.

A smaller extent size than the default can be chosen if your tablespace will not store large table objects. A smaller extent size will save space, but will require more frequent allocation of extents. A extent size of 8 or 16 4 KB pages is usually chosen for regular tablespaces. For long tablespaces, and to avoid the overhead derived from allocating a large number of extents each time a new entry is made, a larger extent size is recommended. However, if LOBs are smaller (less than 100K), the default size of 32 for extent size will suffice.

If the PREFETCHSIZE option is not specified when the tablespace is created, no prefetching will take place. Prefetching is also discussed in 4.7, "Performance Considerations" on page 82. Prefetching will create separate I/O processes (AIX) or threads (OS/2) that will perform the reads of large amounts of data. Prefetching will reduce the I/O activity, and will enable parallel I/O if the PREFETCHSIZE is a multiple of the EXTENTSIZE and the extents being prefetched are on separate disks.

4.5.6.2 Sizing Containers for a DMS Tablespace

To determine the size of each container used by a DMS tablespace, you will use the size of the tablespace as your starting point. You should distribute the number of extents required by the tablespace between the containers, and then add one 4 KB page to each container for overhead.

For our dss database example, we will estimate the size of the containers as being the size of the logical volume, as space available in the logical volume that has not been assigned to a container is wasted.

Table 16. Sizing Containers

Tablespace	Container	Size of the Container (in number of extents)	Extent Size	AIX 4MB Partitions necessary to accommodate the container	Rounded Size of Container (in 4 KB pages)
ts01	/dev/rvol1	1498	8 pages of 4 KB	12	12000
ts01	/dev/rvol2	1498	8 pages of 4 KB	12	12000
ts02	/dev/rvol3	94	8 pages of 4 KB	1	1000
ts02	/dev/rvol4	94	8 pages of 4 KB	1	1000
index01	/dev/rvol5	583	8 pages of 4 KB	5	5000
lobs01	/dev/rvol6	12878	32 pages of 4 KB	413	413000

You can now proceed to create all your user tablespaces and containers. For AIX, ask your system administrator to create the logical volumes you will be using as containers. You will have to provide the size, name and physical drive to use for each raw logical volume. If you want to assign some physical disks to be used only for your database, you may want to create a volume group with these drives. The additional tasks to be performed when using logical volumes as containers can be summarized as follows:

1. If you plan to assign a set of hard disks for the exclusive use of DB2, create a volume group with these disks. By doing so, you will not share these drives with other AIX logical volumes.
2. Create as many raw logical volumes as containers. Place them on the physical drive of your choice. Give them the appropriate number of partitions. Be careful, as space assigned to the logical volume and not given to the container will not be usable.
3. Change owner and group of the devices you have just created to the instance owner and group. This will grant DB2 Read/Write access to the device.

For our dss database example, we will provide the following information to the system administrator:

Table 17. Logical Volumes

Logical Volume Name	**Type**	**Physical disk**	**Number of 4MB partitions**
vol1	raw	hdisk1	12
vol2	raw	hdisk2	12
vol3	raw	hdisk1	1
vol4	raw	hdisk2	1
vol5	raw	hdisk3	5
vol6	raw	hdisk4	413

4.5.7 Creating Tablespaces

The `create tablespace` statement is new in Version 2. It will create a new tablespace within the database to which you are connected. It assigns containers to the tablespace and makes entries into the system catalogs about the tablespace attributes. To execute the `create tablespace` statment, you must have SYSADM or SYSCTRL authority.

The syntax of the `create tablespace` statement is as follows:

```
►►──CREATE──┬───────────┬──TABLESPACE──table-space-name──MANAGED BY──────────────────►
            ├─REGULAR───┤
            ├─LONG──────┤
            └─TEMPORARY─┘

                       ┌─,──────────────────┐
►──┬─SYSTEM USING──(───▼─'container string'─┴──)─────────────────────────────┬───────►
   │                     ┌─,────────────────────────────────────────────────┐ │
   └─DATABASE USING──(───▼─┬─FILE───┬──'container string'──number-of-pages─┴──)─┘
                           └─DEVICE─┘

►──┬──────────────────────────────┬──┬────────────────────────────────┬──────────────►
   └─EXETENTSIZE──number-of-pages─┘  └─PREFETCHSIZE──number-of-pages─┘

►──┬──────────────────────────────────┬──┬──────────────────────────────────────┬───►◄
   └─OVERHEAD──number-of-milliseconds─┘  └─TRANSFERRATE──number-of-milliseconds─┘
```

The description of the command options follows.

REGULAR This is the default value. Regular tablespaces store all data except for temporary tables.

LONG Stores long or LOB table objects. Long tablespaces must be DMS tablespaces.

TEMPORARY Stores temporary tables. One temporary tablespace is created when the database is created.

MANAGED BY SYSTEM
Defines this tablespace as an SMS tablespace.

USING (′container string′)
Identifies the directory used as container.

MANAGED BY DATABASE
Defines this tablespace as a DMS tablespace.

USING (FILE/DEVICE ′container string′ number-of-pages, ...)
Identifies the containers that will belong to this tablespace and its size. Size is measured in 4 KB pages.

EXTENTSIZE number-of-pages
Defines the size of the extent in 4 KB pages. If not specified, the extent size defined when the database was created will be used. The database parameter for the extent size is DFT_EXTENT_SZ.

PREFETCHSIZE number-of-pages
Specifies the number of “read-ahead” pages when reading from the tablespace. The default is 0, which means that no prefetching will be done.

OVERHEAD number-of-milliseconds
This value is only used to determine the cost of I/O during query optimization. Its default value is 24.1, and represents the I/O controller overhead and disk and latency time.

TRANSFERRATE number-of-milliseconds
This value is only used to determine the cost of I/O during query optimization. Its default value is 0.9, and represents

the time to read one 4 KB page from this tablespace into memory.

4.5.7.1 Creating Tablespace Example

For our dss database, we will use a extent size of 8 4 KB pages for regular tablespaces and 32 pages of 4 KB as our prefetch size.

The following commands will create our four tablespaces. We have placed the commands in a file, whose contents is as follows:

```
create tablespace ts01 managed by database using
(device '/dev/rvol1' 12000, device '/dev/rvol2' 12000)
extentsize 8 prefetchsize 16;

 create tablespace ts02 managed by database using
 (device '/dev/rvol3' 1000, device '/dev/rvol4' 1000)
 extentsize 8 prefetchsize 16;

 create tablespace index01 managed by database using
 (device '/dev/rvol5' 5000) extentsize 8 prefetchsize 32;

 create long tablespace lobs01 managed by database
 using (device '/dev/rvol6' 413000) extentsize32;
```

4.5.8 Creating Tables and Indexes

When you create a table, you must specify the table name and the name in attributes of its columns.

Default Placement of Tables in Tablespaces

Unless specified otherwise, the table will be created in the first user tablespace you created for your database, and indexes and LOBs will not be placed in separate tablespaces. The default placement for tables is the first user-created tablespace. If you have not created a tablespace, the default is USERSPACE1.

Three new options have been added to the create table command that allows you to specify in which tablespace you want to place regular data, indexes and long columns. The syntax with the new options is as follows:

```
►►──CREATE TABLE──table-name──( ... table definition ...)──┬──────────────────────┬──────────►
                                                           └─IN──tablespace-name─┘

►──┬──────────────────────────┬──┬─────────────────────────┬───────────────────────────►◄
   └─INDEX IN──tablespace-name─┘  └─LONG IN──tablespace-name─┘
```

The parameters are discussed further:

IN If this option is not specified, the table will be created in the first tablespace created by the user that is issuing the create table command. If the user had not created any tablespace, the table will be created in the tablespace USERSPACE1.

This option identifies the tablespace in which the table will be created. This tablespace must exist already, and must be a REGULAR tablespace. If no other tablespace is specified in the INDEX IN or LONG IN options, all table objects will be stored in this tablespace.

INDEX IN Specifies the tablespace in which any indexes on the table will be created. Only applies to DMS user tablespaces, this tablespace must exist already, and must be a REGULAR tablespace.

LONG IN Identifies the tablespace in which the Long Data or LOB objects are stored. Only applies to user DMS user tablespaces, this tablespace must exist already, and must be a LONG tablespace.

The commands to create our eight tables of our database example will be placed in a file similar to the following:

```
create table orders ( ...... ) in ts01 index in index01 long in lobs01;
create table options1 ( .... ) in ts01 index in index01;
create table options2 ( .... ) in ts01 index in index01;
create table customers ( ... ) in ts02 index in index01;
create table locations ( ... ) in ts02 index in index01 long in lobs01;
create table sitemaps ( .... ) in ts02 index in index01 long in lobs01;
create table employee ( .... ) in ts02 index in index01;
create table empdept ( ..... ) in ts02 index in index01;
```

4.5.9 System Catalog Changes

Information about tables and tablespaces is kept in the system catalog. The DBA will use DB2 commands to list, alter or create tablespaces and tables, but this information can also be obtained querying catalog tables. DB2 commands used to manage tables and tablespaces are described in 4.6, "Managing Tablespaces" on page 77.

There are three catalog views directly related to tablespaces, tables and indexes:

- SYSCAT.TABLESPACES
- SYSCAT.TABLES
- SYSCAT.INDEXES

See the Appendix D of the *Database 2 SQL Reference for Common Servers Version 2*, S20H-4665, for a detailed list of all the system catalog views.

SYSCAT.TABLESPACES contains a row for each tablespace. Each row maintains information about the name of the tablespace, the tablespace id, tablespace type, type of data this tablespace stores, extent size and prefetch size.

```
$ db2 ″select tbspace,tbspacetype,tbspaceid,datatype,extentsize from \
syscat.tablespaces″

TBSPACE            TBSPACETYPE TBSPACEID   DATATYPE EXTENTSIZE
------------------ ----------- ----------- -------- -----------
SYSCATSPACE        S                     0 A                 32
TEMPSPACE1         S                     1 T                 32
USERSPACE1         S                     2 A                 32
TS01               D                     3 A                  8
TS02               D                     4 A                  8
INDEX01            D                     5 A                  8
LOBS01             D                     6 A                 32

  7 record(s) selected.

$
```

The output of the last command shows a total of seven tablespaces, the first three are SMS tablespaces and the last four are DMS tablespaces. The tablespace id is a unique number that identifies the tablespace in the database. Values for DATATYPE column show the type of data than can be stored in this tablespace; A for all types of permanent data and T for temporary tables. Notice that there are different extent sizes for the tablespaces.

SYSCAT.TABLES contains a row for each table, view or alias that is created. Related to tablespaces, each row mantains information about the name of the table, the tablespace where the table is placed, and names of the tablespaces where indexes and Long Object Data or LOBs for this table are placed.

```
$ db2 ″select tabname,tbspaceid,tbspace,index_tbspace,long_tbspace
from syscat.tables where tabschema=′DB2′″

TABNAME           TBSPACEID TBSPACE            INDEX_TBSPACE      LONG_TBSPACE
----------------- --------- ------------------ ------------------ -------------

ORDERS                    3 TS01               INDEX01            LOBS01
OPTIONS1                  3 TS01               INDEX01            -
OPTIONS2                  3 TS01               INDEX01            -
CUSTOMERS                 4 TS02               INDEX01            -
LOCATIONS                 4 TS02               INDEX01            LOBS01
SITEMAPS                  4 TS02               INDEX01            LOBS01
EMPLOYEE                  4 TS02               INDEX01            -
EMPDEPT                   4 TS02               INDEX01            -

  8 record(s) selected.

$
```

Eight tables using this schema (db2) exist in the database. The orders, options1 and options2 tables are placed in tablespace ts01. The customer, locations, sitemaps, employee and empdept tables are in ts02. All the indexes for the eight tables are in tablespace index01. All the lobs for tables containing lobs are in lobs01.

SYSCAT.INDEXES contains a row for each index that is defined for a table. Specifically, each row maintains information on what indexes are created on each table.

```
$ db2 select ″indname,tabname from syscat.indexes where tabschema=′DB2′″

INDNAME            TABNAME
------------------ ------------------
ORD_NO             ORDERS
OPT_1              OPTIONS1
OPT_2              OPTIONS2
CUST_NO            CUSTOMERS
LOC_ID             LOCATIONS
SITE_IT            SITEMAPS
EMP_NO             EMPLOYEE
DEPT_NO            EMPDEPT

  8 record(s) selected.

$
```

Eight indexes have been created for the eight tables. Note that in this output, no information is mantained about the tablespace being used by indexes.

4.6 Managing Tablespaces

Tablespace management includes the tasks of creating, deleting, modifying and monitoring tablespaces and containers. DB2 provides commands and utilities to perform these tasks. The commands available to the DBA are the following:

- list tablespaces

  The `list tablespaces` command lists all the tablespaces contained in the database. For each tablespace, it shows information about type, data type contained, state, size (only for DMS tablespaces), and extent and prefetch size.

- list tablespace containers

  The `list tablespace containers` command lists all the containers for a specific tablespace. Shows information about the name, type and size (only DMS tablespaces) of the containers of this tablespace.

- alter tablespace

  The `alter tablespace` enables the DBA to add containers to a DMS tablespace (not supported for SMS tablespaces), and modify the PREFETCHSIZE, OVERHEAD and TRANSFERRATE of a tablespace.

- drop

  The `drop` command deletes, among other table objects, an index, table, tablespace or view. Objects that are directly or indirectly dependent on that object are also deleted or marked inoperative.

The commands described below do not constitute a full listing of all commands related to tablespaces and containers. Also note that not all the syntax diagrams are complete. The purpose is to highlight new commands and options provided by DB2 Version 2. Refer to the *Database 2 SQL Reference for common servers Version 2*, S20H-4665, and the *Database 2 Command Reference for common servers Version 2*, S20H-4645, for a complete description of commands and command options.

Note that all these tasks can be accomplished using the Database Director. The Database Director is a graphical user interface that helps you perform the database administration function more easily. The Database Director is discussed in Chapter 8, “Data Access” on page 183

4.6.1 List Tablespaces

The syntax for the `list tablespaces` command is as follows:

```
►►──LIST TABLESPACES──┬─────────────┬──────────────────────────────────────────────►◄
                      └─SHOW DETAIL─┘
```

SHOW DETAIL If this option is not specified, this command will list all tablespaces for the current database, their name, type, contents and extent. If specified, it will also list information about the size, extent size and prefetchsize.

The output for the `list tablespaces show detail` command is similar to the following:

```
 Tablespace ID                        = 0
 Name                                 = SYSCATSPACE
 Type                                 = System managed space  1
 Contents                             = Any data
 State                                = 0x0000
   Detailed explanation:
     Normal
 Total pages                          = 754
 Useable pages                        = 754  2.
 Used pages                           = 754
 Free pages                           = Not applicable
 High water mark (pages)              = Not applicable
 Page size (bytes)                    = 4096
 Extent size (pages)                  = 32
 Prefetch size (pages)                = 32
 Number of containers                 = 1

 Tablespace ID                        = 2
 Name                                 = USERSPACE1
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x0000
 .
 .
 .

 Tablespace ID                        = 3
 Name                                 = TS01
 Type                                 = Database managed space
 Contents                             = Any data
 State                                = 0x0000  3
   Detailed explanation:
     Normal
 Total pages                          = 24000   4
 Useable pages                        = 3968    5
 Page size (bytes)                    = 4096
 Extent size (pages)                  = 32
 Prefetch size (pages)                = 32
 Number of containers                 = 2       6
```

Not that when a database is created, creating the system catalogs will require some storage. (1)

The state of the tablespace 3, is expressed in a hexadecimal number, being 0x0000 in a normal state. This hexadecimal number may be a combination of states. DB2 provides the db2tbst DB2 Version 2.1.1 includes more detailed information about the state of a tablespace in a more readable format. utility that converts this number into a readable format. Tablespaces states are discussed in more detail in 4.6.5, “States of Tablespaces” on page 81.

4 and 5 provides information about the total and useable pages of the tablespace. None of these figures represent the free space in the tablespace. The difference between total pages and useable pages is due to the fact that space in a container is only useable if it is a multiple of the extent size. This is shown in the list tablespace containers example.

6 shows that this tablespace is defined to use two containers.

You could obtain similar information using the Database Director. For more information about the Database Director, see Chapter 8, “Data Access” on page 183. The Database Director will also provide information about the amount and percentage of space being used.

4.6.2 List Tablespace Containers

The syntax for the list tablespace containers command is as follows:

```
►►──LIST TABLESPACE CONTAINERS FOR──tablespace-id──┬─────────────┬──────────────────────►◄
                                                   └─SHOW DETAIL─┘
```

The parameters of the command are described:

tablespace-id An integer that uniquely identifies a tablespace in the database. This value can be obtained from the list tablespaces command or querying the SYSCAT.TABLES table.

SHOW DETAIL If this option is not specified, the command will show the container id, name and type (file, disk or path) of every container used by the tablespace. If specified, this option will also show the total number of pages and the number of useable pages if the containers belong to a DMS tablespace.

The following screen shows the output of the list tablespace containers command issued against a DMS tablespace which is using two containers.

```
$ db2 ″list tablespace containers for 3 show detail″

            Tablespace Containers for Tablespace 4

 Container ID                        = 0
 Name                                = /dev/rvol1
 Type                                = Disk
 Total pages                         = 12000
 Useable pages                       = 11992
 Accessible                          = Yes
 Container ID                        = 1
 Name                                = /dev/rvol2
 Type                                = Disk
 Total pages                         = 12000
 Useable pages                       = 11992
 Accessible                          = Yes

$
```

Notice the difference between total pages and usable pages for these containers. The container ′/dev/rvol1′ has 8 pages unusable. One is reserved for the administrative overhead of the container. Depending on the extent size of your tablespace after overhead, the remaining pages are divided into allocations of extent size. Any other remaining amounts less than the extent size are wasted.

4.6.3 Alter Tablespace

The syntax for the alter tablespace command is as follows:

```
►►──ALTER TABLESPACE──tablespace-name──────────────────────────────────────►

    ┌──────────────────────────────────────────────────────────────────┐
    │           ┌─,──────────────────────────────────────────────┐     │
    ▼           ▼                                                │     │
►───┬─ADD──(──┬─FILE───┬──′container string′──number-of-pages──┴──)─┬┴──────────►◄
    │         └─DEVICE─┘                                            │
    ├─PREFETCHSIZE──number-of-pages─────────────────────────────────┤
    ├─OVERHEAD──number-of-milliseconds──────────────────────────────┤
    └─TRANSFERRATE──number-of-milliseconds──────────────────────────┘
```

The command paramters are listed as follows:

ADD Specifies the new container to be added to a DMS tablespace.

Many of these options have already been described in 4.5.7, "Creating Tablespaces" on page 72 when discussing the create tablespace statement. Note the extent size of a tablespace cannot be changed, and that containers can only be added to DMS tablespaces. Note also that containers cannot be modified.

When you add a container to a tablespace, the data placed in a tablespace will automatically be distributed among containers.

4.6.4 Drop Tablespace

The syntax for the drop tablespace command is as follows:

```
►►──DROP──┬─INDEX──index-name───────────┬──────────────────────────────────────────►◄
          ├─TABLE──table-name───────────┤
          ├─TABLESPACE──tablespace-name─┤
          ├─VIEW──view-name─────────────┤
          └─other objects ...───────────┘
```

When you drop a table, all indexes, primary keys, foreign keys and check contraints referencing the table are also dropped. Views and triggers referencing the table are marked inoperative, and packages depending on them will be invalidated.

You will not be able to drop a tablespace in any table stored in the tablespace which has one of its parts (indexes, or LOBs) stored in another tablespace. You will have to drop first the table and then proceed to drop the tablespace.

Dropping a tablespace will drop all tables, indexes, primary and foreign keys, and referential constraints of the tablespace. Views and triggers are made inoperative.

User created containers such as filesystems of logical volumes in AIX will not be deleted when a tablespace is dropped. To recover the space used by DMS containers, you have to delete them manually.

4.6.5 States of Tablespaces

The database manager maintains information about the states of tablespaces and will not allow access to tablespaces that are not in a normal state. Tablespace states are given in hexadecimal numbers. Sometimes a tablespace may have more than one state associated with it. This will result in a combined hexadecimal number. All of the states associated with tablespaces can be found in the sqlutil.h file located in the /usr/lpp/db2_02_01/include directory (AIX). To view the tablespace state, issue the list tablespacees command. DB2 Version 2.1.1 shows the tablespace state and gives the explanation of the state. Alternatively, you can use a utility called db2tbst with the hexadecimal number to evaluate the tablespace state. The db2tbst utility can be found in the sqllib/misc directory of the instance owner.

A tablespace is placed in a non-normal state during load, backup and recovery operations, or if placed in a quiesced condition via the quiesce tablespaces command Inconsistencies in data across tablespaces are avoided by restricting tablespace access. A listing of some of the possible tablespace states are as follows:

- Normal (0x0000)

  Access to the tablespace is allowed.

- Quiesce related states:

  A tablespace is in any of these states when a quiesce request is received by the database manager. Any of these states must be explicitly reset.

  – Quiesced share (0x0001)

  – Quiesced update (0x0002)

  – Quiesced exclusive (0x0004)

- Load related states:
  - Load Pending (0x0008)

    A tablespace is in a Load Pending state when a table is being loaded or when the LOAD utility fails. After correcting the problem that caused the failure, invoking the LOAD utility with the RESTART option will continue with the load process.
  - Delete Pending (0x0010)

    A tablespace is in a Delete Pending state when a table stored in the tablespace has been successfully loaded and the LOAD utility is processing the delete phase to delete rows with duplicate keys.
- Backup/Recovery states:
  - Backup Pending (0x0020)

    A tablespace will be in this state when a table contained in this tablespace has been successfully loaded using the LOAD utility. Access to the tablespace will not be allowed until a backup of the database or the tablespace is taken.
  - Restore Pending (0x0100)

    A tablespace will be in this state when you perform a LOAD with TERMINATE option. The tablespace will have to be recovered from from a backup. If the backup being restored is a database backup, the database will have to be rolled forward to the end of the logs.

Load and Backup/Recovery related states are covered in Chapter 5, “Data Movement” on page 91 and Chapter 7, “Backup and Restore” on page 151

4.7 Performance Considerations

Configuration parameters that affect performance can be defined at different levels:

- Database manager
- Database
- Tablespace
- Table

When a parameter can be defined at two different levels, the value specified at the higher level will be considered the default value. We will discuss several of the parameters that may affect performance.

4.7.1 Buffer Pool Size

Determines the number of 4 KB pages to be allocated on the machine where the database is located. This memory is allocated when the first application connects to the database and released when the last one disconnects. This pool is used to transfer data to and from disk.

If the pool is large enough, disk I/O activity will be reduced. As a rule of thumb, you can start with 50% of total memory. If the machine is a dedicated database server, you may assign a larger amount of machine memory to the buffer pool. Be default, this pool will have 1000 4 KB pages.

This is the most important parameter that will affect database performance and is defined at database level.

4.7.2 I/O Prefetch

I/O prefetch increases the performance of queries that retrieve a large amount of data. Prefetching allows CPU and I/O overlap reducing the query execution time.

- PREFETCHSIZE

  This parameter determines the amount of data that is “read-ahead.” The prefetchsize is determined when the tablespace is created. If no value is specified when the tablespace is created, it will take the default value in the database parameter, DFT_PREFETCH_SZ. The default size is 32 4 KB. It should be set to a multiple of the extent size, as this would allow the database manager to perform parallel I/O if the extents being prefetched are on different disk drives and multiple I/O servers are configured. This value can be modified using the `alter tablespace` command.

  For any prefetching to occur, the buffer pool must be set to 1100 4 KB pages or higher. The default values for the buffer pool are 1000 4 KB for AIX and 250 4 KB for OS/2. Both AIX and OS/2 must have a value greater than 1100 4 KB for prefetching to occur.

- NUM_IOSERVERS

  This parameter specifies the number of processes to be used to perform prefetch I/O and asyncronous I/O (as required by backup and restore utilities). If prefetching is being used, this number should be set to the number of physical disk drives + 2 (for backup and restore).

4.7.3 I/O Cleaners

The object of having I/O cleaners is to ensure that there are enough “clean” pages in the buffer pool when a transaction is started. In this way, the transaction will not have to wait until previously used pages are written out. I/O page cleaners are also discussed in 6.2.11, “NUM_IOCLEANERS - Asynchronous Page Cleaners” on page 137.

- NUM_IOCLEANERS

  Determines the maximum number of I/O cleaners processes that can be started. This processes are started when the ′Changed Pages Threshold′ is reached. It is defined at database level, and the default number is 1.

  In a query-only database, this parameter should be set to 0 as no pages will be need to be written to disk. In a transaction environment, the parameter should be set between 1 and the number of physical disks used by the database.

- Changed Pages Threshold (CHNGPS_THRESH)

  This threshold determines at which percentage of changed pages will the asynchronous page cleaners be started. Page cleaners will write all modified pages to disk and will become inactive until the threshold is again exceeded.

  This parameter is defined at database level, and its default value is 60%.

4.8 Customer Scenario for Data Placement

The steps that the telephone company will execute have been illustrated throughout this chapter. To summarize:

1. Decide on what databases will be migrated and how. Consider what tools might possibly assist in the migration. There must also be consideration for any changes from DB2/6000 Version 1 to DB2 V2 such as using an invalid schema.
2. Perform a logical design of tablespaces, deciding which tablespaces will be SMS, which will be DMS.

   Consider using SMS tablespaces for system catalogs and temporary space and DMS tablespaces for any user tablespaces.
3. Create any new databases
4. Drop USERSPACE1 before creating DMS tablespaces
5. Calculate space needed for regular tables
6. Calculate space needed for indexes
7. Calculate space needed for LOBs
8. Group tables into tablespaces and size tablespaces.
9. Group indexes into tablespaces and size tablespaces.
10. Group LOBs into tablespaces and size tablespaces.

    Determine buffer pool size, EXTENTSIZE, PREFETCHSIZE and number of I/O server and I/o cleaners.

    Before using the database, decide on the logging policy, either circular or log retain. Consider placing the log files on a separate disk from the database. Change the database parameter, NEWLOGPATH, before using the database.

4.8.1 Telephone Company

The phone company uses SMS tablespaces for the catalog and the temporary tablespace for their production environment. They created a filesystem mounted on the $HOME/$DB2INSTANCE/SQL00001 directory where the database′s files and directories are placed. By doing this, they can control the amount of free space left for these tablespaces. The space used or required by any other AIX application or database will have no influence on the space left for these two tablespaces. They assigned 100 MB to this filesystem, estimating that their catalog tablespace would require 25 MB and that the database manager could require up to 75 MB for its temporary tables. The phone company left the EXTENTSIZE and PREFETCHSIZE to their default values (32 pages of 4 KB) for these two tablespaces.

For the user tables, they are using DMS tablespaces. AIX logical volumes are used as containers. Their query environment is very well suited for parallel I/O, and they want to have the indexes in separate drives. They grouped their 52 tables according to their backup policy, their data placement requirements and how often their tables are modified. They classified their tables according the following criteria:

- Size
  - A table is considered Small if its size is under 1000 rows.

  - A Medium size table will have between 1000 and 100000 rows.
  - Large tables have more than 100000 rows.
- Daily modifications
  - A Low number of modifications is considered when less than 1000 rows are modified per day.
  - A Medium number of modifications is considered when the number of rows modified per day is between 1000 and 8000.
  - A table is considered Frequently modified when the number of modified rows exceeds 8000.

The following table show the number of tables that fall in each category:

Table 18. Phone Company - Grouping Tables

	Small Tables	**Medium Tables**	**Large Tables**
Low	23	2	None
Medium	None	9	None
Frequent	None	9	8

They grouped their tables in the following tablespaces:

1. The space01 tablespace would store all the tables that suffer a low level of modifications.
2. The space02 tablespace would store all the tables that suffer a medium number of modifications.
3. The space03 tablespace would store all the tables that suffer frequent modifications and are ′medium′ size tables. considered ′medium′.
4. They would create eight tablespaces, space04 to space11, one for each of the eight large tables that are frequently modified.

For indexes, the database administrator decided to create one tablespace for all the indexes .

The telephone company database would be stored using 12 tablespaces:

- Tablespaces space01 to space11 for table data.
- Tablespace index01 for indexes.

To estimate the size of the tablespaces, they used a extent size of 8 pages of 4 KB. After obtaining the storage space required by tables and indexes, they added the sizes of table objects that are stored in the same tablespace. They added the extents required for overhead, and rounded the total number of extents. They obtained obtain the following figures:

Table 19. Phone Company - Sizes of Tablespaces

Tablespace	Size in MB
space01	48
space02	384
space03	456
space04	1968
space05	288
space06	1104
space07	600
space08	192
space09	372
space10	168
space11	576
index01	516

The AIX logical volumes to be used as containers were determined using the following policies:

- Containers for indexes were placed in separate physical drives from the rest of user tablespaces.
- To enable parallel I/O operations, containers used by the same tablespace were placed in different disks.
- The number of containers for each tablespace was chosen depending on the size of the tablespace. The database administrator decided to use one container for small tablespaces, three containers for medium size tablespaces and six containers for large tablespaces.

The database manager placed the containers using twenty-four physical drives. The distribution the logical volumes used as containers over the available physical drives is shown in the following table:

Table 20 (Page 1 of 3). Phone Company - Containers

Tablespace	Size in MB	Containers	Size of Containers in MB	Physical Disk
space01	48	/dev/rcont10	48	hdisk4

Table 20 (Page 2 of 3). Phone Company - Containers

Tablespace	Size in MB	Containers	Size of Containers in MB	Physical Disk
space02	384	/dev/rcont20	128	hdisk4
		/dev/rcont21	128	hdisk8
		/dev/rcont22	128	hdisk12
space03	456	/dev/rcont30	152	hdisk16
		/dev/rcont31	152	hdisk20
		/dev/rcont32	152	hdisk24
space04	1968	/dev/rcont40	328	hdisk4
		/dev/rcont41	328	hdisk8
		/dev/rcont42	328	hdisk12
		/dev/rcont43	328	hdisk16
		/dev/rcont44	328	hdisk20
		/dev/rcont45	328	hdisk24
space05	288	/dev/rcont50	96	hdisk5
		/dev/rcont51	96	hdisk9
		/dev/rcont52	96	hdisk13
space06	1104	/dev/rcont60	184	hdisk5
		/dev/rcont61	184	hdisk9
		/dev/rcont62	184	hdisk13
		/dev/rcont63	184	hdisk17
		/dev/rcont65	184	hdisk21
		/dev/rcont66	184	hdisk25

Table 20 (Page 3 of 3). Phone Company - Containers

Tablespace	Size in MB	Containers	Size of Containers in MB	Physical Disk
space07	600	/dev/rcont70	200	hdisk17
		/dev/rcont71	200	hdisk21
		/dev/rcont72	200	hdisk25
space08	192	/dev/rcont80	64	hdisk6
		/dev/rcont81	64	hdisk10
		/dev/rcont82	64	hdisk14
space09	372	/dev/rcont90	124	hdisk18
		/dev/rcont91	124	hdisk22
		/dev/rcont92	124	hdisk26
space10	168	/dev/rcont100	56	hdisk6
		/dev/rcont101	56	hdisk10
		/dev/rcont102	56	hdisk14
space11	576	/dev/rcont110	192	hdisk7
		/dev/rcont111	192	hdisk11
		/dev/rcont112	192	hdisk15
index01	516	/dev/ridx10	172	hdisk19
		/dev/ridx11	172	hdisk23
		/dev/ridx12	172	hdisk27

Before creating the logical volumes, the AIX system administrator created a volume group, vgdb2. This volume group included twenty-four physical volumes, from hdisk4 to hdisk27. Separating the drives in a different volume group, isolated them from regular operating system activity. Unless intentionally done, no AIX filesystem is stored in the physical volumes of this volume group.

After creating the volume group, the AIX system administrator created the 40 raw logical volumes that are used as containers. Sizes of the logical volumes and their physical drives where they were created, were obtained from the previous table. After the logical volumes were created, the AIX system adminstrator changed the owner and group of these logical volumes to the owner and primary group of the instance.

The DB2 administrator created the user tablespaces. All the user tablespaces are DMS tablespaces. Containers and sizes of containers of the tablespaces were taken from the previous table. Note that the overhead required by the container (one 4 KB page) is irrelevant relative to the size of the logical volume, so they used the same size for the container and for the logical volume. EXTENTSIZE for all user tablespaces was set to 8 4 KB pages. To enable prefetching and parallel I/O operations, the PREFETCHSIZE of user tablespaces was set to 24 4 KB pages for tablespaces that use three containers and PREFETCHSIZE was set to 48 4 KB pages for tablespaces that use six logical volumes as containers. Both sizes are multiples of the extent size. Using these values, and if enough I/O servers are available, a R/W request coming from an application can be performed in parallel to all the containers of the tablespace.

Tables and indexes were created by the DB2 administrator. When each table and index was created, the tablespace to be used to store de table was determined.

The Buffer Pool Size was increased to improve the performance of the database. Since the processor is a dedicated database server, 50% of the system memory was given to this pool. The Buffer Pool Size was increased to 256 MB using the `update database configuration` command.

The number of I/O servers was increased to 26. These twenty-six I/O servers were chosen because the number of physical drives to be used by the database is twenty-four. Two more I/O servers were added so backup/restore processes can be executed at the same time.

I/O Cleaners and Changed Pages Threshold were left to their default value. No I/O cleaners should be needed, as the database is query-only, and no pages are to be written to disk. Since I/O cleaners are only started when needed, leaving this value (1 I/O cleaner) to its default does not affect the resources of the system.

Chapter 5. Data Movement

This chapter discusses how to move data into DB2 Version 2 databases. Specifically, we will discuss:

- The load utility
- The import/export utility

This chapter will focus on the load utility which is new in Version 2. Import/export was available in Version 1. This chapter discusses the differences in implementation of import/export in Version 2.

The load utility is intended for the initial load or an append of a table where large amounts of data are inserted. There are no restrictions on the data types used by the load utility. You may include LOBs (Large Binary Objects) and user-defined types (UDTs) as data that is going to be loaded. The load utility is faster than performing an import because load writes formatted pages directly into the database, while import performs SQL inserts. Also the load utility does not log each write of data during a load operation.

5.1 Overview of the Load Process

There are three phases of the load process:

1. Load, where the data is written into the table.
2. Build, where the indexes are created.
3. Delete, where the rows that caused a unique key violation are removed from the table.

Figure 23 on page 92 shows each phase of the load process and the actions which occur during these phases:

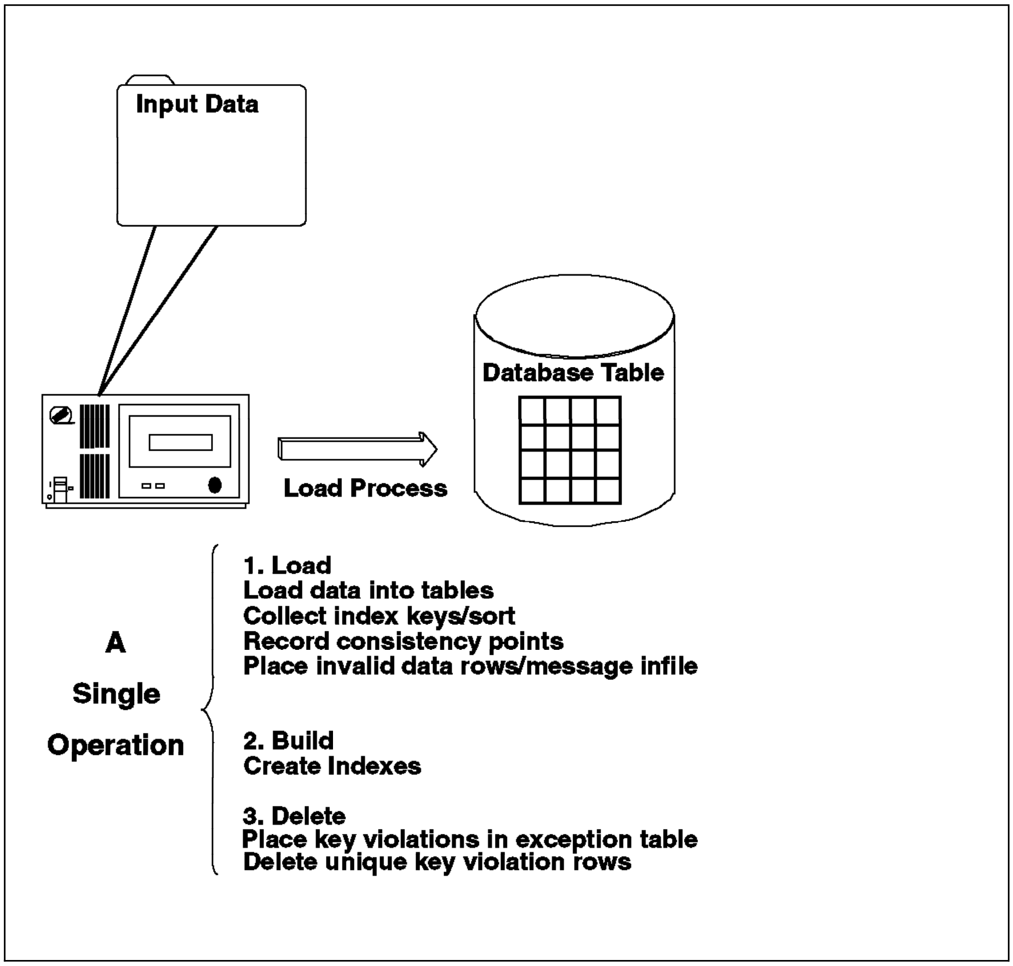

Input Data
Database Table
Load Process
A
Single
Operation
1. Load
Load data into tables
Collect index keys/sort
Record consistency points
Place invalid data rows/message infile
2. Build
Create Indexes
3. Delete
Place key violations in exception table
Delete unique key violation rows

Figure 23. Three Phases of Load

The three phases of the load utility can be further explained in the following way:

1. Load Phase

   During the load phase, data is loaded into the table, and index keys are collected. Save points or consistency points are established at intervals specified by you in the `load` command. Messages let you know how many input rows have been successfully loaded at the time of the save point.

   If a failure occurs, you should use the restartcount option set to the value indicated by the last load consistency/save point indicated in the messages file. If the failure occurs near the beginning of the load, you could consider restarting the load again from the beginning.

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. If a failure occurs, the build is restarted from the beginning of the build phase.

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 of the delete phase. Once the database indexes are rebuilt, information about the rows containing the invalid keys is stored in an exception table, if you have created one before the load began and identified it in the `load` command. Unique key violations are placed into the exception table, and messages on rejected rows are put into the message file. Finally, duplicate keys are deleted.

All phases of the load process are part of one operation which is completed only after all three phases complete successfully. The load utility will generate messages during the progress of each phase. Should a failure occur during one of the phases, then these messages can assist you in deciding on recovery actions.

5.2 Getting Ready for the Load Utility

There are a number of steps to consider before beginning the load operation. It is highly recommended that some care and thought be taken to get ready for the load as it can help and even in some cases prevent you having to restart the utility. These steps are as follows:

1. Know how and in what format data will be available to the load utility.
2. Sort the input data if necessary.
3. Create the target table and the exception table.
4. Calculate the storage required for the loading of data into the target and exception table.
5. Consider the location and estimated storage requirements for both the remote file and the temporary sort files in the sort directory which are generated during the load. The remote file and using directory are options of the `load`. (See 5.2.6, "The Remote File Option" on page 97 and 5.2.5, "The Using Directory Option" on page 97 for more details.)
6. If making a copy of the loaded data on disk, allow for the storage of that copy. This is discussed further in 5.2.7, "The Copy Yes Option" on page 98.
7. Place the `load` command in a file that can be edited and executed from the command line or the CLP.

These steps are described in more detail in the following sections.

5.2.1 Input Data File

Input data for a database load process is likely to come from a variety of sources: other DB2 databases (LAN or host), other databases, transactional systems, PC applications, and so on. You need to consider how to convert and sort, if necessary, this input data into a format that the `load` command can process.

The input data for a load process must be in one of three file formats: Integrated Exchange Format (IXF), delimited ASCII (DEL) or non-delimited ASCII (ASC).

PC/IXF This file format is a database manager adaptation of the Integrated Exchange Format and is the preferred method for exchange between database managers. The IXF architecture is designed specifically to enable exchange of relational database structure and data. You can export a data file using a Distributed Database Connection Services (DDCS) gateway from a host database to the DB2 Server. The advantage of this is that it eliminates the additional steps of creating a table using

DDL (Data Definition Language) and related indexes into a new or empty table.

In general, a PC/IXF file consists of an unbroken sequence of variable-length records. The file will have the following types of records in the order given:

- One header record of record type H
- One table record of record type T
- Multiple column descriptor records of record type C (one record for each column of data in the table)
- Multiple data records of record type D. Each row in the table is represented by one or more D records.

> **Note**
>
> If the host file contains packed fields, you will have to convert these fields before transferring the file to the DB2 common/server database. To perform this conversion, create a VIEW in DB2 for MVS for all the columns that you require; a VIEW automatically forms character fields out of the packed fields. From the VIEW, you can now EXPORT the required data as an IXF file.

DEL If you do not have a DDCS connection with a remote database, or you are transferring data from some other source, it is likely to be in delimited ASCII format. Delimited ASCII is used for exchanging files with a wide variety of industry applications, especially other database products. This is a commonly used way of storing data that separates column values with a special delimiting character. An example of a DEL file is:

```
″Smith, Bob″,4973,15.46
″Jones, Suzanne″,12345,16.34
″Williams, Sam″,452,193.78
```

where (″) is a character string delimiter, (,) is a column delimiter and (.) is a decimal point. Alternatively, (;) can be used as a column delimiter, with (′) as the character string delimiter.

ASC Non-delimited ASCII files are used for loading data from other applications that create flat text files with aligned column data, such as word processors. Each ASCII file is a stream of ASCII characters consisting of data values organized by row and column. Rows in the data stream are separated by a line feed (or a carriage return/line feed). An example of an ASC is:

```
Smith, Bob          4973       15.46
Jones, Suzanne      12345      16.34
Williams, Sam       452        193.78
```

5.2.2 Sorting the Data (Optional)

If the amount of data being loaded is large in size, you may want to consider the sequence of the data in the target table and associated indexes for the load operation. This is an optional step that is not required by the load utility. However, it may improve the performance of the load operation. The steps to consider are the following:

- The logical design of the table
- Creating the indexes according to the nature of the anticipated queries or application requirements
- Before loading the input data, sort the data, using an external sort program with parameters that reflect the target table index environment

For the sort operation itself, you have a number of options. In a host environment, there are sort programs such as DFSORT which could be used, or in an AIX environment, there is the `sort` command. Alternatively, you could consider loading the data as you receive it and, subsequently, run the REORG utility provided with DB2. The REORG utility rearranges the data of a table into a physical sequence according to a specified index. Even in an operational environment, a table can become fragmented due to many updates, causing performance to deteriorate. Thus, apart from the load process itself, you should consider performing a REORG on heavily-used tables on a regular basis.

5.2.3 Creating a Target Table and an Exception Table

The load utility moves data into a target table which must already exist within the database. The target table may be a new table which you have created prior to the load or an existing table to which you will be appending or replacing data. Indexes on the table may or may not already exist. However, the load process only builds indexes that are already defined on a table.

In addition to a target table, it is recommended that an exception table be created to write any rows that violate unique index key or constraint violations. The exception table should be the same as the target table in every respect, except that it also contains two extra columns: a timestamp column and a messages column. The exception table can have more than n+2 columns, (n=the number of columns in the target table) but these columns (n+3) must be nullable or defaultable. All columns of the exception table should be free of any constraints. Constraints include referential integrity, check constraints and unique index constraints that could cause error on insert. The n+1 column of the exception table is an optional TIMESTAMP column. The n+2 column should be of type CLOB (Character Large Binary Objects) (1 MB) and is optional, but is used to give the name of the constraints that the data within the rows have violated. There is no enforcement of any particular name for the above-mentioned additional columns.

The exception table is an optional parameter to the `load` command. It is used to store copies of rows that violate unique index rules. By placing these rows in the exception table, you have the option after the load as to how to proceed with the violating rows. You may choose to alter them or ignore them. However, if the exception table is not created and not specified with the load utility, any rows that violate unique index rules will be discarded without any chance of recovering or altering them.

The following is an example of creation of a table and exception table that will be used for some of the examples in this chapter. The command has been placed in a file as follows:

```
create table cal.par
   (DEPTNO   SMALLINT(2) NOT NULL,
    DEPTNAME VARCHAR(36) NOT NULL,
    MGRNO    SMALLINT(2) NOT NULL
    PRIMARY KEY(DEPTNO));
```

This will create a table that will be used in three of the examples in 5.4, “Four Successful Load Scenarios” on page 103. For the exception table, we will mimic the definition of the target table, cal.par. There are two methods for creating the exception table. One uses the same DDL that was used to create the target table, with two additional columns for the TIMESTAMP and CLOB. If there are many rows in the target table and you have not saved the DDL used to create it, you can use the second method. The second method uses the import and export utilities. An example is as follows:

```
db2 connect to database
db2 ″export to anyfile of ixf messages anymsg select * from cal.par
    where mgrno < 0″
db2 import from anyfile of ixf messages anymsg create into cal.exp
db2 alter table cal.exp add column ts timestamp
db2 alter table cal.exp add column msg clob(1MB)
```

The exception table was created by exporting to a file the definition of the table, cal.par. A condition was specified for which there would be no rows (should the table contain data). Then the additional columns were added with the `alter table` command.

5.2.4 Determining the Storage Used in the Load Utility

There are two tables in the load process for which storage should be calculated, the target table and the exception table. When determining the storage requirements for loading data into the target table, there are two distinct situations:

- The table is being newly created, and the load will be an initial load of data.
- The table contains data, and the load will append to it.

The sizing of tables and containers used in tablespaces is covered in detail in 4.5.3, “Sizing of Tables and Tablespaces” on page 63. If appending to an existing table, you need to estimate what the effect of adding the input data will be on this table. In simple terms, if you are loading 1000 rows of data and each row is 512 bytes in size, then your table will grow to more than 512 kilobytes (allowing for some overhead).

Calculating the size of the exception table is difficult because it depends on the amount of data that does not get loaded to the target table. The exception table will usually be inspected and then discarded. The worst case is that the exception table should be equal to the number of rows being loaded, plus the storage required by the additional two columns. The recommendation is for the exception table to be placed in a separate tablespace that can be dropped after its use.

5.2.5 The Using Directory Option

When loading into a table that has indexes, temporary files will be created in the tmp subdirectory of the instance. Depending on the number and size of indexes, you may wish to have these temporary files created in another location. This can be done with the using option on the load command.

For example, you may wish for the temporary files to be created in the /tmp directory. The load command placed in a file may then look like the following:

```
load from calpar.ixf of ixf messages par.msgs
remote file par.remote
replace into cal.par using /tmp;
```

The amount of space required for the indexes will depend upon the amount of data being loaded and the size of the index, the number of columns within the index and the data type of each column. The *DB2 Administration Guide* provides information for calculating the size of indexes.

As an estimate for storage with the using directory option, calculate the sum of size of your index and allow for possibly twice that amount. As a rough estimate, you should calculate the size of the indexes based on the columns used, multiply by the number of rows being loaded and then multiply by two. This should give you enough space for the load command to successfully complete. The actual space used depends on the number of records and the uniqueness of the collating sequence.

Another consideration is supplying multiple directories to the using directory option. If using more than one directory for the temporary sort tables generated, they must be equal in size.

5.2.6 The Remote File Option

Temporary files are created during the load process. You may redirect the placement of those files using the remote file option of the load utility.

The remote file can be queried using the load query command in the event that a load has been placed in a pending state. For more information on the load query command, see 5.6, “The Load Query Command” on page 113.

The remote file identifies a base file name. DB2 actually creates three internal temporary files that make up the remote file. The files that comprise the remote file are basename.rid, basename.log and basename.msg, created by default in the DB2UTMP subdirectory. You will need to allow some space for these files. The amount of space necessary for the remote file will depend on the number of rows being input and if the table has a unique index that is violated by the rows being input. The basename.rid file stores information about the rows to be deleted in the delete phase of the load operation. These are the rows that have violated a unique key index. So the size of this portion of the remote file depends on the total number of rows being input and the possibility that they may violate a unique key index on the table.

5.2.6.1 Location of the Remote File

The location of the remote file may be specified via the remote file option in the load command. The default location depends on from where the load utility is started. If the command is issued locally at the server, the remote file by default will be placed in the current directory. If the load utility is executed remotely from a client, the remote file will, by default, be generated in the database directory.

5.2.6.2 Other Considerations for The Remote File Option

There are other considerations for using the remote file option with an explicitly stated directory/path. Consider an example of two loads being performed locally at a database server without specifying a directory for the remote file. By default, two loads in the same current directory on a local database server will cause the remote file to be placed in the same directory. If you do not specify an explicit path for the remote file in the load command, the remote file for both of these load operations will be directed not only to the same directory but also to the same file. Should you need access to the remote file, it would be difficult to determine which load operation the file is referencing. Therefore, use the remote file option and explicitly specify a separate directory for each load process. Should you have to restart one of the two load operations, having a fully qualified path for the remote file will avoid confusion.

5.2.7 The Copy Yes Option

The other option that requires additional storage consideration for the load operation is the copy yes option. The default is copy no, meaning that no copy of the load operation will occur. Copy yes creates an image of the table load that can be used in recovery situations.

However, the only time storage is a consideration is if the location for the copy is disk. You can also specify that the copy is directed to tape or to an ADSM server. If the copy is going to disk on the database server, the size consideration should be equivalent to that of the target table.

5.2.7.1 Use of the Load Copy

This copy image cannot be used as a restore copy. It can only be used during a rollforward command. Consider the situation where you have experienced a failure that causes the log files to be replayed. The load utility logs that a load has started and ended. It does not log the load itself. Therefore, if those log files are replayed during a recovery process, and include the load, the copy image is required to replay the load.

This option can only be used if archival logging is enabled. (For more information on logging, see Chapter 6, “Logging” on page 131.) If archival logging is enabled and the load command is issued with the default of copy no, a backup will be necessary when the load completes. If circular logging is used, no backup will be requested. You cannot specify copy yes with circular logging.

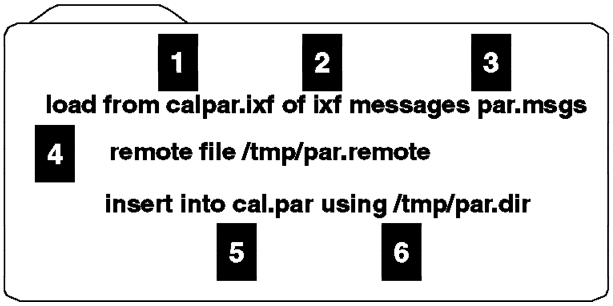
5.3 Using the Load Command

The load command has many parameters and options. Understanding them is important to using the utility.

NOTE

The input file for the load process must exist on the database server.

The `load` command can be issued from the command line, from the Command Line Processor (CLP) or from an API. It is recommended that the `load` command be placed in a file that can be edited and then run from the command line. For example, Figure 24 shows the contents of the file LOADEX.1.



```
load from calpar.ixf of ixf messages par.msgs
    remote file /tmp/par.remote
    insert into cal.par using /tmp/par.dir
```

Figure 24. Example Load Command

It can be executed as follows:

```
db2 -f LOADEX.1
```

The numbers shown in the `load` command file, LOADEX.1 are explained as follows:

1 The input file for the load is calpar.ixf.

2 The file type is ixf.

3 Messages will be directed to a file called par.msgs. This file can be viewed after the load has completed.

4 The temporary remote file names will be prefixed with “par.remote” and will be used during the load process. The remote file will be generated in the /tmp filesystem.

5 The action of load is insert. In our case, the table was empty before the load. The cal.par table is the target table. It gets loaded with all the input data as shown.

6 The using directory option will place any temporary sort files in the /tmp/sort.dir directory/filesystem.

The full syntax of the `load` command can be found in 5.3.1.1, “The Syntax of the Load Command” on page 101.

The `load` command has four different actions associated with it. Understanding the four different actions of load is important to the use of the utility. They are as follows:

INSERT When loading an empty table, you should specify an insert operation. Also, when appending data to an existing table with data, specify insert to add data to the table without changing the existing table data.

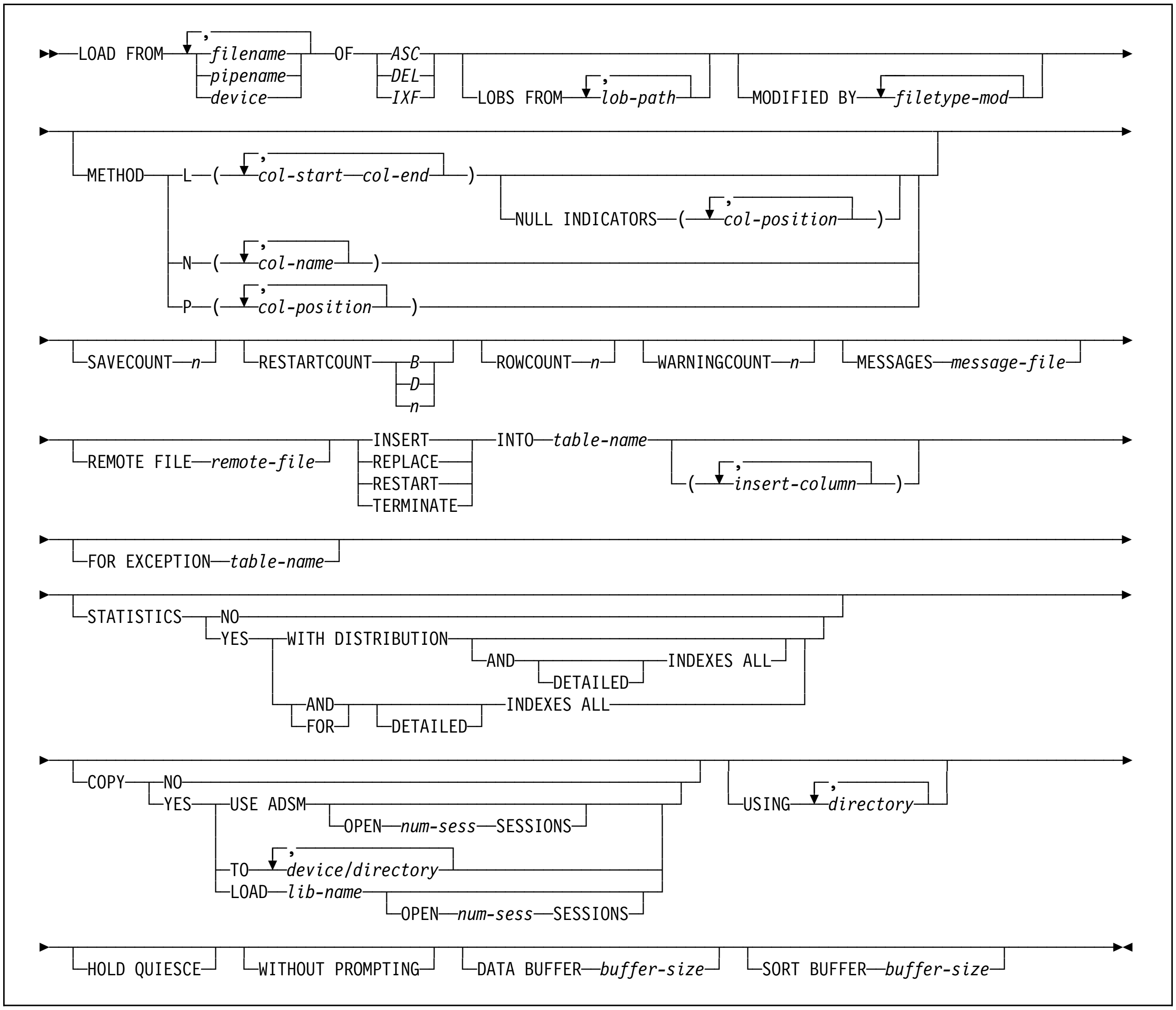
REPLACE If you specify replace in the `load` command, all existing data in a table will be deleted and new data from the input file will be loaded into the target table. The table definition and index definitions are not changed. You can only collect statistics under the replace option. REPLACE has several advantages over INSERT. You can do a REPLACE on a tablespace already in a load pending situation. Secondly, REPLACE may have better performance over INSERT if there are indexes to build during the load process.

RESTART After a load has been interrupted, this action is used to restart the load process. In such a situation, it is important to keep track of the last consistency point. This information is stored in the message file and the remote file.

TERMINATE This action terminates a previously interrupted load and moves the tablespaces, in which a target table resides, from a load pending state to a restore-pending state. The tablespaces cannot be used until a backup has been restored, and the tablespaces have been rolled forward.

5.3.1.1 The Syntax of the Load Command

The full syntax of the `load` command is:



For a description of all of the parameters, refer to the *Command Reference for common servers*.

Some of the parameters of the `load` command are as follows:

filename, pipename, device
This parameter identifies the source of the data being loaded. The source file, pipe or device must be on the same node as the database being loaded. If several data sources are identified, they will be loaded sequentially.

ASC, DEL, IXF
Specifies the format of the source data being loaded:

- ASCII (non-delimited ASCII format)
- DEL ASCII (delimited ASCII format)
- IXF (integrated exchange format, PC version)

method L, N or P There are three possible load method options: L, N or P.

- If the source data is an ASCII file, use the L parameter to identify the first and last byte of each column of data to be loaded.
- If the source data is an IXF file, use the N parameter to identify the name of the column to be loaded.
- If the source data is a delimited ASCII file, use the P parameter to identify the numbers of the columns to be loaded.

SAVECOUNT n This parameter is used to establish consistency points during a load after every n rows. The benefit of specifying this parameter is only realized in a recovery situation, where you can restart the load from the last consistency point, rather than from the first row again. However, there is a processing overhead in creating these consistency points; so you should only consider using them when loading large amounts of data (>1 hour load duration). Consider having a consistency point after loading 10 percent of the rows and at equal intervals thereafter. The default is for no consistency points, SAVECOUNT equaling 0.

RESTARTCOUNT n This parameter can have the following values

B - Specify a B to be used with the RESTART action of the load utility. The load will be restarted at the beginning of the build phase.

D - Specify a D to be used with the RESTART action of the load utility. The load will be restarted at the beginning of the delete phase.

n - Specify a number that is used with the RESTART action of the load utility. The load is restarted at n+1. The n is determined from the last consistency point found in the SAVECOUNT option. This last consistency point is stored in the remote file that is used during the load operation. You must use the `load query` command to view the remote file to obtain this consistency value.

message-file Specifies the location for warning and error messages that occur during the load. You can omit the message file only through using the CLP when invoking the `load` command. If a directory/file for the message file is not omitted, the messages are written to standard output. If the complete path to the file is not specified, load uses the current directory and the default drive as the destination. If the name of a file that already exists is specified, load appends the information to it.

remote file Identifies a base file name from which the system will create three internal temporary files (basename.rid, basename.log and basename.msg) during the load process. The default base file name is DB2UTMP, and the default location is in the current directory if issued locally at the database server, in the database directory if issued remotely. These files are destroyed on completion of the load, but some of their information content is copied to the message file. However, in the case of a load failure where you might not have access to the message file, the remote file can be interrogated by the `load query` command to identify the last consistency point reached (if the SAVECOUNT option has been specified).

into table-name
Specifies the target table within the database and, optionally, the table columns into which the data is to be loaded.

exception table-name
Specifies the exception table into which rows in error will be copied. An exception table is a user-created table which mimics the definition of the target table being loaded. This table is used to store copies of rows that violate unique index rules, have check constraint or foreign key violations, or invalid rows from a previous load operation. All columns of the exception table should be free of any constraint; constraints include referential integrity, check constraints and unique index constraints that could cause error on insert. The n+1 column of the exception table is an optional TIMESTAMP column; the n+2 column should be of type CLOB (1MB) and is also optional, but is used to give the name of the constraints that the rows have violated.

copy yes/no
If you specify the copy no option, the tablespace in which the table resides will be placed in a backup pending state after loading, if forward recovery is enabled. The data will not be accessible until a tablespace backup or a full database backup is made. If you choose the copy yes option, a copy of the changes caused by the load process will be saved either to an ADSM server, tape or directory.

using directory
When loading into a table that contains indexes, temporary files will be created in the sort directory. This directory is sqllib/tmp subdirectory of the instance owner′s home directory. This may be set to one or more alternative directories by including the using directory option.

5.4 Four Successful Load Scenarios

Four successful load examples are given to help understand the command parameters. By successful, either the input data has been completely loaded into the target table; or the data has been partially loaded into the target table, and we have a complete record of the failing rows that can be subsequently corrected and reloaded. Another aspect of the successful load is the completion of all three phases, indicating the total load operation has been completed.

5.4.1 Example 1 - Using the Load Utility

In this scenario, as represented in Figure 25, a user has prepared an input file (calpar.ixf) and created a target table (cal.par). The target table has been created using the DDL found in 5.2.3, “Creating a Target Table and an Exception Table” on page 95.

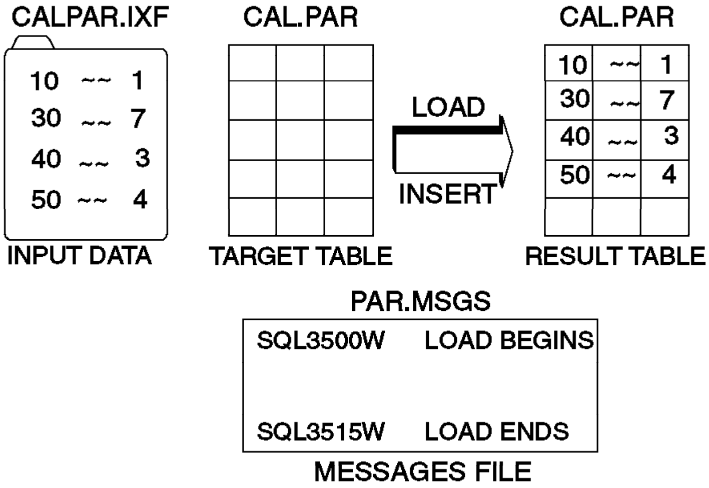


Figure 25. A Simple Load Scenario

The `load` command as placed in a file as :

```
load from calpar.ixf of ixf messages par.msgs
remote file /tmp/par.remote
insert into cal.par using /tmp/par.dir;
```

After the completion of the load, you may view the message file, par.msgs, using an editor of your choice. The message file after completion of the load will be similar to the following:

```
SQL3500W  The utility is beginning the ″LOAD″ phase at time ″08-25-1995
10:07:55.856344″.  1

SQL3109N  The utility is beginning to load data from file
″/home/inst02/data/table1.ixf″.  2

SQL3150N  The H record in the PC/IXF file has product ″DB2  01.00″, date
″19950814″, and time ″111218″.  3

SQL3153N  The T record in the PC/IXF file has name ″calpar.ixf″,
qualifier″, and source ″           ″.  4

SQL3110N  The utility has completed processing.  ″4″ rows were read from
input file.  5

SQL3519W  Begin Load Consistency Point. Input record count = ″4″.  6

SQL3520W  Load Consistency Point was successful.  7

SQL3515W  The utility has finished the ″LOAD″ phase at time ″08-25-1995
10:07:56.842711″.  8

SQL3500W  The utility is beginning the ″BUILD″ phase at time ″08-25-1995
10:07:56.884353″.  9

SQL3515W  The utility has finished the ″BUILD″ phase at time ″08-25-1995
10:07:57.009092″.  10
```

1 This is an informational message indicating that a phase is about to begin and that the previous phase has ended. During the load phase, data is loaded into your table. If there are any indexes to be built, the build phase will follow the load phase. If there were any duplicate keys found for a unique index, the delete phase will follow the build phase.

2 This is the normal beginning message.

3 Information is given about the product that created the PC/IXF file and when it was created.

4 Optional information is given about the name of the table where data was extracted, the product that created the table and the original source of the data.

5 This is the normal load phase ending message.

6 The load utility is about to attempt to perform a consistency point for the load phase.

7 The consistency point performed by load was successful.

8 This is an informational message indicating that the load phase has finished.

9 This is the normal beginning message for the build phase where the indexes are being created.

10 This is the normal ending message for the build phase after the indexes have been built.

There is no delete phase indicated as no rows were rejected from the load phase.

5.4.2 Example 2 - An Index Key Violation

In this load scenario, as represented in Figure 26, a user has prepared an input file, calpar.ixf, and created a target table, cal.par, and an exception table, cal.parexp. The target table has a unique key on the first column.

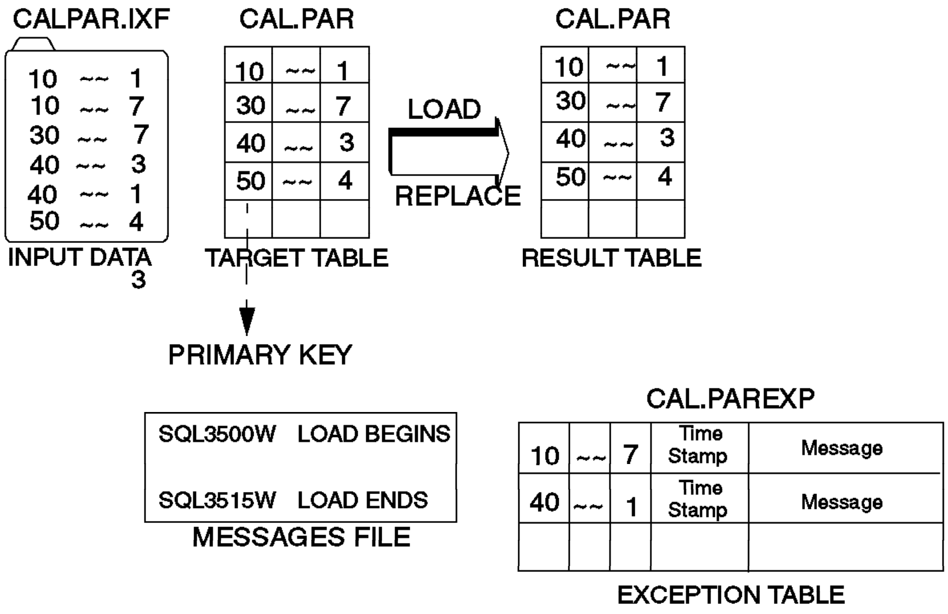


Figure 26. A Load with an Index Key Violation

In this situation, we want to replace records in the target table and we want to save any records that violate the unique key into the exception table. The `load` command as placed in a file is:

```
load from calpar.ixf of ixf messages par.msgs
remote file /tmp/par.remote
replace into cal.par for exception cal.parexp
using /tmp/par.dir;
```

If no exception table is specified in the `load` command, only the total number of rows which violate the unique key index will be displayed as a warning message on the screen. No copy of the specific rows will be kept.

From the message file, you will get the following information on completion of the load:

```
SQL3500W  The utility is beginning the ″LOAD″ phase at time ″08-25-1995
10:52:34.877189″.

SQL3109N  The utility is beginning to load data from file
″/home/inst02/data/calpar.ixf″.

SQL3150N  The H record in the PC/IXF file has product ″DB2  01.00″, date
″19950825″, and time ″103855″.

SQL3153N  The T record in the PC/IXF file has name ″calpar.ixf″,
qualifier″, and source ″            ″.

SQL3110N  The utility has completed processing.  ″4″ rows were read from
input file.

SQL3519W  Begin Load Consistency Point. Input record count = ″4″.

SQL3520W  Load Consistency Point was successful.

SQL3515W  The utility has finished the ″LOAD″ phase at time ″08-25-1995
10:52:35.851971″.

SQL3500W  The utility is beginning the ″BUILD″ phase at time ″08-25-1995
10:52:35.893648″.

SQL3515W  The utility has finished the ″BUILD″ phase at time ″08-25-1995
10:52:36.014830″.

SQL3500W  The utility is beginning the ″DELETE″ phase at time ″08-25-1995
10:52:36.815156″.

SQL3509W  The utility has deleted ″2″ rows from the table.  *

SQL3515W  The utility has finished the ″DELETE″ phase at time ″08-25-1995
10:52:37.079315″.
```

When a table with a unique index is loaded, rows causing a violation of the index will be deleted from the table during the delete phase. This message (*) provides information on how many rows have been deleted.

The load has ended successfully, but two input rows have been written into the exception table because they violated the index key. You need to decide if these rows do indeed contain the correct data or if they need to be corrected. In either case, you can then reload the rows into the target table from the exception table.

5.4.3 Example 3 - A Constraint Violation

In this load scenario, as represented in Figure 27 on page 108, a user has prepared an input file and created a target table (cal.par). In this case, the target table has a parent table (cal1.par) containing a unique key on the first column. The target table was created with a foreign key referenced on the parent table. This represents a scenario that can occur while you are loading data into a table with referential constraints.

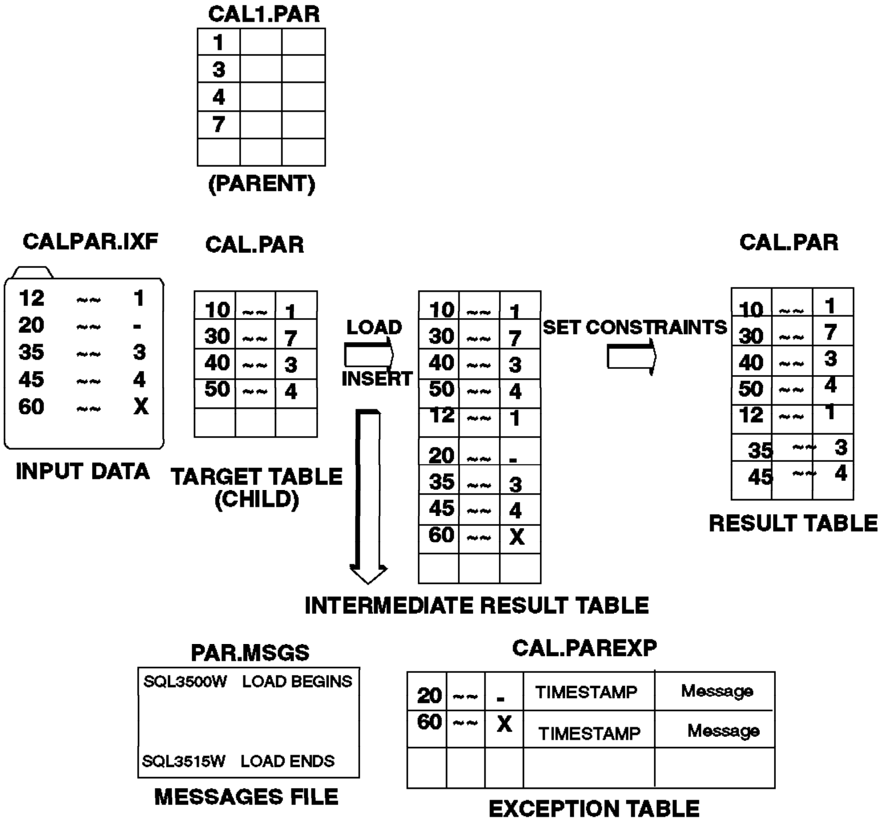


Figure 27. A Load with a Constraint Violation

The load command for this example as placed in a file is:

```
load from calpar.ixf of ixf messages par.msgs
remote file /tmp/par.remote
using /tmp/par.dir
insert into cal.par for exception cal.parexp;
```

From the message file, you can get the following information

```
SQL3500W  The utility is beginning the ″LOAD″ phase at time ″08-29-1995
10:40:32.654674″.

SQL3519W  Begin Load Consistency Point. Input record count = ″0″.

SQL3520W  Load Consistency Point was successful.

SQL3109N  The utility is beginning to load data from file
″/home/inst02/data/calpar.ixf″.

SQL3150N  The H record in the PC/IXF file has product ″DB2  01.00″, date
″19950825″, and time ″111045″.

SQL3153N  The T record in the PC/IXF file has name ″calpar.ixf″,
qualifier″, and source ″            ″.

SQL3110N  The utility has completed processing.  ″5″ rows were read from
input file.

SQL3519W  Begin Load Consistency Point. Input record count = ″5″.

SQL3520W  Load Consistency Point was successful.

SQL3515W  The utility has finished the ″LOAD″ phase at time ″08-29-1995
10:40:33.599667″.
```

This message file listing would seem to indicate that the load has completed successfully. In addition, the messages that appear on the workstation screen indicate that everything has proceeded normally:

```
    Number of rows read         = 5
    Number of rows skipped      = 0
    Number of rows loaded       = 5
    Number of rows rejected     = 0
    Number of rows deleted      = 0
    Number of rows committed    = 5
```

These messages still do not indicate that the load has encountered any difficulty. Next, you should check tablespace states by using this command:

```
db2 list tablespaces
```

The output of this command is:

```
 Tablespace ID                        = 4
 Name                                 = TS033
 Type                                 = Database managed space
 Contents                             = Any data
 State                                = 0x0020 *
   Backup Pending
```

***** 0x0020 means that the tablespace, in which the table resides, is in a backup pending state. Even though the tablespace state is not in a normal state, the load was successful. Backup pending is placed on a tablespace when the default option of copy no is taken and archival logging is being done on the database. After doing a tablespace backup as required by the state information, the tablespace is returned to a normal state. That is, the tablespace state will be 0x0000. However, if you try to select from the table, the following error message may be received:

```
SQL0668N Operation not allowed when the underlying table is in Check
Pending. SQLSTATE=57016
```

5.4.3.1 Check Pending

Check pending is a table state, not a tablespace state. It means that some of the rows from the operation, in this case a load, have violated a constraint condition. If you try to access a table that is in check pending, you will receive an error code of SQL0668N.

This message indicates that some of the rows attempted by the load violated a constraint. In this case, the child table, cal.par, has a foreign key that has violated referential integrity placed on it by the parent table, cal1.par. To verify the status of tables, check the table status in the syscat.tables system catalog with the following command:

```
   db2 ″select tabname, status, const_checked from syscat.tables
   where tabname in (′cal1.par′,′cal.par′)″
```

Following is an example where one of the tables is in check pending state.

```
TABNAME            STATUS CONST_CHECKED
------------------ ------ --------------------------------
cal1.par           N      YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
cal.par            C      NYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
```

Status C indicates check pending state; N is normal state. The N in the CONST_CHECKED column indicates that one constraint needs to be applied.

If a table is in the check pending state, then you should execute the set constraints command by using the exception table, cal.parexp. The command to do this is similar to the following:

```
   db2 ″set constraints for cal.par immediate checked
   for exception in cal1.par use cal.parexp″
```

This results in the following message:

```
SQL3602W  Check data processing found constraint violations and moved
them into the exception table.  SQLSTATE=01603
```

Even though the load process is complete, you could not access the table until the set constraints command is performed. As a final step, you can now verify that the status of the table in the syscat.tables system catalog is normal.

```
TABNAME            STATUS CONST_CHECKED
------------------ ------ --------------------------------
CAL1.PAR           N      YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
CAL.PAR            N      YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
```

A decision can be made about the rejected rows that were placed in the exception table as to what actions, if any, will be taken to correct the data, and either load or import them back to the target table. Alternatively, the constraints could be relaxed to accommodate the rows.

5.4.4 Example 4 - Loading LOBs

The load command also allows you to load large objects or LOBs into a table. If the LOBs are contained within the load input file or device, then the load command requires no additional parameters to include the LOBs. However, the LOBs may be located separate from the load file. They may be stored in a directory on the database server. The column in the input file for the load operation will contain the name of the LOB file on the server. For example, a non-delimited ASCII file may look like the following:

```
Test data     12345     lobfile1
More data here234556    lobfile2
```

In the above example, columns 1 to 14 represent the first column of the table and columns 15 to 24 may be the second column, while column 25 to 35 represent the LOB column and contain the name of the LOB file on the server.

In the second example, the ixf file and the LOB files may be created from an existing database table by using the export command. The following command exports the emp_resume table from the sample database.

```
        db2 ″export to emp_resume.ixf of ixf
        lobs to ′./lobs/′ modified by lobsinfile
        ′select * from emp_resume′″
```

Then, to load the exported data into a table called resume, you would use the following load command:

```
       db2 ″load from ′emp_resume.ixf′ of ixf
        lobs from ′./lobs′ modified by lobinsfile
        messages ′./resume.msg′ remote file ′./resume.rem′
        insert into resume″
```

5.5 What Happens If A Load Fails?

Diagnosing a load failure situation is a non-trivial exercise. Should the load utility fail, the tablespace will be placed either in a load pending or delete-pending state, which must be corrected to access the tablespace and the tables. The following points may be helpful in determining the problem:

1. An SQLCODE return code and a short explanation will be returned when an error occurs. This return code may be misleading since it may only yield information for a symptom, rather than the cause of the problem.

2. By viewing the messages file, you may get information about the progress of the load operation. This is where warning and error messages that occur

during the load will be written. However, the messages file may not be available for viewing, depending on the failure and the phase in which the failure occurred. The remote file may be viewed using the `load query` command. The remote files are temporary internal files from which information from the remote file gets written to the messages file. This is explained in 5.6, “The Load Query Command” on page 113.

3. Finally, you could check the db2diag.log file to understand the sequence of events taking place within the database. The path of the db2diag.log file is set by the database manager configuration parameter, DIAGPATH. By default, DIAGPATH is not set. The default placement of db2diag.log is in the db2dump subdirectory for AIX and in the instance directory for OS/2. A time stamp will tell when the error occurred. The instance name, function name and database alias are all listed to assist in pinpointing the error. A diagnostic message (DIAxxxx) explains the error that occurred.

Given an understanding of the database environment you are working with, the above points should allow you to determine the current state of your database, tablespaces and tables.

The following table contains most of the SQL return codes which can occur during a load operation, along with an explanatory note.

Table 21 (Page 1 of 2). Valid Load Parameters, Invalid Conditions

Description of Situation	**SQL Return Code**
Invoke load when not connected to database	1024
Input file does not exist	3025
Do not have read permission on input file	2061
Directory specified for remote file does not exist	3508
No write permission on remote file	3508
Directory specified for message file does not exist	3006
Copy_target does not exist	3508
Copy_target not writable file	2061
Working directory does not exist	3508
Copy yes, forward recovery disabled for the database	3522
Ask for statistics with insert option	2032
Table specified does not exist	3304

Table 21 (Page 2 of 2). Valid Load Parameters, Invalid Conditions

Description of Situation	**SQL Return Code**
Exception table specified does not exist	3304
Exception table and load table do not match	3604
Constraints/triggers on exception table	3604

5.6 The Load Query Command

If your load operation fails, you will need to decide if you should either restart the load operation or restore to a prior state. The `load query` command can be used to get more diagnostic or recovery information to speed up the recovery/restart process. It allows you to access the temporary files (remote file) that are created during the load process to identify at what point the load failed and where it can be restarted from. To use it, you require a connection to the database. The command can be used by local or remote users. The syntax for the `load query` command is:

```
►►──LOAD QUERY──remote file──TO──local-message-file──────────────────────────────►◄
```

where

remote file Identifies the base file name from which the system created three internal temporary files (basename.rid, basename.log and basename.msg) during the load process. The default base file name is DB2UTMP, and the default location depends where the command was invoked. If the load is started from the database server, the default directory is the current directory. If the load is initiated from a remote client, the default directory is the database directory. These files are destroyed on completion of the load, but some of their information content is copied to the messages file. However, in the case of a load failure where you might not have access to the messages file, the remote file can be viewed by the `load query` command to identify the last consistency point reached (only if the SAVECOUNT option has been specified in the `load` command).

local-message-file Specifies a file where you want the contents of the load query command sent to. This file should not be the messages file identified in the `load` command.

The output of a `load query` command is similar to the messages file. If during the load the system crashed, the messages file will never be built. However, you can still view the remote file with the `load query` command to assess the progression of the load and to identify the last consistency point, having specified the SAVECOUNT option in the `load` command. (See **1** and **2** in the remote file example below). The load command was executed from the following file:

```
load from staffbig.ixf of ixf messages staffbig.msgs
remote file staffbig.remote
insert into staffbig.tab for exception staffbig.exp
SAVECOUNT 1000
```

At 1 and 2, consistency points were established after 1000 and 2000 rows were loaded. You could restart from either of these points without having to repeat the entire load. That is, you could restart the load using the restart option and restartcount 1000 or restartcount 2000.

```
SQL3500W The utility is beginning the ″LOAD″ phase at time
″07-13-1995 19:40:29.645353″.

SQL3519W Begin Load Consistency Point.  Input record count = ″0″

SQL3520W Load Consistency Point was successful.

SQL3109N The utility is beginning to load data from the file
″/u/mydir/data/staffbig.ixf″

SQL3150N The H record in the PC/IXF file has product ″DB2 01.00″,
date ″19950611″, and time ″194554″.

SQL3153N The T record in the PC/IXF file has name
″data/staffbig.ixf″, qualifier ″ ″, and source ″ ″.

SQL3519W Begin Load Consistency Point.  Input record count=
″1109″. 1

SQL3520W Load Consistency Point was successful.

SQL3519W Begin Load Consistency Point.  Input record count=
″2117″. 2

SQL3520W Load Consistency Point was successful.
```

Note that although the SAVECOUNT option was specified as 1000, the consistency points do not occur exactly every 1000 rows. An approximation takes place.

5.7 Load Command Tests

We carried out a number of tests to determine the operating characteristics of the load process under various environmental conditions. We then identified how to resolve the resulting error situations. The error situations result in the following pending states:

- Load pending
- Delete pending

The condition that caused these pending states is likely to be one of these:

- Storage Media Constraint
- DB2 System Crash

The impact and resolution of these situations varies depending on environmental and configuration factors you have put in place:

- Logretain off (circular logging enabled)
- Logretain on (archive logging enabled)
- Using Database Managed Storage/Space (DMS) for user tables and tablespaces
- Using System Managed Storage/Space (SMS) for user tables and tablespaces
- Copy yes (copy of load being made during load)
- Copy no (no copy of load being made)

5.8 Storage Media Constraints

From the many failure situations that may occur, the most likely will be storage related. In this section, we show test results for five possible situations where different areas of storage are exhausted and cause the load to fail.

5.8.1 Remote File

When performing a load, there are three temporary files created which are collectively referred to as the remote file. These files are created on the server, and their location may be specified in the load command. These files may grow quite large and may possibly fill the storage media. By default, the three files that make up the remote file are placed in the current directory is the load is started from the database server. If the load is started from a remote client, the default location is the database directory.

Using the following load command, we simulated the situation where the remote file ran out of storage space.

```
load from calpar.ixf of ixf.messages par.msgs remote file par.remote
replace into cal.par
```

When the remote file ran out of storage, the SQL return code was:

```
SQL3508N Error in accessing a file of type ″LOGFILE″ during load or load
query. Reason code: ″3″. Path: ″par.remote.log″
```

The explanation of this message is that an error occurred in trying to open, read from, write to, or close the remote log file during the processing of the load command. The prescribed user response is to ensure that your remote file directory is specified properly; there must be enough disk space to write out index keys for all the indexes to be built in this directory.

You should check the state of the tablespace where the target table resides. Depending on the stage at which this error occurred, you may need to take further action to recover fully. If the tablespace is in a recoverable state, then you may correct the error and re-run the load. Here, correcting the error would be to redirect the remote file to a larger filesystem/directory.

It is possible that the load failed at a point that it is unable to fully recover from. You may try using the load query command to check the status of the load.

```
db2 load query par.remote to par.query
```

You will normally receive information about the load status; however, in some failure situations, you may receive the following message:

```
SQL3523W  There are no messages to be retrieved from the message file.
Reason code: ″1″.
```

Reason code ″1″ means that the remote file does not exist. The recommended recovery action is to perform one or more of the following tasks:

1. Remove files or increase size of file system where the remote file is located. After more space has been allocated, you can restart your load from the beginning by again using this command.

   ```
   load from calpar.ixf of ixf messages par.msgs
   remote file par.remote
   restart into cal.par
   ```

2. Alternatively, you could do the following:

   a. Reissue the load again with the replace option and using as input an empty file (zero length).

   b. This should take the tablespace out of load pending

   c. Drop the table

   d. Reissue the load command.

5.8.2 Using the Directory Option of the Load Command

The using directory option of the load command redirects the temporary sort files that are built when a load is performed on a table that has indexes. Should these files fill the directory/filesystem that they are stored in, the load will fail with the following error:

```
SQL3508N  Error in accessing a file of type ″SORTDIRECTORY″ during load
or load query.  Reason code: ″3″.  Path: ″/home/data/small/″.
```

The load has failed, and the tablespace will be left in a load pending state. To recover from this, you will need to increase the size of the storage available to the using directory option, and restart the load command.

5.8.3 DMS Tablespace and Container

Having created a target table in a DMS tablespace, if the extent size or the size of the load has been miscalculated, it is possible for the DMS tablespace to fill up. The following command was used in creating a situation where this occurred.

```
load from calpar.ixf of ixf savecount 5000 messages par.msgs
remote file par.remote replace into cal.par
```

When the DMS container ran out of storage, the SQL return code was:

```
SQL0289N Unable to allocate new pages in a tablespace  SQLSTATE=57011
```

The explanation of this message is that all the containers assigned to this tablespace are full. Details can be found in the db2diag.log and/or the database manager error log.

The tablespace states looked like this:

```
 Tablespace ID                        = 3
 Name                                 = TS03
 Type                                 = Database managed space
 Contents                             = Any data
 State                                = 0x0008  *
 Total pages                          = 1000
```

* 0x0008 means Load Pending state.

View the message file or remote file (if no message file has been generated) to find the last successful commit point:

```
SQL3500W  The utility is beginning the ″LOAD″ phase at time ″08-31-1995
13:56:42.659689″.

SQL3519W  Begin Load Consistency Point. Input record count = ″0″.

SQL3520W  Load Consistency Point was successful.

SQL3109N  The utility is beginning to load data from file
″/home/inst02/data/ixf.AODB24.AOAOCTTD″.

SQL3150N  The H record in the PC/IXF file has product ″DB2   01.00″, date
″19950509″, and time ″153923″.

SQL3050W  Conversions on the data will be made between the IXF file code
″850″ and the application code page ″819″.

SQL3153N  The T record in the PC/IXF file has name ″ixf.AODB24.AOAOCT″,
qualifier ″        ″, and source ″            ″.

SQL3519W  Begin Load Consistency Point. Input record count = ″6042″.

SQL3520W  Load Consistency Point was successful.

SQL3519W  Begin Load Consistency Point. Input record count = ″12122″.

SQL3520W  Load Consistency Point was successful.

SQL3519W  Begin Load Consistency Point. Input record count = ″18202″.

SQL3520W  Load Consistency Point was successful.

SQL3519W  Begin Load Consistency Point. Input record count = ″24272″.

SQL3520W  Load Consistency Point was successful.

SQL3519W  Begin Load Consistency Point. Input record count = ″30352″.

SQL3520W  Load Consistency Point was successful.

SQL0289N  Unable to allocate new pages in a table space  SQLSTATE=57011

SQL0289N  Unable to allocate new pages in a table space  SQLSTATE=57011
```

To recover from the load fail situation, add new container(s) to the tablespace, and try the operation again. The command to add a new container to an existing DMS tablespace is:

```
   db2 ″alter tablespace caltbsp add (file ′/calpath′ 1000)″
```

In this example, we are adding 1000 new pages to the tablespace. You should re-estimate the required size of your tablespace based on the data being input, and add sufficient pages to allow the load to complete successfully.

Before continuing, you should ensure that the data has been re-balanced across all the available containers. To check this, you should look for the following message in the db2diag.log, which is located in the db2dump subdirectory of the instance (AIX), in the instance directory (OS/2) or as set by the DBM configuration parameter, DIAGPATH (AIX and OS/2).

```
inst02 pid(29116) process (db2rebal)
buffer_pool_services  sqlb_rebalance Probe:2877

Rebalancer completed successfully 0003 *
```

* Check that this number correctly matches the tablespace ID.

You may now restart the load from the last successful consistency point. In our example, this was at row 30352.

```
db2 ″load from calpar.ixf of ixf restartcount 30352
messages par.msgs remote file par.remote restart into cal.par″
```

At this stage, the tablespace states are:

```
Tablespace ID                         = 3
Name                                  = TS03
Type                                  = Database managed space
Contents                              = Any data
State                                 = 0x0020 *
Total pages                           = 4000
```

* 0x0020 means backup pending. You may notice an increased number of pages for this tablespace. This was after the container was added to the tablespace. You are now required to perform a backup before the tablespace will return to normal state.

5.8.4 Container in an SMS Tablespace

As with the container example in an DMS tablespace, it is possible for a container in an SMS tablespace to fill during a load. However, in an SMS tablespace, you cannot add another container. You can increase the size if it is an AIX filesystem and if space exists, or delete files from the container (both AIX and OS/2). Having created a target table in an SMS tablespace, and with logretain on, we used the following load command to cause the situation where the SMS tablespace filled during the load.

```
db2 ″load from ixf.file of ixf messages parmsgs
remote file par.remote insert into caltable″
```

The following message will be received when the SMS tablespace storage is exhausted:

```
SQL0901N  The SQL statement failed because of a non-severe system error.
Subsequent SQL statements can be processed.  (Reason ″-10740″.)
```

If you cannot free disk space by removing files from the filesystem or increasing the size of the filesystem, there are several options you may consider.

1. You may try to re-issue the load command using the restart option with RESTARTCOUNT B. This will truncate the table to the last consistency point achieved during the load process. This may be sufficient for your storage requirements. However, by specifying a RESTARTCOUNT B to the restart option, the load utility will start at the build phase. Should you have indexes, you may still encounter storage problems.

2. As another alternative, you could restart the load using the restart option with RESTARTCOUNT n. However, n in this situation should be greater than

the total number of rows in your input file. Again, this will truncate the table to the last consistency point.

3. Another alternative is to perform a full database restore to recover from the failure. This is described further.

```
restore database caldb from '/calpath'
```

If you are doing a database restore, you will also rollforward your database to a point in time just before the load was started. The timeline below represents the different points in time when the backup, load, failure, and restore take place.

```
       T                      T+1     T+2          T+3
time____|______________________|_______|____________|______
        |                      |       |            |
     Backup                  Load   Load Fails    Restore
```

You have to rollforward the database to time > T and < T+1 to return to normal state. The command to rollforward to the time 8:33 on 19 July 1995 would look like the following:

```
rollforward database caldb to '1995-07-19-08.33.00' and stop
```

If you specify logretain off, the recovery action is the same, except that you will not rollforward. Instead, you restore to the point of the last backup, and all transactions since the backup will be lost.

5.8.5 Copy Target File/Device

If you are using the copy option of the load command, a situation may exist where the target file or device is exhausted of storage. To create this situation, we used the following load command:

```
    db2 "load from calpar.ixf of ixf messages par.msgs
    remote file par.remote insert into cal.par copy yes to '/calpath'"
```

This is the message you obtain from the system:

```
SQL3706N  A disk full error was encountered on "/calpath".
```

Your tablespace looks like this:

```
  Tablespace ID                         = 3
  Name                                  = TS03
  Type                                  = Database managed space
  Contents                              = Any data
  State                                 = 0x0008  *
```

* 0x0008 means Load Pending state. Because you have specified the copy yes option of the load command and the filesystem/directory for the copy was not large enough, your tablespace is in a load pending state. You have two choices:

- Free-up storage in the container (filesystem). Here, we used /calpath.

- Change the target directory for the copy.

It is recommended that you restart the load from the beginning. The copy image from the initial load did not complete properly.

To restart the load, we used the following command:

```
   db2 ″load from ixf.file of ixf savecount 5000 messages par.msgs
   remote file par.remote restart into table-name
   copy yes to ′/newcalpath′″
```

You cannot specify copy yes in a logretain off environment. If you try to do this, you will receive the following message:

```
SQL3522N  A copy target cannot be provided when both log retain and
user exits are disabled.
```

5.9 DB2 System Crash

In the event that the DB2 system crashes due to some operating or processing system error or user intervention, the DB2 agent or process controlling a load will be killed. Only one scenario is presented: one where a load copy is not being taken, the default, and LOGRETAIN is enabled.

5.9.1 DB2 Agent/Process Killed and Copy No Option

With the copy no option, a tablespace will be placed in a backup pending state after the load completes.

The load command issued was :

```
load from calpar.ixf of ixf savecount 5000 messages par.msgs
remote file par.remote restart into cal.par
```

The SQL return code that will appear on the screen is:

```
SQL1224N  A database agent could not be started to service a request, or
terminated as a result of a database system shutdown or a force command.
SQLSTATE=55032
```

This is the only diagnostic information available to you as the entire DB2 instance will be brought down, and you will not be able to execute the load query command because you have no database connection.

At this point, you will have to start the database manager again, and connect to the database. Now, the tablespace states are:

```
Tablespace ID                          = 3
Name                                   = TS03
Type                                   = Database managed space
Contents                               = Any data
State                                  = 0x000c *
Total pages                            = 4000
```

* 0x000c means Quiesced Exclusive and Load Pending.

The action to take is to perform a quiesce reset to put the tablespaces in a load pending state only. Here is the command:

```
db2 ″quiesce tablespaces for table caltable reset″
```

The tablespace states are now:

```
Tablespace ID                          = 3
Name                                   = TS03
Type                                   = Database managed space
Contents                               = Any data
State                                  = 0x0008 *
Total pages                            = 4000
```

* 0x0008 means load pending.

A remote file will have been generated if you have specified it in your `load` command. Execute the following `load query` command to extract information from the remote file on the progress of the load process:

```
db2 ″load query par.remote to par.query″
```

Here are the contents of the remote file:

```
SQL3500W  The utility is beginning the ″LOAD″ phase at time ″08-31-1995
15:05:10.793656″.

SQL3519W  Begin Load Consistency Point. Input record count = ″0″.

SQL3520W  Load Consistency Point was successful.

SQL3109N  The utility is beginning to load data from file
″/home/inst02/data/ixf.AODB24.AOAOCTTD″.

SQL3150N  The H record in the PC/IXF file has product ″DB2   01.00″, date
″19950509″, and time ″153923″.

SQL3050W  Conversions on the data will be made between the IXF file code
″850″ and the application code page ″819″.

SQL3153N  The T record in the PC/IXF file has name ″ixf.AODB24.AOAOCT″,
qualifier ″        ″, and source ″            ″.

SQL3519W  Begin Load Consistency Point. Input record count = ″6042″.

SQL3520W  Load Consistency Point was successful.

SQL3519W  Begin Load Consistency Point. Input record count = ″12122″.

SQL3520W  Load Consistency Point was successful.

SQL3519W  Begin Load Consistency Point. Input record count = ″18202″.

SQL3520W  Load Consistency Point was successful.

SQL3519W  Begin Load Consistency Point. Input record count = ″24272″.

SQL3520W  Load Consistency Point was successful.
```

You can use the file information to restart the load at the correct point, if you have specified the SAVECOUNT option in the load command. The number of rows you have successfully loaded becomes the starting point (RESTARTCOUNT) for the load restart process. Execute the following command to restart the load process:

```
   db2 ″load from calpar.ixf of ixf restartcount 24272
   messages par.msgs remote file par.remote restart into cal.par″
```

Messages appear after you restart the load. The “Number of rows skipped” is the number you found on the local message file after the load query, a number that you specified in the RESTARTCOUNT option. Other messages that appeared in our test were:

```
                Number of rows read = 61608.
                Number of rows skipped = 24272.
                Number of rows loaded  = 61608.
                Number of rows rejected  = 0
                Number of rows deleted = 0
                Number of rows committed  = 61608.
```

After the restart load has completed, the tablespaces are in a backup pending state (0x0020). Perform the backup to return them to normal state.

5.10 Load Statistics

The load utility allows for building indexes and gathering statistics for a table as part of the load. This is more efficient than executing the three phases separately. Keys are sorted and statistics are collected during the load phase, which removes some of the overhead involved in issuing separate CREATE INDEX and RUNSTATS statements, but for larger tables, the number of indexes and whether statistics are collected or not become key factors in the total load time. Collecting statistics will affect the performance of the load.

One of the parameters of the `load` command is whether or not you require statistics to be gathered during the load process. By specifying statistics yes, statistics will be gathered for the target table and for any existing indexes. However, this option is not supported if the load is in insert or restart mode. It is only supported in replace mode. Parameter options include:

- With Distribution - means that distribution statistics are kept.
- And Indexes All - update statistics for both the table and its indexes.
- For Indexes All - update statistics for the indexes only.
- Detailed - means that extended index statistics are requested.

By specifying statistics no, no statistics will be gathered, and that the statistics in the catalogs will not be altered.

5.11 Import/Export

Although both the import and load utilities are used to load data into a database, it is important to understand the differences between the two utilities:

Table 22 (Page 1 of 2). Differences between the IMPORT and the LOAD Utilities

The Import Utility	**The Load Utility**
Creation of table and indexes supported with IXF format	Table and indexes must exist
WSF format is supported	WSF format is not supported
Can import into aliases, views and table	Can load into tables only
The tablespace(s) that contain the table and its indexes are online for the duration of the import	The tablespace(s) that contain the table and its indexes are offline for the duration of the import
Triggers will be fired	Triggers are not supported
If an import is interrupted and a commit count had been specified, the table is usable and will contain the rows that were loaded up to the last commit. The user has the choice to restart the import or use the table as it is	If a load is interrupted and a SAVECOUNT option had been specified, the table remains in load pending state and cannot be used until the load is restarted to continue the load or until the tablespace is restored from a backup image created some time before the load

Table 22 (Page 2 of 2). Differences between the IMPORT and the LOAD Utilities

The Import Utility	**The Load Utility**
All constraints are validated during an import	Uniqueness is verified during a load, but all other constraints must be checked using the check data API
The keys of each row are inserted into the index one at a time during the import	During a load, all the keys are sorted, and the index is built after the data has been loaded. There will not be any page splits in the indexes following a load
If up-to-date statistics are required after an import, RUNSTAT must be executed	Statistics can be gathered during the load
You can import into host database with DDCS	You can not load into host database
Files that are imported must reside on the node where import is invoked	Files/pipes that are loaded must reside on the node where the database resides

The following changes have been made to import/export to support new data types in Version 2:

1. UDTs - During export, the base type will be stored in IXF files. If using the IXF file to create a new table doing an import, the new table will have the base type that the column types have instead of the UDT. The import utility will support putting a base type that can be cast into the UDT. Import will cast the UDTs so that the qualifier is explicitly stated when importing into a UDT.

2. LOBs - For export, you can select LOB column types and have the data stored in the file itself or in separate files for each LOB column/row by using the filetype-mod option lobsinfile and the LOBPATH/LOBFILE parameters. The LOBPATH parameter specifies the paths in which the individual files containing the LOBs are to be placed and must end with a valid path separator. If it is not provided but lobsinfile is specified, the same path as the datafile will be used.

3. Support for non-atomic compound SQL is added to import/export for performance reasons, especially when in use with DDCS gateway machines connecting to host database. Insert statements are blocked together and sent in a block to limit the network traffic as compared to doing individual insert statement. Back-level servers not supporting compound SQL will use individual INSERT statements.

5.11.1 Converting Between SMS and DMS Tablespaces

By using the export and import/load utilities, you can move data from an SMS tablespace environment to a DMS tablespace and back again. This may be necessary for operational and/or performance reasons. Because the export utility exports data from a database into an operating system file, it does not carry with it any references to the tablespace type it was stored in. Once the data is in an operating system file, you can then input it back into the database by using the import or load utilities and selecting the tablespace type of your choice.

An example of the command for exporting data is:

```
db2 ″export to staff.ixf of ixf select * from userid.staff″
```

The following information is required when exporting data:

- A SELECT statement specifying the data to be exported.
- The path and name of the operating system file that will store the exported data.
- The format of the data in the file. This format can be IXF, WSF (worksheet or spreadsheet format), or DELimited ASCII. Note that IXF is the recommended format for transferring data between DB2 common server databases. To provide compatibility within the DB2 common sever family, the export utility creates files with numeric data in Intel format, and the import utility expects it in this format.
- A message file name.

The import utility inserts data from an input file into a table or a view. If table or view receiving the imported data already contains data, you can either replace or append the existing data with the data in the input file.

An example of the command for importing data is:

```
db2 ″import from stafftab.ixf of ixf insert into userid.staff″
```

The following information is required when import data to a table or a view:

- The path and input file name where the data to import is stored.
- The name or alias of the table or view where the data will be imported.
- The format of the data in the input file. This format can be IXF, WSF (worksheet or spreadsheet format), delimited ASCII or non-delimited ASCII.
- Whether the data in the input file is to be inserted, updated, replaced, or appended to the existing data in the table or view.
- A message file name.

For delimited ASCII, WSF and ASCII data-file formats, define the table, including column names and data types, before importing the file. The data types in the operating system file fields are converted into the corresponding type of data in the database table. For IXF data file formats, the table does not need to exist before beginning the import; it can be automatically created when the data is imported.

If you use the load utility to input a previously exported file, be aware that it does not support the WSF data type; that is data stored as spread sheets. This restricts you to either using IXF or or delimited ASCII files if using export followed by load in converting tablespace types.

5.12 Data Propagator Relational (DPropR)

Data Propagator Relational copies data automatically between DB2 relational database management systems on platforms supporting Distributed Relational Database Architecture (DRDA) connectivity.

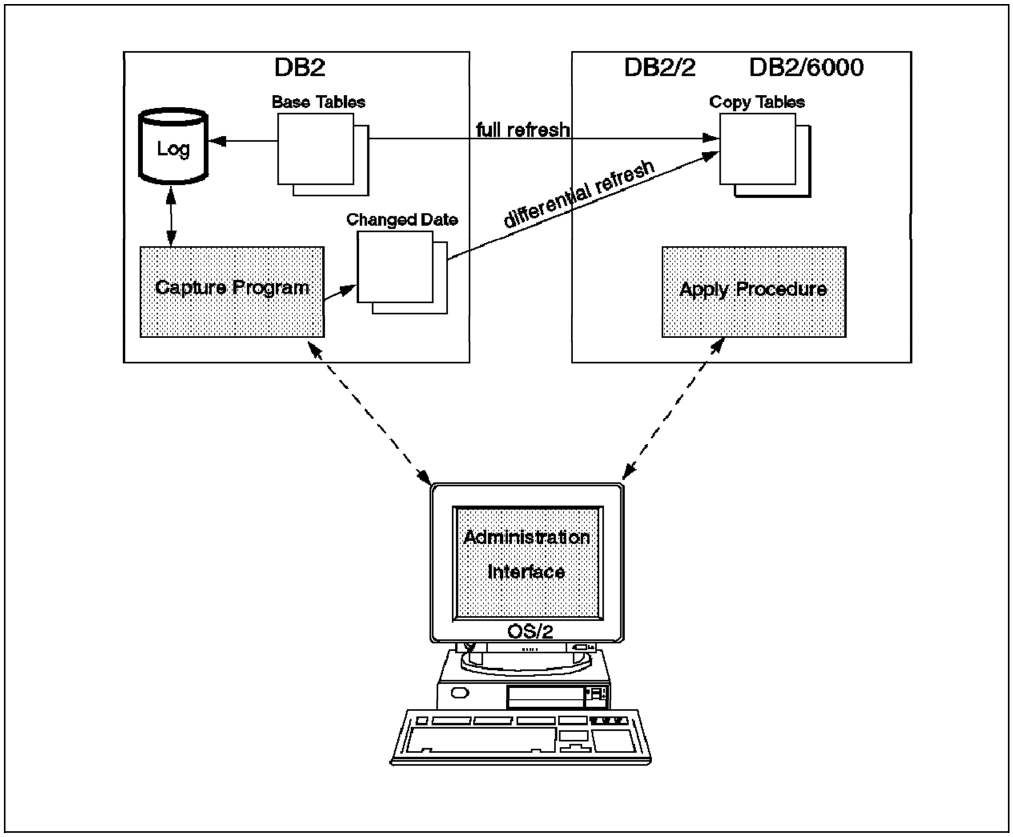


Figure 28. Data Propagator Relational

There are three pieces or components to the DPropR product: the Change Capture Program, the Apply Program and the Administrator Facility.

The Change Capture Program captures the changes from selected tables. These tables can be external tables containing SQL data from a file system or non-relational database manager, from existing tables in the database or from tables that have previously been updated by the Apply Program.

The Apply Program takes the changes from the changed data tables and copies the data to the target tables.

The Administrator Facility assists with creating and maintaining the copying environment.

There are no standard program linkages between these distributed components. All control data and user data flows between them via SQL tables. None of these components has any dependency on any particular type of physical network, network protocol or communications API.

Each of the components is able to operate, regardless of the state of the other components. Changed data is stored in SQL tables; so user data is not lost as a result of a program failure.

Refer to the *DataPropagator Relational User′s Guide* for further information.

5.13 Customer Scenario

This section discusses the way in which the telephone company will handle moving data into a DB2 database.

5.13.1 Telephone Company

The database being used by the phone company is reloaded every three months as the DB2/MVS database that is used as a "source" to propagate data to the DB2/AIX database, is also reloaded. Reloads of DB2/MVS databases take place when new data is acquired from external information systems.

This "data acquisition" is scheduled to occur over the next two years. The volume of data being acquired impedes use of the regular channel of updating tables through DPropR. Planning for these upcoming eight loads will use the following schedule:

- Loads will take place during weekends. Data is first loaded into the DB2/MVS database. After loading, it goes through a validation process, and it is dumped to tape. DB2/MVS administrators are committed to have the tapes available by Sunday at 8 a.m.
- DB2/AIX administrators will load the database from these tapes. The load process deadline is Sunday at 4 a.m. By then, the entire database must be loaded. Functional tests will take place from 4 p.m. to 6 p.m.
- In case of severe problems during the dump/load of the data, administrators will restore their backups. The time window for restoring the databases (both DB2/MVS and DB2/AIX) is from Sunday at 6 p.m. to Monday at 6 a.m.

Before loading the tables, the AIX database administrator will take a complete backup of the database. As tables to be loaded will inevitably be larger that the previous tables, special attention should be given to the sizes of the tablespaces. Containers should be added to tablespaces. Information on the storage requirements of the tables that will be loaded is gathered during the days prior to the load.

Before loading data, it is beneficial to lower the amount of memory assigned to the buffer pool and increase the size of the utility heap size. This heap indicates the amount of memory that the backup, restore and load utilities can use. By default, it is 5000 4 KB pages. Since the buffer pool is not used while loading data, the database administrator lowered it to 100 MB during the load. Memory freed from the buffer pool is given to the utility heap size. The UTIL_HEAP_SZ is increased to 40000 4 KB pages. This parameter is a database configuration parameter.

For the load process, tables are classified according to the tablespace where they are stored. Grouping tables into tablespaces was made according to the size of the tables and the frequency of modifications.

Table 23. Phone Company - Tables Per Tablespace

space01	**space02**	**space03**	**space04 to space11**
25 Tables	9 Tables	9 Tables	1 Table/tablespace

All the tables being loaded share the following options of the load utility:

- Loading is done from the 3490 tape device.
- The format of the source data being loaded is delimited ASCII.
- The load action will be REPLACE. REPLACE will delete all the existing data from the table and will insert the loaded data. Table and index definitions are not changed.
- No exception tables will be created as the data is already “verified” in DB2/MVS.
- DATA BUFFER is set to 20000 4 KB pages. This is memory allocated for data buffers during the load.
- SORT BUFFER is set to 10000 4 KB pages. This is memory used for sorting key index.

5.13.1.1 Tables Stored in Tablespace space01

These 25 tables are small tables (under 1000 rows). Since they are grouped in a single tablespace, the load will be performed with the copy yes option, so the tablespace will not be placed in a backup pending after each table is loaded. When the 25 tables of the tablespace are loaded, the database administrator will take a tablespace backup. Since the number of rows of each table is small, the SAVECOUNT option will not be specified. Messages will be kept for statistical purposes. For these tables, the load utility will be invoked with the following options:

- SAVECOUNT will not be specified.
- Messages will be sent to a different message file for each table.
- STATISTICS YES and INDEXES ALL options will be specified so no REORGs will be needed after the load.
- COPY YES, as already discussed.

5.13.1.2 Tables Stored in Tablespaces space02 and space03

These 18 tables are medium size tables (between 1000 and 100000 rows). Tables will be loaded with the copy yes option to avoid having to take a backup after each load. The SAVECOUNT option is set to 20000 rows, to minimize the time required to reload the table if the load is interrupted. For these tables, the load utility will be invoked with the following options:

- SAVECOUNT 20000
- Messages will be sent to different message file for each table.
- A different remote file will be specified for each table.
- Statistics yes and indexes all
- Copy yes

5.13.1.3 Tables Stored in Tablespaces space04 to space11

These nine tables are large tables (more than 100000 rows). Each table is stored in a separate tablespace. Tables will be loaded with the copy no option. Loading with this option will place the tablespace in a backup pending state, but will be faster. After the load is finished, the database administrator will backup the tablespace to a different tape drive (an 8 mm drive) than the one being used by the load utility (3490 drive). The SAVECOUNT option is set to 100000 rows. For these tables, the load utility will be invoked with the following options:

- SAVECOUNT 100000
- Messages will be sent to different message file for each table.
- A different remote file will be specified for each table.
- STATISTICS YES AND INDEXES ALL
- COPY NO

Chapter 6. Logging

The aim of this chapter is to introduce the concept of logging in a relational database system. We examine why logs are used as well as how log files relate to SQL statements. This is followed by a detailed look at the different logging options available when setting up a DB2 database. Also, a discussion of how log files are used in recovery situations is included.

6.1 Overview

It would be impractical, from a performance point of view, for a database system to write out changes in data for each user whenever they were made. Most, if not all, relational database systems will batch these changes so that data modified by several users will only have to be written once. This allows the database system to determine when the most efficient time would be to write the changes to permanent storage. The flaw with this implementation is how to recover from a system crash. Since the changes were not written out immediately, there is no way to recover them. This is why the concept of database logs was developed. Logs are files which are used by DB2 to ensure the integrity of your database even when the system crashes due to some unforeseen problem, such as a power failure. To fully understand the purpose of logging, the concepts of unit of work and database transactions must first be explained.

6.1.1 Unit of Work

In order to ensure consistency of the data in a database, it is often necessary for applications to apply a number of changes all at once, or not at all. This is called the unit of work. A unit of work is a recoverable sequence of operations within an application process. The unit of work is the basic mechanism that an application uses to assure that it doesn′t introduce inconsistent data in a database. For example, when a bank executes a transfer of money from one account to another, it has to execute two operations at once:

- Subtract the amount from one account
- Add the same amount to the other account

After the application subtracts the amount from one account, the two accounts are inconsistent. They aren′t consistent again until the amount is added to the second account. When both steps are completed, a point of consistency is reached. The two changes can be applied to the database and made available to all other applications. If, for any reason, it is impossible to execute the second of the two operations, the first may, under no circumstance, be applied to the database alone. Instead, the database has to be returned to its previous state by undoing the first change. At any time, an application process has a single unit of work, but the life of an application process may involve many units of work.

6.1.2 Transaction

In relational databases, such as DB2, the unit of work is called the transaction. A transaction is a recoverable sequence of SQL operations within an application process. Any reading or writing to a database is done within a transaction. When an application makes changes to the database, the rows involving these changes are usually unavailable to other processes. They become available all at once or not at all when a transaction terminates.

Any application that successfully connects to a database automatically starts a transaction. The application must end the transaction by issuing an SQL `commit` or SQL an `rollback` statement. At this moment, a new transaction starts. The SQL `commit` statement tells the database manager to apply all database changes (inserts, updates and deletes) in the transaction to the database at once. The database manager thus becomes available to the other processes. The SQL `rollback` statement orders the database manager not to apply the changes but to return the affected rows to their original state just before the beginning of the transaction.

6.1.3 Write-Ahead-Logging

DB2 has implemented a Write-Ahead-Logging (WAL) scheme to ensure the integrity of your data. The basis for Write-Ahead-Logging is that when an SQL call is made which deletes, inserts, or updates any data in the database, the changes are first written to the log files. Then, at some later time, these changes are written to the data files. This will allow DB2 to write out the data when it deems it most efficient, depending on the current workload. When an SQL `commit` is issued, DB2 will ensure that all log file(s) required for that transaction to be replayed are written to disk. The database manager will not return to the caller (application) until the write of the log files has completed. In the case of a system failure, such as a power failure, the log files would be used to bring the database back to a consistent state. All committed transactions will be guaranteed to be redone, and all uncommitted transactions will be rolled back.

6.1.4 Use of Log Files by Multiple Transactions

All databases have log files associated with them. Log files are files in which the database manager synchronously records changes to a database issued by all applications accessing that database. These files play a role in recreating a consistent database after any incident which may have introduced an error. Log files have a predefined length. Therefore, when one log file gets filled, logging continues in another log file.

The following diagram will illustrate how multiple, concurrent transactions will simultaneously use multiple log files.

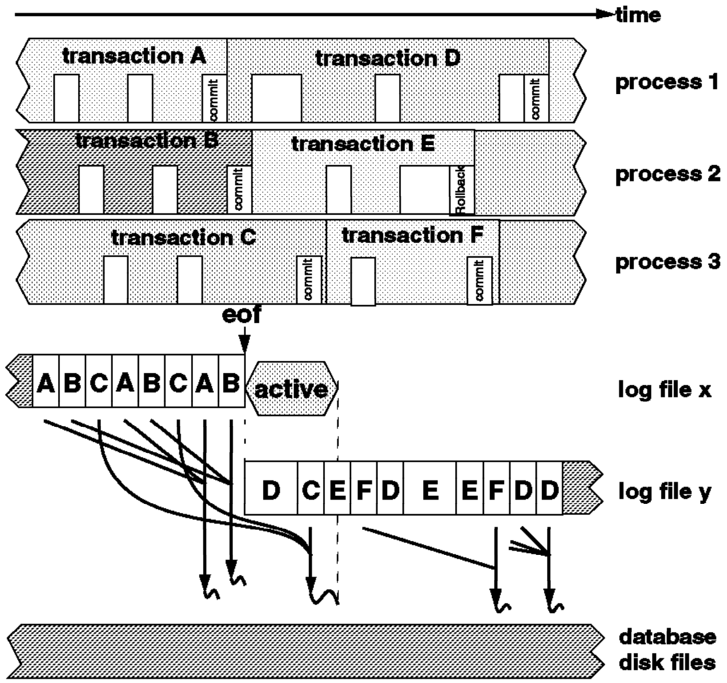


Figure 29. Transaction Log File Use

The top part of the diagram represents the evolution in time of three user processes (1 - 3) accessing the same database. The blank boxes represent database changes such as insert or update. The shaded boxes surrounding them show the life span of the different transactions (A - F). The lower-middle of the diagram, shows how the database changes are synchronously recorded in the log files (x, y). The letter in each box indicates the transaction to which the database change belongs. When an SQL COMMIT is issued, the log buffer containing the transaction is written to disk. This is represented by the arrows and the small wavy lines. Transaction E is never written to disk because it ends with an SQL ROLLBACK instruction. When log file x runs out of room to store the first database change of transaction D, the logging process switches to log file y. Log file x remains active until the writing of all changes of transaction C to the database disk files is complete. The period of time during which log file x remains active after the moment logging switched to log file y is represented by the hexagon.

6.2 Log Management Configuration Parameters

Figure 30 on page 134 shows the database configuration parameters associated with logging. The database administrator should have an understanding of these parameters and their effect.

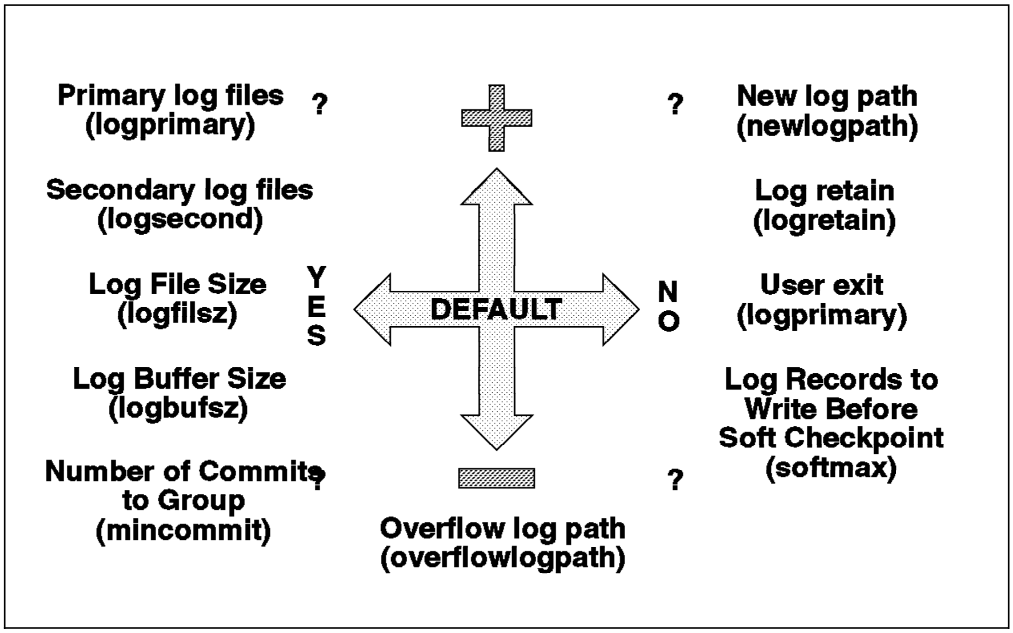


Figure 30. Logging Configuration Parameters

The configuration parameters shown in Figure 30 can impact logging for the database. The proper values for these parameters are highly dependent on the installation requirements; so care must be taken regarding “rules of thumb.” The following database configuration parameters are used by the logging process within DB2:

- LOGBUFSZ
- LOGFILSIZ
- LOGPRIMARY
- LOGSECOND
- NEWLOGPATH
- SOFTMAX
- LOGRETAIN
- USEREXIT
- MINCOMMIT
- OVERFLOWLOGPATH

Changes to any of these database configuration parameters will not take effect until all database connections have terminated. Then on the very first connection all configuration changes will be activated.

6.2.1 LOGBUFSZ - Log Buffer Size

The log buffer size parameter determines how much database shared memory will be allocated to buffer the log records before they are written out to disk. Buffering the log records will result in more efficient log file I/O since the log records will be written to disk less frequently, and more log records will be written at each time. Therefore, a large value can improve logging I/O for active databases, but the cost is memory. The value for this parameter will indicate the number of 4K pages, up to a maximum of 128-4 K pages; the default being 8-4 K pages. The units of 4 K are new for DB2/2 because previously it used 2 K pages and was limited to a maximum of 17-2 K pages. In DB2/6000 (AIX platform), the maximum was 32-4 K pages. In Version 2, both AIX and OS/2 have a maximum of 128-4 K pages.

6.2.2 LOGFILSIZ - Log File Size

This value determines the number of pages to be allocated when a log file is requested. Combined with LOGPRIMARY and LOGSECOND, this value determines the disk space required to support logging. Since all primary logs are allocated, even if not used, the value of this parameter can have major impact on the system disk utilization. The value specified must be balanced between the available disk space and the activity level of the database. A database that supports a high level of insert, update and/or delete transactions favors a larger log file size, but at the expense of disk space.

LOGFILSIZ is measured in units of 4 K pages. The size of the log files limits the number of log records that can be written before a new log files is required. In AIX, the default is 1000, with an upper limit of 16384; whereas the default on OS/2 is 250, with an upper limit of 4095.

6.2.3 LOGPRIMARY - Number of Primary Logs

This value represents the number of primary log files that will be allocated to support database logging. Regardless of the type of logging you are using, there will always be at least LOGPRIMARY log files allocated, each of LOGFILSIZ in size. The appropriate value is highly dependent on installation-specific requirements. For circular logging that is frequently using secondary logs, it may be necessary to increase this value. For log retention (archival) logging with highly active systems, it may be necessary to use a value greater than the default so that waiting for logs to be allocated is not experienced. Conversely, a database that is not accessed frequently may be properly supported by a smaller value in LOGPRIMARY, thus saving disk space.

The default number of primary log files is three and can be increased to a maximum of 128. DB2 has restricted the number of active log files to 128; therefore the sum of LOGPRIMARY and LOGSECOND must be less than or equal to 128, regardless of the type of logging being used.

6.2.4 LOGSECOND - Number of Secondary Logs

This parameter specifies the maximum number of secondary log files that can be created when needed by the system. This parameter is now used for both archival as well as circular logging. When the primary log files become full, the secondary log files of size LOGFILESZ are allocated one at a time, as needed. The default number of secondary log files is two, with a maximum of 126. The maximum is 126 because there is a minimum requirement of 2 primary log files. DB2 has restricted the number of active log files to 126; therefore the sum of LOGPRIMARY and LOGSECOND must be less than or equal to 128, regardless of the type of logging being used.

6.2.5 NEWLOGPATH - New Log Path

Database log files are, by default, written to the SQLOGDIR which is a subdirectory of the database directory. For recovery purposes, it will be beneficial for installations to place log files on a different physical disk than the database files. This parameter identifies the path for placement of log files. It is recommended that this parameter be set to direct log files to a disk that does not have high I/O requirements and does not contain the database itself. The path name must be a fully qualified path name which must already exist and cannot exceed 242 bytes.

6.2.6 SOFTMAX - Percentage of Records Reclaimed Before Soft Checkpoint

At the time of a database failure resulting from an event such as a power failure, there may have been changes to the database which:

- Have not been committed, but have updated the data in the buffer pool
- Have been committed, but have not been written from the buffer pool to the disk
- Have been committed and written from the buffer pool to the disk

When a database is restarted, the log files will be used to perform a crash recovery of the database which ensures that the database is in a consistent state (that is, all committed transactions are applied to the database, and all uncommitted transactions are not applied to the database).

To determine which records from the log file need to be applied to the database, the database manager uses a log control file which is periodically written to disk. Depending on the information in this log control file, the database manager may attempt to apply log records which have been committed and written from the buffer pool to disk. These log records have no impact on the database since the changes contained within them have already been written to disk. However, attempting to apply these log records introduces some overhead into the database restart phase.

The log control file is always written to disk when a log file is full, known as a hard checkpoint, or when the database is cleanly shut down. You may use this configuration parameter to take an additional soft checkpoint, which will:

- Write the log control file to disk
- Call an asynchronous page cleaner to flush the buffer pool to disk. For more information on asynchronous page cleaners, see 6.2.11, “NUM_IOCLEANERS - Asynchronous Page Cleaners” on page 137.

This value is a percentage of the LOGFILSIZ that determines if a log control file should be written more frequently than the default, which is to write the log control file when a log becomes full. The log control file is used during crash recovery to determine which log records are truly necessary to restore the database to a consistent state. If the log control file is not frequently written, crash recovery may apply log records that relate to events that have been committed and written to disk. While this will not corrupt the integrity of the database, it will increase the time for crash recovery to complete. Reducing this parameter will cause the log control file to be written as the log file fills past certain percentages. While this reduces crash recovery time, it will also increase overhead. Also, reducing this parameter may not be beneficial if the database supports large transactions with few commit points, or is configured with a large bufferpool and/or a small log file size.

6.2.7 LOGRETAIN - Recoverable Database

Setting this parameter to Yes indicates that log retention or archival logging is to be used. The log files become archived log files when full instead of being resued. If an installation wishes to enable rollforward recovery, this parameter (or USEREXIT) must be enabled. If this or the USEREXIT parameter is enabled, then the database is considered to be using archival logging. The first time either of these parameters are enabled, a full database backup will be required.

6.2.8 USEREXIT - Log Archiving

This parameter indicates whether a USEREXIT should be called when processing archive or retrieval requests for log files. Setting this value to Yes enables archival logging and rollforward recovery, regardless of the setting for LOGRETAIN. Installations that wish to automate the process of managing archive log files may benefit from utilizing this database parameter. The first time either the USEREXIT or LOGRETAIN parameters are enabled, a full database backup will be required.

6.2.9 MINCOMMIT - Number of Commits to Group

This value indicates that grouping of commits issued by multiple applications is to be attempted if the value is set to greater than 1. If the number of applications connected to the database is greater than or equal to this parameter, commits will not cause immediate physical writes of the log. The writes will be delayed one second or until the number of commits requested by all applications equals this value. Grouping commits can enhance the performance of a database servicing multiple connects with high change activity. Setting this value greater than 1 can introduce one-second commit waits for changed applications that do not execute concurrently with other changed applications.

6.2.10 OVERFLOWLOGPATH - Overflow Log Path

This parameter is used to specify an alternative log path to search for archived logs. Archive log files need to be brought back into LOGFILEPATH. If there is no additional space in LOGFILEPATH, OVERFLOWLOGPATH is used. The default is null; no path is set.

6.2.11 NUM_IOCLEANERS - Asynchronous Page Cleaners

A new class of database processes, called page cleaners, have been added to Version 2. Asynchronous page cleaners are not a logging configuration parameter. However, they do have an affect on which pages are written to disk, and that does concern logging. Figure 31 on page 138 shows how pages were written to disk in DB2/6000 Version 1 and how they are written to disk in Version 2.

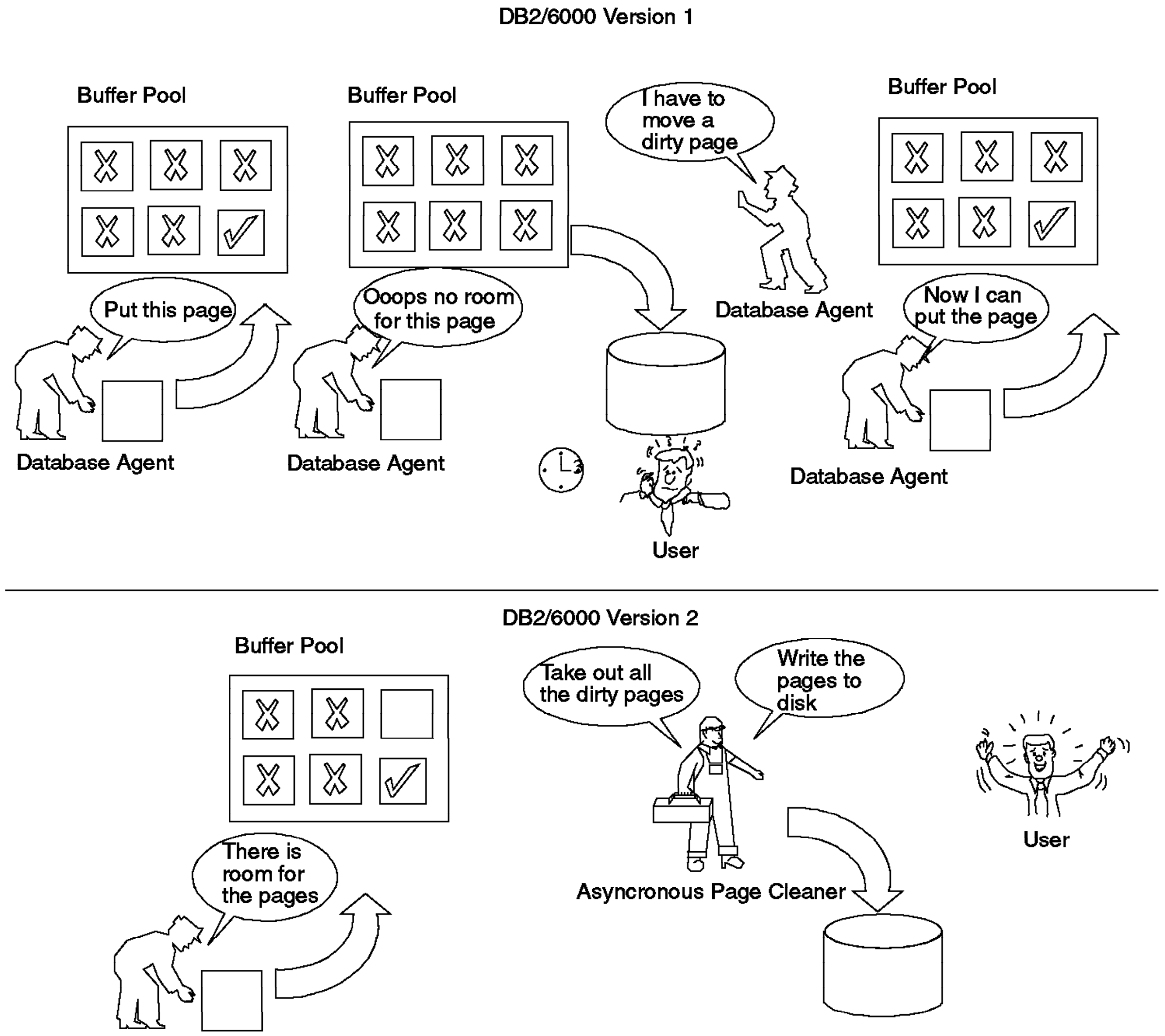


Figure 31. Asynchronous Page Cleaners

Page cleaners examine the database buffer pool, looking for pages which need to be written to disk. This allows user agent processes to use buffer pool pages without having to wait for I/O to complete processing.

DB2 moves pages of data from disk to the buffer pool in order to read and/or modify data. If a page has been modified, it should be written back to disk. With previous versions of DB2, this was done when a database agent needed some pages in the buffer pool, but discovered that the slots contained changed pages. So, the agent had to wait for an I/O operation to continue. DB2 Version 2 uses page cleaner agents to handle modified pages.

The purpose of the page cleaners is to write out most changed pages to disk so regular database agents can find empty slots and do not have to write pages out. This means that the agents will not have to wait for additional I/O, and the user transaction should execute faster.

The page cleaners are processes external to the database. They run in parallel with the database agents. They are activated when the number of written pages in the buffer pool is greater than MAXCHNGPGS, a database parameter. This value acts like a trigger for the page cleaners. The database manager also has other criteria to decide when to activate page cleaners.

The number of asynchronous page cleaners for a database is specified by the NUM_IOCLEANERS database configuration parameter.

6.2.12 Log File Information

DB2 uses a control file to determine the status of log files. The control file identifies the active log file with the "lowest" name. Figure 32 shows the log file information over time. The control file also identifies the name of the next log file to be used. This file is called the nextactive.

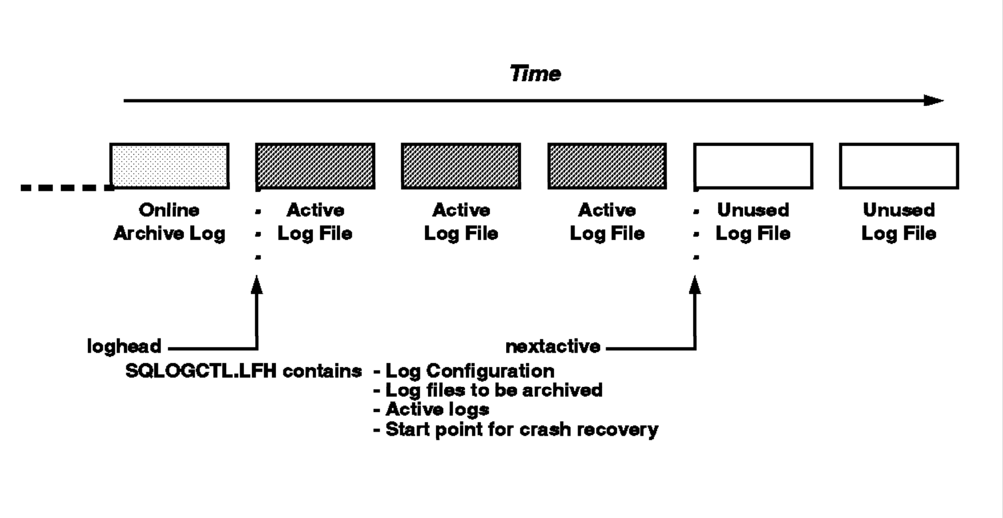


Figure 32. Log File Information

The values for loghead and nextactive can be valuable to the database administrator. They can be obtained via a `get database configuration` command. (The control file is identified for information purposes only. It is not in a readable format and should not be edited.)

Log files with names that are "less" than the loghead are archive files. They are not required for crash recovery and could be moved to a different media. Crash recovery is one of three types of recovery methods. For more information, see 7.5.1, "Methods of Recovery" on page 173 and 6.4.1, "Crash Recovery" on page 146.

The nextactive value, used in conjunction with loghead, can be used to determine the number of active logs currently allocated. If the number allocated becomes abnormally large, an application may not be committing on a timely basis.

6.2.13 Location of Log Files

This section discusses the naming and placement of log files. The naming convention for log files is as follows:

```
Sxxxxxxx.LOG
```

where xxxxxxx is a number starting from 0000000 through to 9999999. The numbers are assigned by the database manager.

By default, log files are located in SQLOGDIR, which is a subdirectory of the database directory. Figure 33 on page 140 shows the default location of log files when a database is created. It is not generally good practice to store log files on the same physical device as the database files for which they provide recovery support.

It is recommended to change the log path BEFORE using the database in order to direct the logs to a device that does not contain database files. It is recommended not to have a recovery strategy involving a change to the log path during the recovery process.

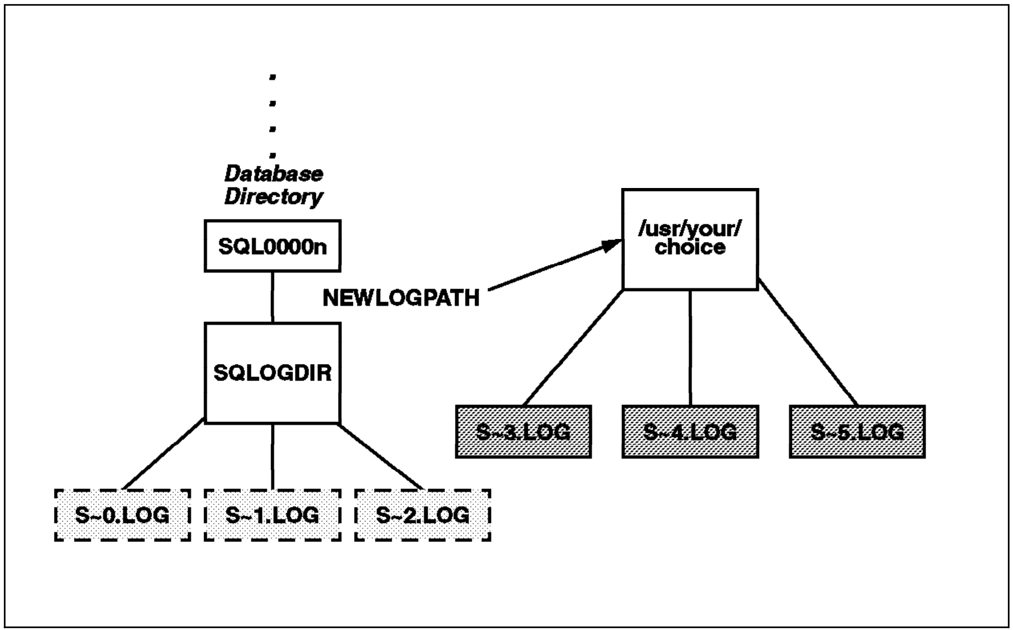


Figure 33. Location of Log Files

The location of log files currently in use is identified by the informational parameter, LOGPATH. The NEWLOGPATH database configuration parameter allows the database administrator to redirect logging support to a specified path. The new path does not become active until all connections to the database end and the database is in a consistent state. (A database may be in an inconsistent state due to an incomplete recovery process. This simply means that all units of work are not complete.) The information database configuration parameter, DATABASE_CONSISTENT, contains this status.

The situation illustrated in Figure 33 is not generally desirable. Here, a database was created and used before the log path was changed. Therefore, log files exist in the default directory, SQLOGDIR. At some point in time, while S0000002.LOG was the active log file, the database administrator updated the database configuration file to indicate a new logpath of /usr/your/choice in AIX or \usr\your\choice in OS/2.

Assume all applications completed their units of work and disconnected from the database before allocation of S0000003.LOG. The log path was changed to /usr/your/choice in AIX, or to \usr\your\choice in OS/2, at the time a new application connected to the database. The file S0000003.LOG was allocated in the new location. This means that the previous log files under SQLOGIDR must be manually moved to the new log path to be available if needed for recovery purposes. It is recommended to change the log path, if desired, before the first connection to the database is made.

6.3 Types of Logging in DB2

Two types of logging can occur in a DB2 database:

- Circular logging (the default)
- Archival Logging

6.3.1 Circular Logging

Circular logging will use the number of primary log files specified by the database configuration parameter, LOGPRIMARY. The necessary log information is for in-process transactions. The log files are used in sequence and can be reused when all units of work contained within it are committed or rolled back, and the committed changes are reflected on the disks supporting the database. Figure 34 shows the concept of circular logging.

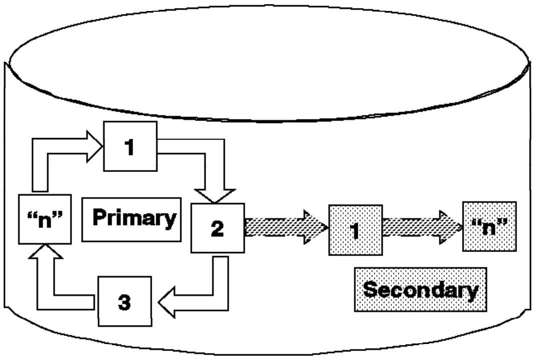


Figure 34. Circular Logging

6.3.1.1 Log Files Used with Circular Logging

Circular logging uses two types of log files:

- Primary log files
- Secondary log files

Primary log files are pre-allocated, while secondary log files are only allocated when necessary. If the database manager requests the next log in the sequence and it is not available for reuse, a secondary log file will be allocated. After it is full, the next primary log file is checked for reuse again. If it is still not available, another secondary log file is allocated. This process continues until the primary log file becomes available for reuse or the number of secondary log files permitted for allocation is exceeded.

Secondary log files are de-allocated once the database manager determines that they are no longer needed.

Primary log files are allocated when the database becomes active, while secondary log files are allocated as needed. Therefore, the database administrator may elect to use the primary log files for typical processing, but permit the allocation of secondary log files to permit periodic applications that have large units of work. For example, submission of an IMPORT utility with a large commit count may require the use of secondary log files. Supporting such applications via primary logs would waste space since all primary logs, whether used or not, are allocated when the database becomes active.

The number of primary and the maximum number of secondary log files are database parameters that may be tuned. The parameters are LOGPRIMARY and LOGSECOND and can be changed via the command line processor or via the Database Director. The maximum number of active log files (T) is the sum of LOGPRIMARY and LOGSECOND. Version 2 has increased the maximum number of active log files from 63 to a total of 128. If the number of active log files is equal to T and the last log file is filled up, all database changes from all applications accessing the corresponding database will fail with a log full error condition. This can be caused by two problems:

- The number and/or size of the log files are undersized for the type and number of transactions performed.
- A process has made a change in the database without completing it with a COMMIT or ROLLBACK.

If the number or size of the log files are undersized, the database has to be stopped and reconfigured. If secondary log files are frequently being allocated, you may be able to improve system performance by increasing the log file size, LOGFILSIZ, or by increasing the number of primary log files.

Circular logging provides support for crash and version/restore recovery, but does NOT support rollforward recovery.

6.3.1.2 Default Log Files

When a database is first created, the following characteristics are used:

- Circular logging is enabled (LOGRETAIN and USEREXIT are disabled).
- Log File Size is 1000 x 4 K pages on AIX and 250 x 4 K pages on OS/2.
- Three primary log files are allocated.
- LOGSECOND is set to 2.
- The log path is initialized to SQLOGDIR, which is a database subdirectory.

This means that a newly created (empty) database on DB2/2 will consume approximately 3 MB of disk space for the logs, whereas an empty DB2/6000 database will consume approximately 12 MB of disk space for the logs.

6.3.1.3 Circular Logging Changes in Version 2

The following enhancements have been made to circular logging in version 2:

- Total number of active logs increased from 63 to 128
- Log page size changed from 2 K pages to 4 K pages, DB2/2 only. DB2/6000 always had 4 K pages
- Max log buffer size changed from 32x2 K pages to 128x4 K pages on DB2/2 and from 32x4 K pages to 128x4 K pages on DB2/6000.
- Default log file size has changed to 1 MB in DB2/2 only.
- Secondary log files are de-allocated dynamically, either when a threshold limit has been reached or when a hard checkpoint is performed.

6.3.1.4 Recommendations

For databases which are not frequently accessed, set LOGPRIMARY to 2 to save disk space. Use secondary logs for databases that have periodic needs for large amounts of log space.

Circular logging should never be used in a production environment unless the database is query only and is never modified. The reason for this comes into play in a recovery scenario. Databases configured with circular logging are only recoverable to the point at which the backup was taken. All work done on the database since the backup was taken is lost when the database is restored. Note that the first time that either the LOGRETAIN and/or USEREXIT configuration parameters are enabled, the database is placed in a backup-pending state. This gives you a snapshot of the database at the current point in time, in case you need it for a recovery scenario.

6.3.2 Archival Logging

Archival logging is the log file management technique where log files are archived when they become inactive. Usually, the database configuration will permit several primary log files so that a log file being allocated is not immediately needed for logging. (Allocation is done ahead of the need for the file.) Figure 35 shows the concepts of archival logging.

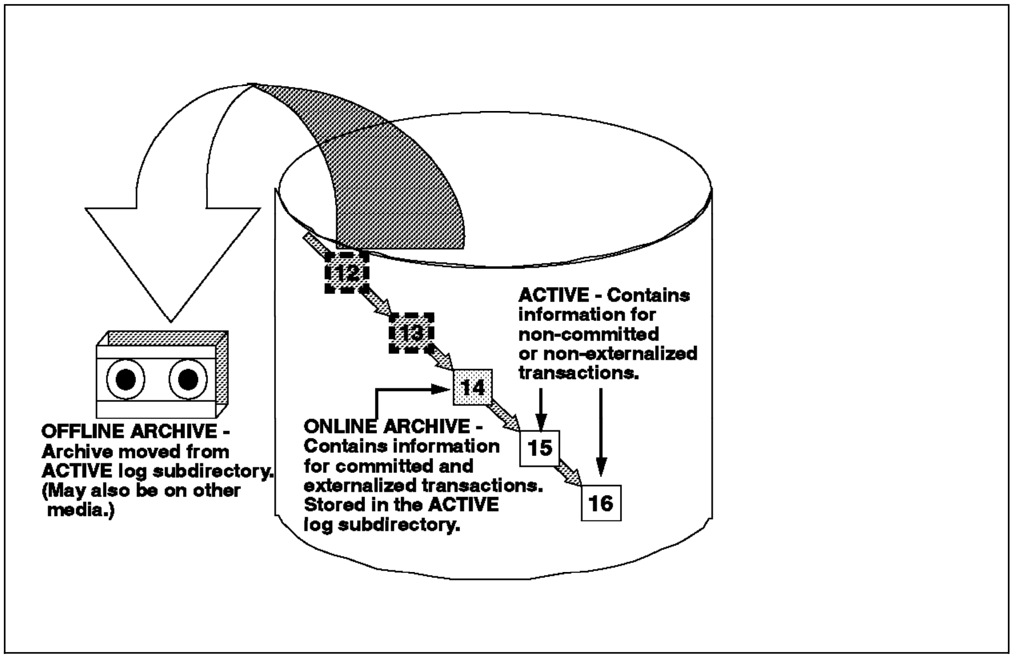

12
13
14
15
16
ACTIVE - Contains information for non-committed or non-externalized transactions.
ONLINE ARCHIVE - Contains information for committed and externalized transactions. Stored in the ACTIVE log subdirectory.
OFFLINE ARCHIVE - Archive moved from ACTIVE log subdirectory. (May also be on other media.)

Figure 35. Archival Logging

There are three types of log files associated with this method:

1. Active - (Indicated in Figure 35 numbers 15 and 16)

   These files contain information related to transactions that have not yet committed (or rolled back) work. They also contain information for transactions that have been committed but whose changes have not yet been written to the database files. (The changes could be in the buffer pool.)

   A log file is called active as long as it contains records of changes to the database which haven′t been committed or rolled back or as long as the committed changes haven′t been completely written to disk. An inactive log file can be either an online or an offline archived log file.

2. Online Archive (Figure 35 on page 143 number 14)

   These files contain information related to completed transactions no longer required for crash recovery protection. They are termed "online" because they reside in the same subdirectory as the active log files. By enabling the LOGRETAIN parameter, inactive log files are left in the log file directory, thus becoming online archived log files.

3. Offline Archive (Figure 35 on page 143 numbers 12 and 13)

   These files have been moved from the active log file subdirectory. The method of moving these files could be a manual process or a process invoked through a USEREXIT. Archived log files can be placed offline simply by moving them to another directory in the file system, by storing them on tape or by storing elsewhere, such as on an ADSTAR Distributed Storage Manager (ADSM) server.

   ADSM is a client/server archiving product. It allows client system files to be archived on and retrieved from host storage media, where the host can be MVS, AIX or OS/2. In this scenario, the DB2 server is a client to the ADSM server, and must be configured as such. Refer to the *DB2 Administration Guide* for details regarding configuring ADSM with DB2.

Note: When using archival (log retention) logging, DB2 will truncate and close the last log file written to free-up space when the last application disconnects from the database. This is a positive feature when the database is to be inactive for some period of time. However, if an installation has a low level of activity and there are short periods where no application will be connected to the database, it will be costly to truncate the last active log and then re-allocate the primary log files when the application connects.

The database administrator should consider using an application that connects to the database and simply waits for a request to disconnect. This "dummy" application will keep the database active and prevent log file truncation. (An additional benefit of such an application would be that memory allocated at first connect and released when the database becomes inactive would be retained. This can enhance the performance of databases experiencing periods of inactivity.

Two configuration parameters allow you to configure a database for archival logging:

- LOGRETAIN
- USEREXIT

When the LOGRETAIN database configuration parameter is enabled, log files are not deleted when they become inactive. When the USEREXIT database configuration parameter is enabled, the database manager will call the db2uexit program each time a log file is no longer needed for log writes, for example it is full. A number of parameters are passed to the program, such as the database name and path of the log file. These parameters allow the program to archive the log file. The customer must write this program and tailor it to the needs of their environment. Three sample C source db2uexit programs are included in the DB2 product. If USEREXIT is enabled, the log file will not be deleted until it is no longer needed by any transaction, for example to complete a rollback.

Conversely, if USEREXIT is disabled, the log file will never be removed and will continue to be stored in the directory used for logging.

6.3.2.1 Advantages of Archival Logging

Archival logging, even though not the default logging method, is the only method which will allow you to perform rollforward recovery. In Version 2, to restore a tablespace, you must perform the rollforward operation; so you must have archival logging as the form of logging for the database. The advantages can be summarized as follows:

- Ability to perform online backups
- Ability to perform tablespace-level backups
- Ability to recover the database to any point in time past the end of the backup

6.3.2.2 Archival Logging Changes in Version 2

The following enhancements have been made to archival logging in version 2:

- Total number of active logs increased from 63 to 128.
- Log page size changed from 2 K pages to 4 K pages; DB2/2 only, DB2/6000 always had 4 K pages.
- Max log buffer size changed from 32x4 K pages to 128x4 K pages on DB2/6000.
- Default log file size has changed to 1 MB, DB2/2 only. The default in AIX is 4 MB.
- LOGSECOND configuration parameter is now used to allocate secondary logs in cases when the all of the log primary files are filled. This is particularly useful if a log full condition is encountered during crash recovery as this parameter can be increased, and the crash recovery can be restarted.

6.3.2.3 Recommendations

All production databases should be configured for archival logging. Non-production databases may be configured with circular logging so that log file management does not have to be done.

The log path should be set so that it does not reside on the same physical disk as any of the tablespaces. This will allow the operation system to write to both the logs and data concurrently.

The USEREXIT database configuration parameter should be enabled so that inactive log files are deleted from your current log path, allowing disk space for new active log files to be allocated.

The two most important configuration parameters used by archival logging are LOGFILSIZ and LOGPRIMARY. Unfortunately, there are no hard and set rules for recommending values for these database configuration parameters. You will have to run experiments to determine what is optimal for your environment and workload. The log file size will directly affect how often the log fills and, therefore, is archived. The number of primary log files is not as important now that the LOGSECOND database configuration parameter is supported, which was not the case in Version 1.

6.4 Log File Usage

Log files are written to during normal processing; however, they are only ever read in three situations:

1. Rollback
2. Crash Recovery
3. Rollforward Recovery

The SQL `rollback` statement uses the log files to terminate a unit of work and back out the database changes that were made by that unit of work. Crash recovery and rollforward recovery are covered in the following sections.

6.4.1 Crash Recovery

In the event of certain kinds of failure (disk crash, power outage), an operation called crash recovery is needed to bring the database back to a consistent, usable state. Crash recovery, also known as database restart, consists of two consecutive phases. During the first phase of crash recovery, all transactions are reapplied to the database, regardless of whether they were committed or not. This phase completes when the end of the active log files is reached. The second phase of crash recovery is to roll back all uncommitted transactions. Both phases are required to complete successfully before the database is considered to be “transaction consistent.”

When a database is created, the default setting for the database configuration parameter to enable crash recovery is ON. This parameter is called AUTORESTART. In the event of a system crash, the first person to connect to the database will trigger the database manager to be restarted. This connect may take quite a while as the logs will have to be replayed and uncommitted transactions rolled back. It is possible to disable this feature by setting AUTORESTART to OFF

6.4.2 Rollforward Recovery

`Rollforward` applies transactions recorded in the database log files. The command is invoked after a database or tablespace backup has been restored or if any tablespaces have been taken offline by the database manager due to a media error.

If an I/O error is encountered while trying to read from or write to disk, a tablespace in which the page resides is disabled and placed in a “roll forward pending” state. It is possible that a roll forward to the end of the logs will clear the state. If the pending state cannot be cleared with just a roll forward of the tablespace, a restore followed by a roll forward is required. For a discussion on restore of a database or tablespace, see Chapter 7, “Backup and Restore” on page 151. In either case, the `rollforward` command is issued.

Restore is the first phase of a complete rollforward recovery of a database or tablespace. After a successful database restore, a database that was configured for rollforward recovery at the time of a backup was taken enters a rollforward pending state. It is not usable until the `rollforward` command has been run successfully. If the restore used a tablespace-level backup, the tablespaces restored enter a rollforward pending state.

When the rollforward database command is issued, if the database is in a rollforward pending state, the database is rolled forward. If the database is not in a rollforward pending state, all tablespaces in the database in the rollforward pending state are processed.

Another database restore is not allowed when the rollforward process is executing. Note that if you restore from a full offline database backup image, you can bypass the rollforward pending state during the recovery process. The restore database command gives you the option to use the restored database immediately without rolling the database forward.

You cannot bypass the rollforward phase when recovering at the tablespace level or if you restore from a backup image that was created using the ONLINE option of the backup database command.

6.4.2.1 Log File Considerations with Roll Forward

If enabling an existing database for rollforward recovery, change the number of primary log files to the sum of the number of primary log files and secondary log files + 1. This recommendation assumes the prior configuration was determined to meet the needs of the largest unit of work in the system. More information will be logged for LONG VARCHAR fields and LOB data in a database that has been enabled for rollforward recovery.

If a tablespace is being rolled forward and the database is configured with USEREXITs enabled, the log files will have to be manually copied from the archive source. Only a database roll forward will call the USEREXIT to retrieve log files. This is not yet possible with a tablespace-level roll forward.

6.4.2.2 Syntax of Rollforward Database

The syntax for the command is as follows:

```
►►──ROLLFORWARD──DATABASE──database-alias──────────────────────────►

►──┬────────────────────────────────────┬──────────────────────────►
   └─USER──username──┬─────────────────┬┘
                     └─USING──password─┘

►──┬────────────────────────────────────┬──┬──────────────────────┬──►
   ├─TO──┬─isotime──────┬──┬──────────┬─┤  └─TABLESPACE ONLINE─┘
   │     └─END OF LOGS─┘   └─AND STOP─┘ │
   ├─STOP───────────────────────────────┤
   └─QUERY STATUS───────────────────────┘

►──┬──────────────────────────────────┬──────────────────────────►◄
   └─OVERFLOW LOG PATH──log-directory─┘
```

The key parameters of the rollforward command are as follows:

database-alias database name which will be rolled forward.

username The authorized UserID under which the database will be rolled forward.

password The password for the supplied username; if one is not supplied the user will be prompted to enter it.

TO isotime The point in time to which all committed transactions are rolled forward. This parameter is only valid for full database restore. The value is specified as a timestamp in the following format:

```
yyyy-mm-dd-hh.mm.ss.nnnnn
```

(year, month, day, hour, minute, seconds, microseconds) This timestamp is in Universal Coordinated Time (UCT), which is Greenwich Mean Time.

TO END OF LOGS This will cause the roll forward to process as many log files as it can locate in the current log path directory. If the database is configured with USEREXITs enabled and this is a full database roll forward, then archived logs will be automatically retrieved. Otherwise, the log files will have to be manually copied into the current log path or their path specified on the OVERFLOW LOG PATH parameter.

STOP / AND STOP This will indicate that you have processed all of the log files, and you want to make the database consistent. Be very careful when specifying one of these parameters as once they are issued, all uncommitted transactions are rolled back, and the database is made transaction consistent.

QUERY STATUS List the log files which have been rolled forward, the next archive log file required, the timestamp (UCT format) of the last committed transaction since rollforward processing began. QUERY STATUS is the default value of the TO and STOP phases which are omitted from the `rollforward` command. It is recommended that you issue the `rollforward` command with the QUERY STATUS parameter before using the STOP parameter to ensure that you have rolled forward to the correct point in time.

TABLESPACE ONLINE
Indicates that the roll forward will be done at the tablespace level. This will also allow other agents to connect to the database and access all tablespaces which are accessible and not being rolled forward.

OVERFLOW LOG PATH
Specifies an alternative log path to search for archived log files. This is particularly useful when restoring a database to a different system as the original log path may not exist. It is also useful for tablespace-level restore because the USEREXIT will not be called to retrieve archived log files.

6.4.3 How Far to Roll Forward

The database administrator can clear a rollforward pending condition by issuing the `rollforward` command. The point in time to which the rollforward stage processed is also controlled by the administrator. There are a number of considerations for determining how far to roll forward the log files. These considerations include:

- End of logs (typical)

The end of logs means usually means the end of the current log path. Other logs, however, may need to be moved into the path. End of logs is a requirement for any tablespace recovery strategy.

- Online backup requires a roll forward past the end of the backup.

  The integrity of the database must be protected. Therefore, the earliest point in time at which the rollforward stage can end is the end of the online backup image.

- Point in time

  The parameter ISOTIME can be coded on the `rollforward` command to identify a particular point in time up to which the logs should be applied. This time is specified as the iso format of the Universal Coordinated Time (UCT).

  The Universal Coordinated Time is used in log records so that the database manager does not have a recovery dependency regarding daylight savings time or other local time anomalies. However, this introduces a degree of complexity for the database administrator. While backup images are identified via timestamps reflecting local time, rolling forward (since it applies to logs) must designate times in UCT format. The format is: yyy-mm-dd-hh.mm.ss.

In many cases, the point of time for recovery will be the most current one possible. Therefore, the end of logs option will be commonly used. The database administrator or instance administrator can stop the rollforward process and allow access to the database.

The AND STOP is a necessary parameter to permit the database manager to roll back any transactions that are not completed after applying the log records to the indicated point. This is true even if END OF LOGS is utilized. Otherwise, the database will remain in rollforward pending status.

Chapter 7. Backup and Restore

In this chapter we will describe the backup and restore utilities which are used to safeguard and recover databases in the event of a failure. We will discuss database and tablespace backup and restore considerations and how a customer might implement a backup/restore strategy.

7.1 Overview

In DB2 Version 1, backup and restore were supported at the full database level only. With databases growing to several gigabytes in size or more, the maintenance window available may not be sufficient to allow for a full database backup. In Version 2, backup and restore can be done at a finer level of granularity, namely, the tablespace.

The ability to record the recovery history of a database was also a desirable feature in Version 1. A recovery history file is now added in Version 2 to track backup/recovery/load activities performed on the database. This is done through an interface that allows the user to query and manage the file.

This chapter is outlined as follows:

- Discussion of the Version 1 and Version 2 process models
- A detailed look at the backup and restore utility at both the database and tablespace level
- Guidelines for a backup and recovery strategy
- Some backup and restore scenarios
- Types of failures that may occur and the recommended action for recovery
- Performance Issues
- Recovery history file
- The customer scenario of the telephone company

7.1.1 Review of Version 1 Process Models

As a review, we will look at the different process models that were used for backup/restore in Version 1. There were three distinct models used:

1. DB2/2 Version 1 Backup/Restore Process Model
2. DB2/2 Version 1 Backup/Restore User Exit Process Model
3. DB2/6000 Version 1 Backup/Restore Process Model

7.1.1.1 DB2/2 Version 1 Backup/Restore Process Model

In Version 1 of DB2 for OS/2 (DB2/2), each database was restricted to residing on a single partition. As such, the OS/2 backup and restore utilities were used to perform the database level backup and restore. These utilities required the caller to back up to a partition other than the one on which the database resided. As well, the target output was restricted to OS/2 support devices; namely disk and diskette. Figure Figure 36 on page 152, illustrates how the backup utility functioned.

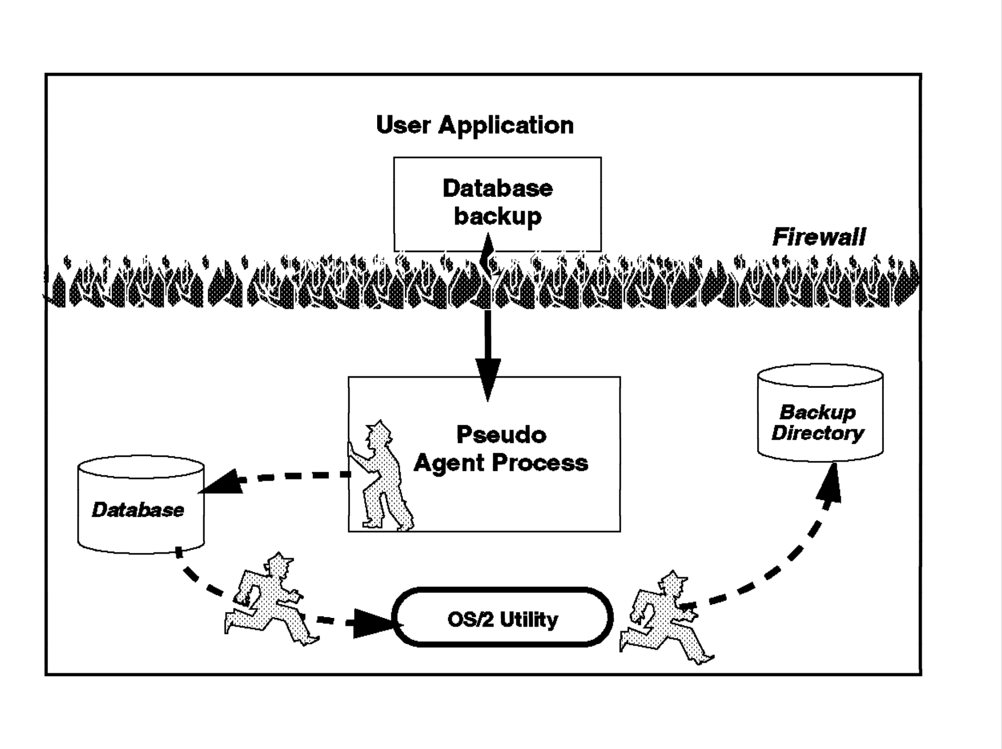


Figure 36. DB2/2 Version 1 Backup/Restore Processing

The main points illustrated in Figure 36 are the following:

- DB2/2 Version 1 used a different backup/restore utility than is used in DB2 Version 2.
- Only disk or diskette was supported for backup and restore.
- This model is still supported in DB2 for OS/2 Version 2.

7.1.1.2 DB2/2 Version 1 User Exit Process Model

OS/2 versions 1.0 - 3.0 did not support tape drives as a logical device. Therefore, the only way to backup/restore from a tape drive was through the use of a user exit calling a third party vendor. This restriction is still enforced with Version 2 and support for user exits is still available. Figure Figure 37 on page 153, illustrates how user exits were supported.

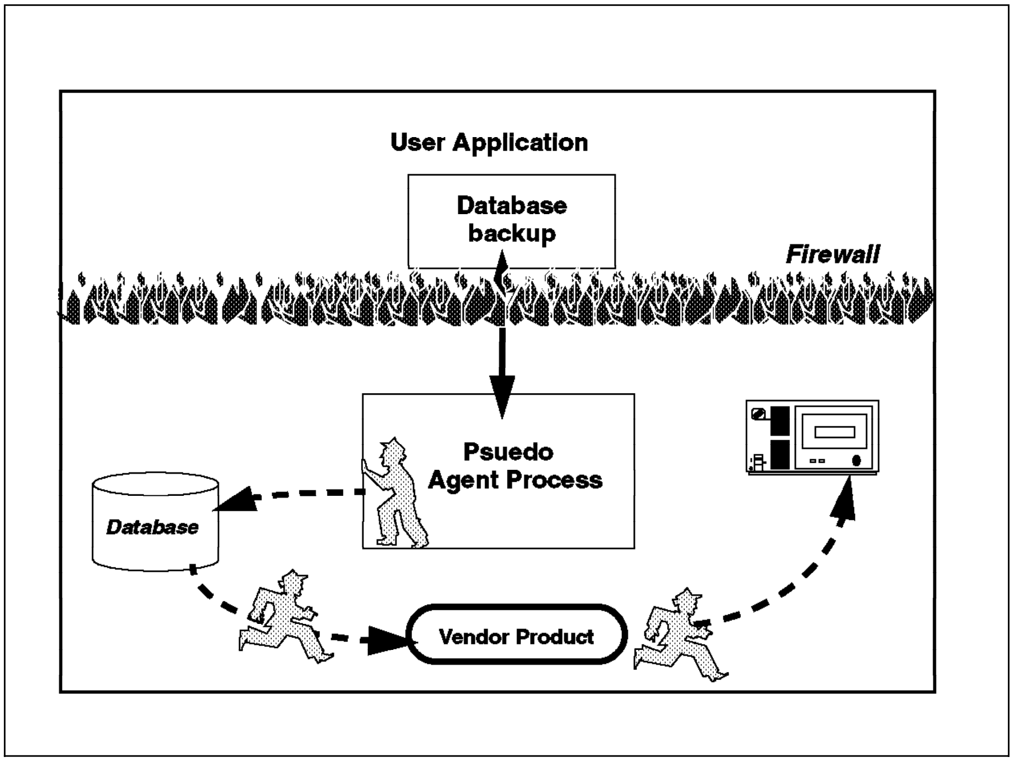


Figure 37. DB2/2 Version 1 User Exit Support

The main points in Figure 37 are the following:

- A third party vendor product was necessary to provide user exit support using a tape device in DB2/2 Version 1.
- This support was available in Version 1.x and still exists Version 2.x for OS/2 through the use of APIs. The target-area in the backup/restore command parameters will be :0.
- You had to either quiesce or pause the database before starting the backup process.

7.1.1.3 DB2/6000 Version 1 Process Model

In Version 1, when a database was backed up or restored, only one database process was involved. The data pages were read into a single buffer and written out serially when the buffer was full. In doing this, the process could become I/O bound, leaving the CPU idle. The performance of the backup and restore utilities in Version 2 is improved by the ability to use multiple buffers and I/O streams. In Figure 38 on page 154, you can see the process activity for the backup utility that was used in Version 1 of DB2/6000.

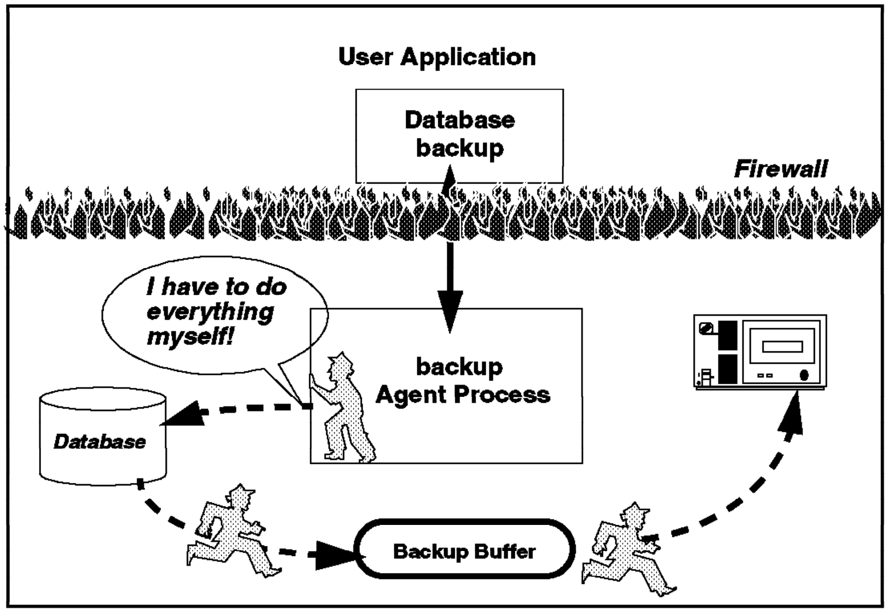


Figure 38. DB2/6000 Version 1 Backup/Restore Processing

7.1.1.4 DB2 for AIX Version 2 Process Model

Version 2 has increased in flexibility and usability by allowing the caller to specify the size of the internal buffer to use, the number of internal buffers, as well as the number of devices to be read from or written to. The optimal number of buffers and media I/O devices to use will depend on the environmental setup for each individual database. The system administrator can tune the performance by changing the input parameters. In Figure 39 on page 155, you can see the process activity of a backup operation using multiple buffers and media I/O processes.

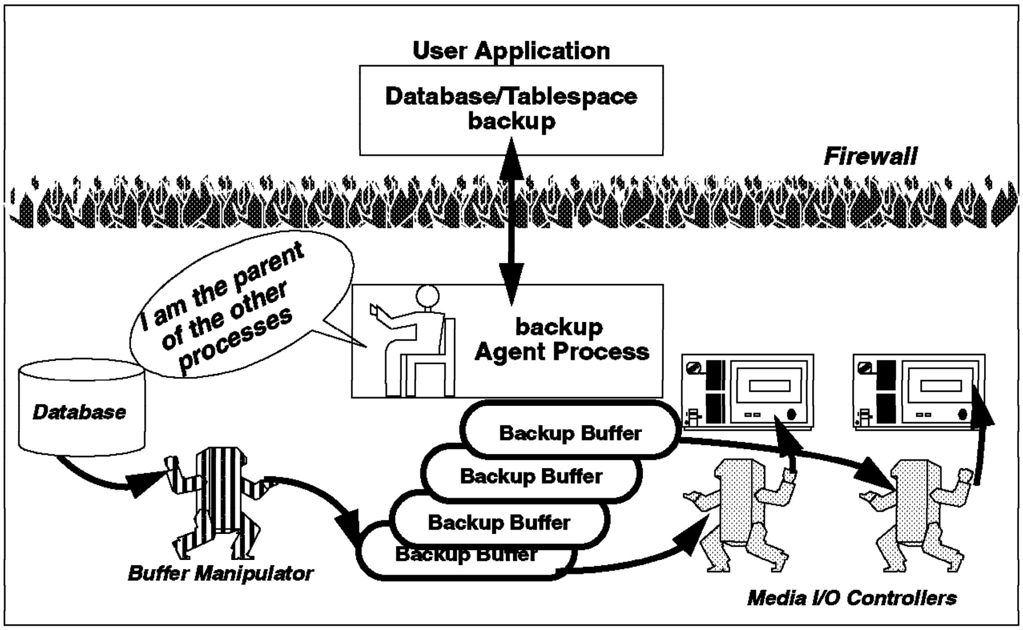


Figure 39. DB2 Version 2 Backup/Restore Processing

To increase the performance in backup and restore in Version 2, the portion of the process that manipulates the backup and restore buffers has been removed from the agent process and is now a separate process. This separation allows the backup or restore agent (parent process) to handle any error conditions that might be returned from the media I/O controllers without impacting the speed at which we read the database data. The main purpose of the buffer manipulator is to either read data from the database and place it in the media buffers or to take data from the media buffers and put it back in the database. All error processing and communication with the clients will be done by the parent agent.

During a restore procedure, you also have the ability to select multiple buffers to improve the performance of the restore process. The multiple internal buffers may be filled with data from the backup media. You may specify the number of pages to use for this restore buffer when invoking the restore database utility. If you do not, the buffer will be allocated based on the database manager configuration parameter, RESTBUFSZ. If there is not enough memory available to allocate the buffer, an error will be returned. If a database was created with a previous version of DB2 and the database has not been migrated, you must migrate the database before performing a backup. See Appendix A, “Database Migration” on page 207 for more information on migrating a Version 1 database.

7.2 DB2 Backup and Restore Considerations

Version 2 of DB2 allows you the option of backing up the entire database or individual tablespaces or groups of tablespaces. To make a informed choice as to which method to use, a full understanding of the backup/restore operation is necessary. Figure 40 on page 156 shows some of the considerations that must be determined before running the utility.

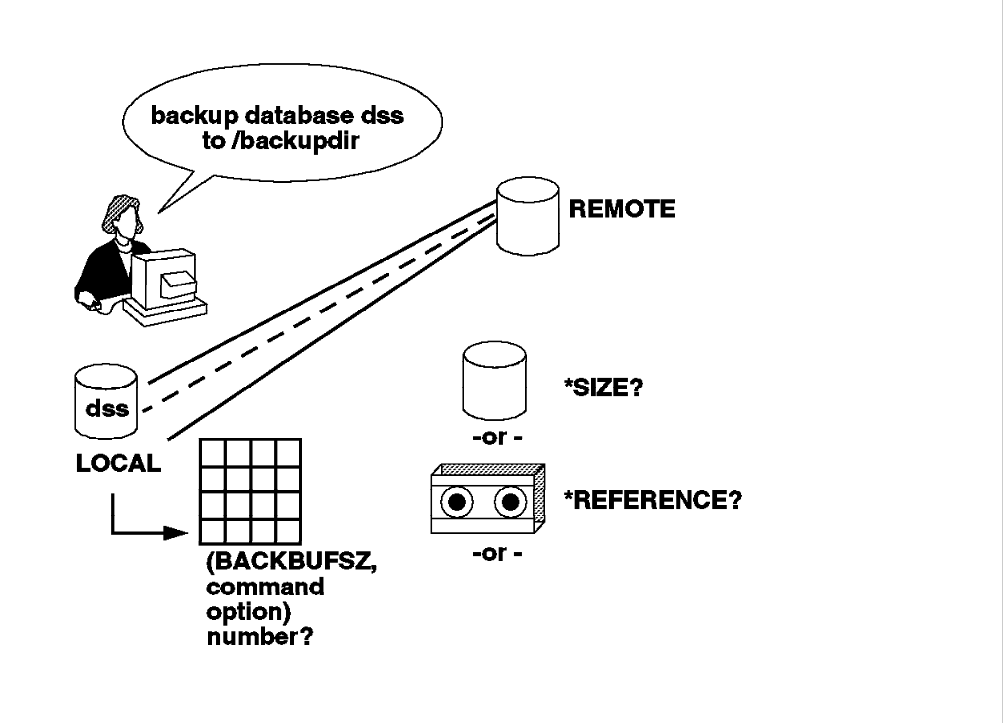

backup database dss to /backupdir
REMOTE
dss
LOCAL
*SIZE?
-or -
*REFERENCE?
-or -
(BACKBUFSZ, command option) number?

Figure 40. Backup Utility Considerations

A full offline database backup provides you with a complete snapshot of the data at a fixed point in time. This level of backup is a requirement for disaster recovery and should be an essential part of any backup/restore strategy.

Some of the considerations should include:

- You must have SYSADM, SYSCTRL or SYSMAINT authority to use the `backup database` command.
- You must start the database manager (db2start) before running the `backup database` command or API. However, when using the Database Director, you do not need to explicitly start the database manager.
- If you are using the command or API from the Database Director, you must specify a database alias name, not the database name itself.
- You can do a backup while the database is either offline or online. The default is offline.
  - If the backup is performed offline, only the backup task can be connected to the database. The stored data must be consistent. An offline backup can be used either as a restore only type of recovery or a restore followed by the roll forward phase.

    The implication of an offline backup is that the rest of your organization cannot connect to the database while the backup task is running.
  - If the backup is performed online, other applications or processes can continue to connect to it while the backup task is running.

    Online backups are supported only if rollforward recovery is enabled. The backup will not be valid for recovery purposes, if you do not retain the log files written while the backup was taken. While the online backup operation is running, changes can be performed on the tables. The data, when restored from the backup files, is not consistent until the logs are applied during roll forward recovery.

An offline backup will acquire an exclusive connection to the database, failing if anyone else is already connected. Whereas an online backup will merely acquire a shared connection, permitting other shared connections to exist.

- The database may be local or remote. The backup image remains on the database server unless a storage management product such as ADSTAR Distributed Storage Manager (ADSM) is used.
- You can back up a database or tablespace to a fixed disk, a tape or a location managed by the ADSM utility or another vendor. If using a utility from a third party vendor, they must provide the shared library.
- If you change a database configuration parameter to enable roll forward recovery (either LOGRETAIN or USEREXIT), you must take an offline backup of the database before it is usable.
- If your database is enabled for roll forward recovery and you are using a tape system that does not support the ability to uniquely reference a backup image, it is recommended that you do not keep multiple backup copies of the same database on the same tape.
- Multiple files may be created to contain the backed up data from the database or tablespace.
- You may specify multiple target areas (directory or tape devices) if you wish to take advantage of parallelism. For more information, see the *DB2 Administration Guide and Reference*.
- During the backup procedure, an internal buffer is filled with data to be backed up. When this buffer becomes full, the data is copied to the backup medium. You have the ability in Version 2, to optionally select to use multiple buffers and I/O streams to improve the performance of the backup procedure. The number of buffers you allocate should be at least twice as many as the number of I/O devices you are using. For more information on backup performance, see 7.7, “Performance Issues” on page 176.
- You may specify the number of pages to use for the backup buffer(s) when you invoke the `backup database` command. The minimum number of pages is 16. If you do not specify the number of pages, each buffer will be allocated based on the database manager configuration parameter, BACKBUFSZ. If there is not enough memory available to allocate to the buffer, an error will be returned.
- If a system crash occurs during a critical stage of backing up the database, you cannot successfully connect to the database until you re-issue the `backup database` command.
- If you are using the `backup database` command for concurrent backup processes to tape, ensure that the processes do not target the same tape.

7.2.1 Understanding the Backup Command

The syntax for the utility is as follows:

```
►►──BACKUP DATABASE──database-alias──┬──────────────────────────────────┬──────────►
                                     └─USER──username──┬──────────────────┬┘
                                                       └─USING──password─┘

►──┬──────────────────────────────────────┬──┬──────────┬──────────────────────────►
   │             ┌─,──────────────────┐   │  └─ONLINE─┘
   └─TABLESPACE──▼─tablespace-name────┴───┘

►──┬─────────────────────────────────────────────────────┬─────────────────────────►
   ├─USE ADSM──┬─────────────────────────────────┬─────┤
   │           └─OPEN──num-sessions──SESSIONS─┘        │
   │      ┌─,────────────┐                              │
   ├─TO──▼─target-area─┴───────────────────────────────┤
   └─LOAD──library-name──┬───────────────────────────────────┬┘
                         └─OPEN──num-sessions──SESSIONS─┘

►──┬───────────────────────────────┬──┬─────────────────────────┬──┬─────────────────────┬───────►◄
   └─WITH──num-buffers──BUFFERS─┘  └─BUFFER──buffer-size─┘  └─WITHOUT PROMPTING─┘
```

The following describes the parameters to the backup command:

DATABASE database-alias
Specifies the alias name of the database to back up. When using the command, API or task under the Database Director, you must specify a database alias name, not the database name itself.

USER username
Identifies the user name under which the backup is processed.

USING password
The password used to authenticate the user name. If the password is omitted, the user is prompted to enter it.

TABLESPACE tablespace-name
List of one or more tablespaces within the database to be backed up.

ONLINE
Specifies online processing, the default is offline. If a database connection exists and the default option of offline is attempted when backing up, the offline process will be stopped. Online processing requires that the database is enabled for rollforward processing.

USE ADSM OPEN num-sessions SESSIONS
Specifies that ADSM will be the backup target for this backup and and “num-sessions” ADSM sessions will be used throughout the backup. The necessary ADSM environment variables must be previously set up and the DB2 Server registered as an ADSM Client with the ADSM Server. See the *DB2 Administration Guide* for more details.

TO target-area
Specifies where the target placement for the backup will be. This can be a directory or tape device name. If backing up to a directory, the full path name must be provided. You may specify multiple target areas (directory or tape devices) if you want to take advantage of parallel backup. If more than one target is specified, the first one specified will be opened and contain the media header and special files. The remaining targets are opened and then

used in parallel during the backup. This may be useful for tablespace backups of large tables (for example LOBs). If no parameter is issued, the backup will be placed under the current directory (for example: /u/instance). The target-area must reside on the database server if it is a directory.

LOAD library-name OPEN num-sessions SESSIONS

Specifies that a third party vendor product will be used as the target for this backup and that “num-sessions” will be used.

WITH num-buffers BUFFERS

The number of buffers to be used.

BUFFER buffer-size

The size in pages of the buffer used for the appropriate process. The minimum value for this 16 pages. The default value is 1024 pages. If you specify a buffersize of 0, the value in the database manager configuration for this parameter will be used. This parameter is also discussed in 7.7, “Performance Issues” on page 176.

WITHOUT PROMPTING

Perform the backup without prompting for new media. If any other type of warning occurs or all the backup target devices fill, then an error is returned to the caller and the backup fails.

7.2.2 The Backup File

Since AIX and OS/2 do not share identical conventions for naming files, DB2 could not make the backup file names identical across platforms. However, the format of each file is easy to understand. Figure 41 shows the backup file format for both OS/2 and AIX.

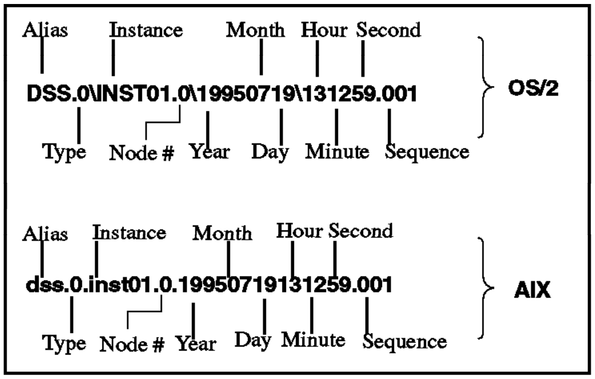


Figure 41. Backup File Format in DB2

Figure 41 shows a database backup of database, dss, in inst01 taken on July 19, 1995 at 13:12:59. There are three possible file types:

1. Type 0 is for full database
2. Type 3 is for a tablespace backup
3. Type 4 is for a copy for a table load

The number after the instance name is reserved for the node number.

Tape images for backups are not named, but contain the same information in the backup header for verification purposes.

Backup history provides key information in an easy to use format and is kept in the recovery history file. The recovery history file is updated automatically with summary information whenever you carry out a backup or restore of a full database or tablespace. The contents of the history file is sufficient to support management of the backup images. However, there may be cases where knowledge of the naming convention used by DB2 regarding backup files could be useful. One such case would involve a damaged history file. The recovery history file is discussed in more detail in 7.8, "Recovery History File" on page 177.

7.3 DB2 Restore Utility

When restoring a database, consider the following:

- You must have SYSADM, SYSCTRL or SYSMAINT authority to restore to an exiting database from a full database or tablespace backup. To restore to a new database, you must have SYSADM or SYSCTRL authority.
- The database manager must be started before restoring a database.
- You can only use the `restore database` command if the database or tablespace has been backed up with the `backup database` command.
- The database to which you restore the data may be the same one as the data was originally backup up from, or it may be different. You may restore the data to a new or an existing database.
- You can choose at the time of the restore which type of restore is to be performed. You can select from the following types:
  1. A full restore of everything from the backup image.
  2. A tablespace restore (using a backup image that only includes tablespaces).
  3. A restore of only the recovery history file from the backup image.
- The database may be local or remote.
- The `restore database` command can be used the ADSM utility.
- Another vendor product may also be used if that product was used to store the original backup image.
- A database restore requires an exclusive connection. That is, no applications can be running against the database when the task is started. Once it starts, it prevents other applications from accessing the database until the restore is completed.
- The size and number of the buffers(s) used to support the restore can be specified as a command option. If the hardware configuration has multiple I/O controllers available, supporting I/O with multiple buffers can improve the performance of the restore.
- During the restore procedure, you have the ability to optionally select to use multiple buffers to improve the performance of the restore operation. The multiple internal buffers may be filled with data from the backup media. The

number of buffers you allocate should be at least twice as many of the number of I/O devices you are using.

- You may specify the number of pages to use for each restore buffer when you invoke the restore database command. The minimum number of pages is 16. If you do not specify the number of pages, each buffer will be allocated based on the database manager configuration parameter, RESTBUFSZ. If there is not enough memory available to allocate the buffer, an error will be returned.
- The backup copy of the database or tablespace to be used by the restore database command can be located on a fixed disk, tape or location managed by ADSM or another vendor. If using a third party vendor, they must supply the shared libraries.
- Once the restore database command starts for a database backup, the database is not usable until the command completes successfully. It is also possible that a roll forward will be required before access is permitted.
- If a system failure occurs during any stage of restoring a database, you cannot connect to the database until you reissue the restore database command and successfully complete the restore.
- If a Version 1 backup copy of the database has not been migrated, the restore database command migrates the database after the database is restored. The backup copy of the database is not affected. If the restore operation cannot successfully call the migration utility, you will receive a message indicating that the database must be migrated with the migration utility.

7.3.1.1 OS/2 Restore Considerations

The following list pertains only to the OS/2 platform and the restore operation:

- In OS/2, the restore database command can call a user exit program only if a the backup copy was made using an exit program.
- There is ADSM support for restore in OS/2. Version 2.1.2 of the ADSM client is a prerequisite for ADSM support in OS/2.
- The restore copy may be located on diskette in OS/2.

7.3.2 Target Database Existence (Database Level Restore)

When restoring a database, you have two options: to restore to an existing database or to restore into a new database. Figure 42 on page 162 shows the two options.

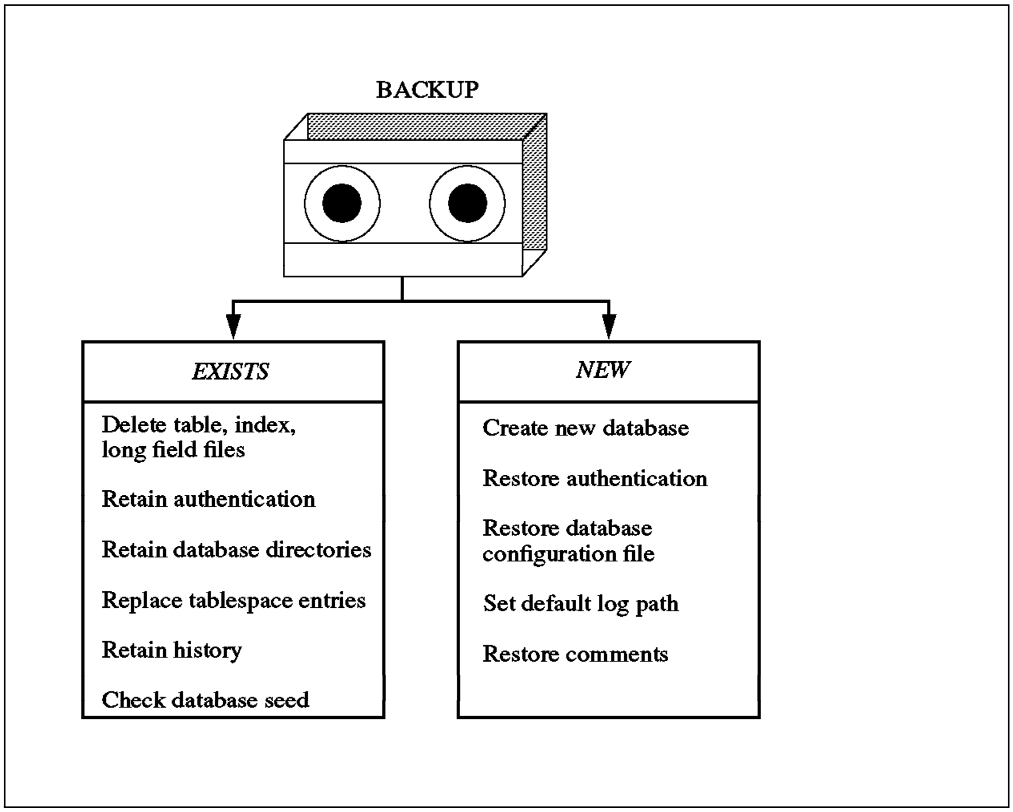


Figure 42. Restoring a Backup Image to a Database

There are certain considerations for each case:

1. Restoring to an Existing Database

   You may restore an image of full database backup or tablespace backup to an existing database of one or more tablespaces. To restore to an existing database, you must have SYSADM, SYSCTRL or SYSMAINT authority.

   When restoring to an existing database (the database alias in the image matches the target alias), the restore task performs the following:

   a. Delete table, index and long field contents for the existing database and replace them with the contents from the backup image.

   b. Retain the authentication for the existing database.

   c. Retain the database directories for the existing database that define where the database resides and how it is cataloged.

   d. Replace tablespace table entries for each tablespace being restored.

   e. Retain the recovery history file unless the one on disk is damaged.

   f. Other tasks, which depend on whether or not the database seed in the backup image matches the seed of the target database. A database seed is a unique identifier of a database that remains constant for the life of the database. This seed is assigned by the database manager when the database is first created.

2. Restoring to a New Database

   As an alternative to restoring to a database that already exists, you may create a new database and then restore the backup image of the data. To restore to a new database, you must have SYSADM or SYSCTRL authority.

In this case, the restore database command will perform the following functions:

a. Create a new database, using the database name and database alias name that was specified by the target database alias parameter. (If this target database alias was not specified, the restore database command will create a database with the name and alias the same as the source database alias parameter.)

b. Restore the authentication type from the backup image.

c. Restore the database configuration file from the backup image.

d. Modify the database configuration file to indicate that the default log file path should be used for logging.

e. Restore the database comments from the backup image for the database directories.

7.3.2.1 Other Database Restore Considerations

If the target alias that you are restoring to and the alias in the backup image are the same, the database seed is checked. Figure 43 shows the differences when the database seed is the same and when the database seed is different.

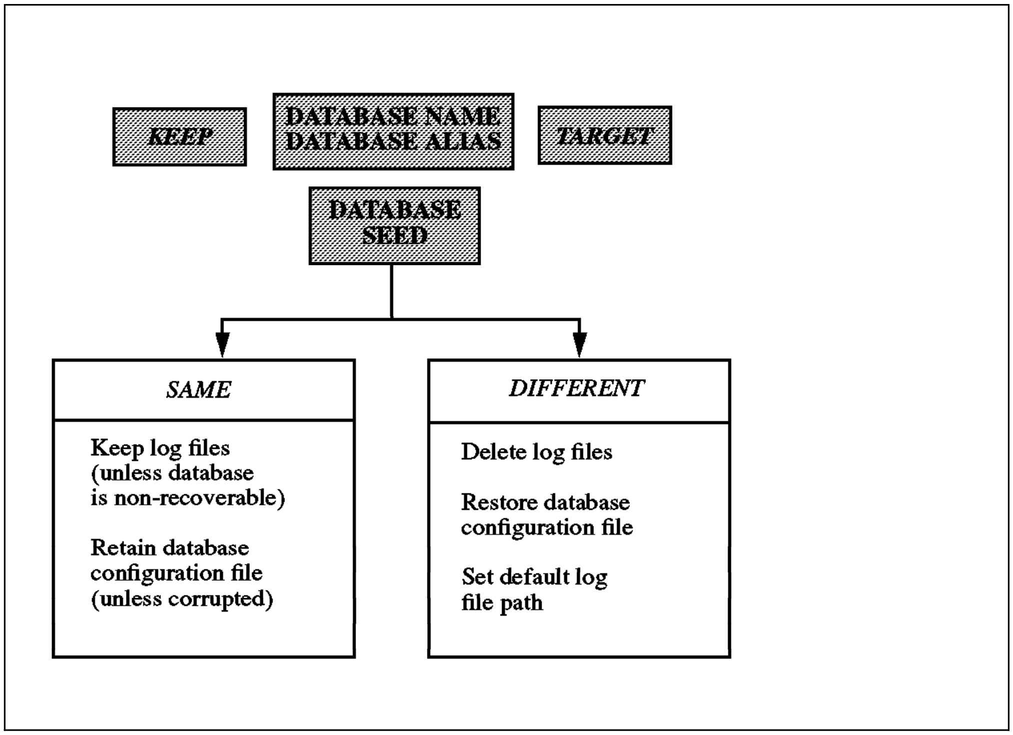

KEEP
DATABASE NAME
DATABASE ALIAS
TARGET
DATABASE SEED
SAME
Keep log files (unless database is non-recoverable)
Retain database configuration file (unless corrupted)
DIFFERENT
Delete log files
Restore database configuration file
Set default log file path

Figure 43. Checking the Database Seed Upon a Restore

If the database seeds are different, the backup image was not made from the database being restored. Restore will continue with the following:

- Delete the logs associated with the existing database
- Copy the database configuration file from the backup image
- Change the database configuration file to indicate that the default log file path should be used for logging.

When the database seeds are the same, the backup image was created from the database being restored. Restore will continue with the following

- Retain the current database configuration file, unless the file is corrupted, in which case this file will be copied from the backup image.
- Delete the logs if the image is of a non-recoverable database. Otherwise, the log files will be kept.

7.3.3 Understanding the Restore Utility Parameters

The first step to understanding the restore utility is to discuss the parameters of the command:

```
►►──RESTORE DATABASE──source-database-alias──┬──────────────────────────────────┬──────────►
                                             └─USER──username──┬──────────────────┬─┘
                                                               └─USING──password─┘

►──┬──────────────────┬──┬────────────────────────────────────────────────────┬──────────►
   ├─TABLESPACE ONLINE─┤  ├─USE ADSM──┬─────────────────────────────────┬─────┤
   └─HISTORY FILE──────┘  │           └─OPEN──num-sessions──SESSIONS─┘     │
                          │         ┌─,─────────────┐                        │
                          ├─FROM──▼─┬─directory─┬─┴────────────────────────┤
                          │         └─device────┘                          │
                          └─LOAD──shared-library──┬────────────────────────────────┬─┘
                                                  └─OPEN──num-sessions──SESSIONS─┘

►──┬─────────────────────┬──┬──────────────────────────┬──┬────────────────────────────────┬──────►
   └─TAKEN AT──date-time─┘  └─TO──target-directory─┘  └─INTO──target-database-alias─┘

►──┬──────────────────────────────┬──┬──────────────────────────┬──┬─────────────────────────┬──────►
   └─WITH──num-buffers──BUFFERS─┘  └─BUFFER──buffer-size─┘  └─WITHOUT ROLLING FORWARD─┘

►──┬──────────────────────┬──────────────────────────────────────────────────────────────►◄
   └─WITHOUT PROMPTING─┘
```

The following list provides some of the options of the commands and their descriptions.

DATABASE source-database-alias
Specifies the alias name of the database to restore. The database to which you restore the data may be the same one as the data was originally backed up from, or it may be different. You may restore the data to a new or an existing database.

USER username Identifies the user name under which the database is to be restored.

USING password The password used to authenticate the user name. If the password is omitted, the user is prompted to enter it.

TABLESPACE ONLINE
Specifies that this is an online process, the default is offline. The means that other users can connect while the backup is being restored. Once the restore process begins for a tablespace, that tablespace is not usable until the restore completes and a successful roll forward to the end of log files has occurred. Online tablespace processing requires that the database is enabled for rollforward processing.

HISTORY FILE
Specifies that only the history file from the backup image will be restored.

USE ADSM OPEN num-sessions SESSIONS
Specifies that the database is to be restored from ADSM managed output using “num-sessions.” The necessary ADSM environment variables must be set and the DB2 Server registered as an ADSM Client with the ADSM Server before starting the restore operation.

FROM directory / device
Specifies the device or directory on which the backup image reside. You can choose more then one source if you backed up to several tapes and directories.

LOAD library-name OPEN num-sessions SESSIONS
Specifies that a third party vendor product will be used as the target for this backup and that “num-sessions” will be used.

TAKEN AT date-time
This is a parameter for the `restore` command that allows you to choose between several backup images. You must issue the full timestamp to indicate which database image is being restored.

To target-directory
Specifies the path to be used to create the target database. This path must exist on the server.

INTO target-database-alias
Specifies the name of the database to restore into. This allows you to change the name of a database at restore time.

WITH num-buffers BUFFERS
The number of buffers to be used.

BUFFER buffer-size
The size in pages of the buffer used for the appropriate process. The minimum value for this 16 pages. The default value is 1024 pages. If you specify a buffersize of 0, the value in the database manager configuration for this parameter will be used. These two parameters will discussed further in 7.7, “Performance Issues” on page 176.

WITHOUT ROLLING FORWARD
This option is only valid with an offline backup image of a recoverable database. This will indicate that you do not want to roll forward and the database should be made consistent to the end of the backup.

WITHOUT PROMPTING
Perform the restore without prompting for new media. If any other type of warning occurs or all the restore source devices have been processed and there are still additional images required to complete the restore, an error is returned to the caller and the restore fails.

7.3.4 Backup/Restore Tablespace Considerations

This section deals with only those considerations for backup/restore as it pertains to the tablespace. However, the database backup/restore considerations also exist for the tablespace. These have been discussed in 7.2, "DB2 Backup and Restore Considerations" on page 155.

The following are backup/restore considerations at the tablespace level:

- A tablespace backup and tablespace restore cannot be run at the same time even if the backup and restore cover different tablespaces.
- If you have tables that span more than one tablespace, you should backup (and restore) the set of tablespaces together. This will eliminate some complexity of your recovery strategies.
- A tablespace backup can include a single tablespace or multiple tablespaces. However, restore is not selective. Therefore, planning should be completed when determining the tablespaces to place on the backup image.
- The decision to select which tablespaces to backup will also reduce the time when the database is not available.
  - Long field and large object (LOB) data for a table must be placed in the same tablespace.
  - Long field and LOB data can be in a separate tablespace from the rest of the table data.
  - Index data could also be kept in a separate tablespace from the rest of the table data.
- Backup strategies could be planned to take advantage of tablespace backup, thereby shortening the time for the backup to complete.

  It is also possible to group several related tables together so that the tablespaces they share can be considered a "unit" for backup and restore. This could provide the "granularity" required for backup and restore to complete in a shorter time frame, while keeping the management of backups reasonable.

7.3.4.1 Tablespace Level Restore and Roll Forward Recovery

Tablespace restore can be completed starting with a backup image that contains one of more tablespaces as shown in Figure 44 on page 167. The tablespace level restore must be to the same database. This is different than the options that exist for a full database restore. The name and seed for a tablespace restore must be identical. When a tablespace level backup is done, part of the information that is stored in the header is the "lifelsn." This is the log sequence number that exists for the database recorded during the tablespace backup. There is a timestamp that is recorded in this information. This must match when doing the tablespace restore.

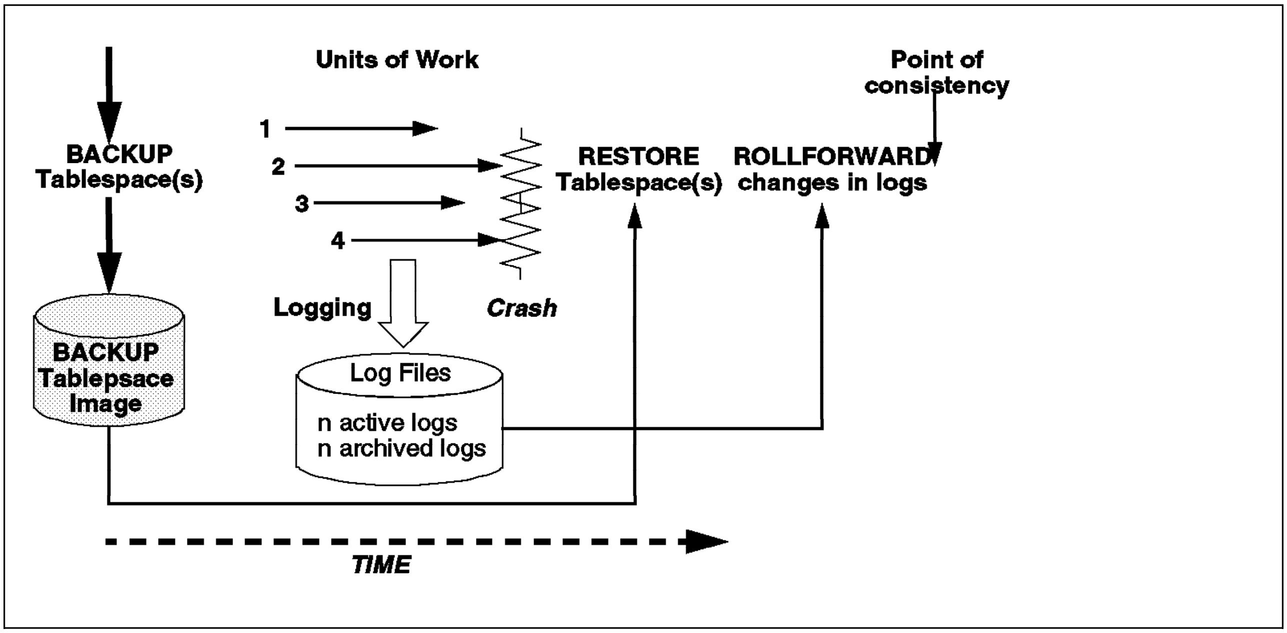


Figure 44. Tablespace Level Restore

To ensure that restored tablespaces are synchronized with the rest of the database, the tablespaces must be rolled forward to the end of the log files. For this reason, tablespace 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 tablespace level backup is executed, it will not be possible to restore from the backup, and then to roll the tablespace forward to the current point in time. In this case, all tablespace 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.

Another important consideration in tablespace backup/restore is that you must manually retrieve any archived log files. This is not done automatically for a tablespace level roll forward.

In cases where it cannot be determined that the backup is 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. This technique does not require the backup restored in phase one of the recovery process to reflect a point of consistency. Figure 44 shows the backup of the tablespace reflecting a point of consistency. Here, the backup image of the tablespace(s) was taken. Logging continued with transactions being recorded in the log files at a point of time after the backup completed. At a later point in time, a crash occurred that required the tablespace backup to be restored. The roll forward must take place and go to the end of the log files to ensure a point of consistency within the tablespace and database.

Each component of a table may be backed up and restored with the tablespace in which it resides, independently of the other components of the table. An exception to this general rule concerns recovery strategies for tables involving LOB and long field data. If the roll forward phase of such a restore process includes a REORG of such tables, the backup image previously restored must contain all related LOB and long field data. This could mean that the backup image must contain more than one tablespace if the LOB and long field data was placed in separate tablespaces.

7.3.4.2 Summary of Backup/Restore Considerations

Table 24 summarizes the considerations for backup/restore at both the tablespace and full database level.

Table 24. Limitations and Restrictions of Backup

	Full Database Backup Offline		**Full Database Backup Online**	**Tablespace Backup Offline**	**Tablespace Backup Online**
Logging Type Allowed	Archive	Circular	Archive	Archive	Archive
Access Allowed During Backup Process	No Access to Database		Full Access to Database	No Access to Database	Full Access to Database
Database State Following Restore	Database in Rollfoward Pending State	Database Consistent	Database in Rollforward Pending State	Database in Rollforward Pending State	Database in Rollforward Pending State
Rollforward Required	Any Point in Time	N/A	Any Point in Time at Past End of Backup	End of Log	End of Log

Table 24 points out considerations for your backup/restore strategy:

- Consider the type of logging to be used - circular versus archival
- Decide what type of access you can live with for both backup and restore
- Realize that your recovery action may include a roll forward operation
- Understand the point in time to which a roll forward must take place

7.3.5 Tablespace Backup and Restore Scenarios

In 4.5, “Planning Your Tablespace Environment” on page 57, a sample database environment was presented. We will use that environment to discuss some backup and restore strategies.

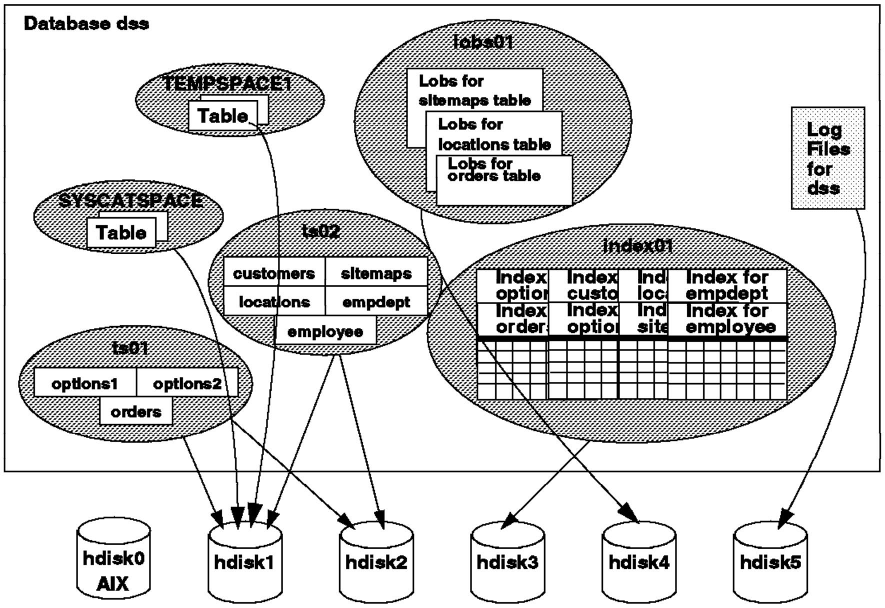


Figure 45. Example Database and Tablespace Environment

Figure 45 shows the database, dss, with both DMS and SMS tablespaces. Not all the tables are represented here for this discussion. We will discuss a number of examples using this environment as it pertains to the tablespaces in the dss database.

7.3.5.1 Log Files and NEWLOGPATH Parameter

As shown in Figure 45, the log files for the dss database will be placed on a separate disk from the database itself. This will help in any recovery scenario. The database parameter, NEWLOGPATH, has been set to point to the directory/filesystem that is created on hdisk5. This is done before any connections are made to the database.

7.3.5.2 System Catalog Backup/Restore

If the tablespace being restored is the system catalog tablespace (SYSCATSPACE), then no other connections to the database are possible until restore and rollforward recovery have completed. Figure 45 shows the dss database environment. If the SYSCATSPACE tablespace that store the system catalogs is damaged, no users will be able to connect to the database. If a tablespace level backup containing the system catalog tablespace is available, you can restore the tablespace backup and do a roll forward to the end of the log files. If there is no backup copy that can be restored for the system catalog tablespace, then you must perform an entire database restore.

7.3.5.3 Long/LOB Tablespaces

With Database Managed Storage (DMS), the user may choose to separate the data, index and long field (LONG/large objects (LOBs)) into different tablespaces. Only LONG and LOB for the same table must reside in the same tablespace. Each component of a table may be backed up and restored if in separate tablespaces. Figure 45 shows that a separate backup of the lobs01 tablespace may be made only as often as needed.

7.3.6 Online Archived Log File Reuse

There are special restrictions to be checked when restoring a tablespace backup to determine if the backup can be used. This is necessary because a full database restore changes the valid sequence of logs for roll forward recovery. The log files are re-used. After restoring a tablespace backup, it must be possible to roll forward the tablespace to the current point-in-time.

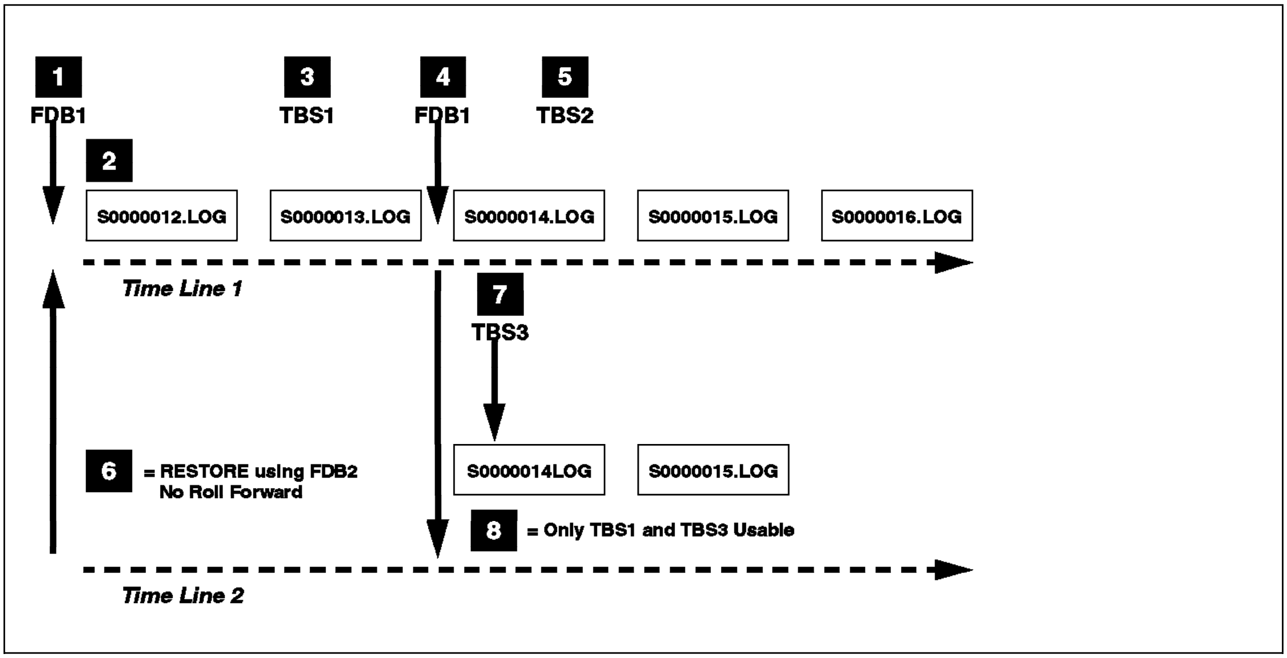


Figure 46. On-Line Archived Log File Re-Use

Figure 46 shows a scenario where two full backups (FDB1 and FDB2) and three tablespace backups (TBS1, TBS2 and TBS3) are taken. This is indicated by Time Line 1. The next time line, Time Line 2, shows that the full backup, FDB2, is restored with no roll forward. Processing continues and tablespace backup TBS3 is taken. This is also represented with Time Line 2. If a tablespace found in all three tablespace backups becomes disabled, the only useable tablespace backups are TBS1 and TBS3. The reason why TBS2 is unusable is that there is no path from backup TBS2 through the logs to the current point-in-time (second copy of S0000015.LOG); TBS2 was taken when an old version of log S0000015.LOG was in use.

For completeness, Figure 46 can be broken down into the following steps:

1. Full database backup (FDB1) is taken
2. Logging begins in S0000012.LOG
3. Tablespace backup (TBS1) is taken
4. Second full database backup (FDB2) is taken
5. Second tablespace backup (TBS2) is taken
6. Full database restore is done using FDB2 with no roll forward
7. Tablespace backup (TBS3) is taken
8. Only TBS1 and TBS3 are usable for tablespace restores; TBS2 is lost

7.4 Redirected Restore

Redirected restore is an option that allows you to redefine containers during recovery. It can only be performed using the Database Director. Performing this option using the Database Director is covered in 8.3.6, “Restoring a Database” on page 200.

Redirected restore may possible be required to support:

- A container that cannot be accessed.
- Disaster Recovery
- “Copying” a Database
- Adding containers through the alter tablespace Command

These considerations are discussed in more detail.

7.4.1.1 Validating the Containers during a Restore

The redirected restore may be necessary as validation of the list of containers while doing a restore. As part of the restore process, RESTORE validates the list of containers needed by the tablespace. RESTORE first tries to create the containers if they do no exist.

If the containers that cannot be created are currently used by another tablespace or are inaccessible for any other reason, then the list of containers needed must be redefined before the restore can continue.

7.4.1.2 Disaster Recovery/Copy

When you are restoring the database to a new location, the list of tablespace containers will not likely match between the backup database image and the new database. There is no way to prevent having to redefine the container list in this case. The Database Director will indicate which containers need to be defined before the restore can proceed.

After the list of containers has been redefined, a confirmation or re-validation of the list takes place. You will be able to correct the list of containers or cancel the action.

7.4.1.3 The Alter Tablespace Command

The alter tablespace command to ADD FILE or ADD DEVICE adds a tablespace container to a specific tablespace.

When restoring to the same database, the list of tablespace containers in the backup image will all be validated during the first phase of restore. It is only during roll forward that the log records associated with the alter tablespace command are encountered. If the container associated with the alter tablespace command is accessible, the application of the log records proceeds and the ALTER is re-applied. Otherwise, the tablespace is left in roll forward pending status.

One way to limit having to correct the list of containers is by doing a backup following the addition of containers to a tablespace. This is not a system requirement, but as long as the RESTORE uses the most recent backup image of the database, then redefining the container list is avoided.

Directory and file containers are created automatically if they do not exist. No redirection is necessary unless the containers are not accessible for some other reason. Device containers (AIX only) are not automatically created. Conversely, device containers are not removed.

7.5 Failures and Recovery Actions

You need to know the strategies available to you to help when there are problems with the database. Typically, you will deal with media and storage problems, and application failures.

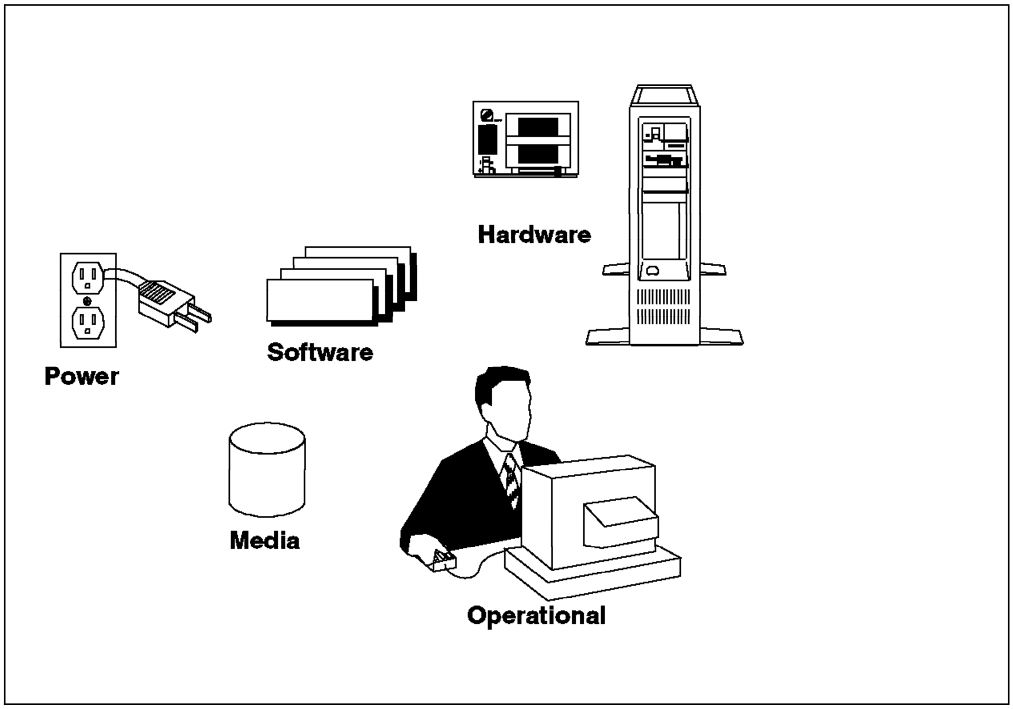


Figure 47. Potential Problems

You need to know that you can backup up your database, or individual tablespaces, and then rebuild them should they be damaged or corrupted. The rebuilding of these objects is called recovery. DB2 provides the capability to recover from a variety of data processing problems.

The degree of recovery is dependent on the type of recovery required. For example,

- Recovery from a program logic flaw can only be made to a point before the erroneous program began.
- Recovery from a power failure can restore the database to a consistent state up to the last committed unit of work.

In order to support recovery from all the potential problems, the database administrator needs to take full advantage of all the recovery features of DB2.

7.5.1 Methods of Recovery

There are basically three types or methods of recovery as shown in Figure 48:

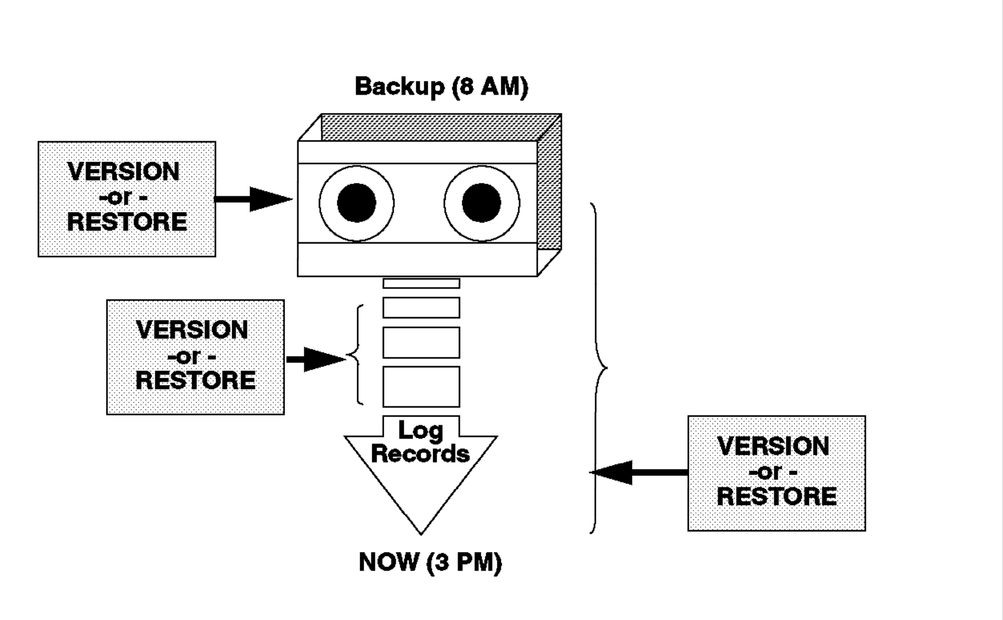


Figure 48. Methods of Recovery

As shown in Figure 48, these three methods of recovery are:

1. Crash Recovery

   Entails using the log files to recovery from power interrupts or application abends. Crash recovery can be initiated by entering a `restart database` command. However, it is more common to use the automatic restart enable (AUTORESTART) database configuration parameter. This will cause crash recovery to occur automatically at the first connect after a failure. The default for this configuration parameter is that the `restart database` command will be started every time it is needed.

2. Restore or Version

   Entails using a backup copy of the database to replace the current data in the database. This type provides recovery from media, hardware, operational and software problems, but only to the point that the backup copy was made. Therefore, the more frequent the backups, the more current the recovered database will be.

   You must schedule and perform a full backup of the database on a regular basis. The longer the time between the backups, the greater the number of units of work that may be lost. The loss of units of work must be acceptable within your business operation or you must have a way to re-apply the missing units of work against the restored database.

   Tablespace recovery is not supported by Restore or Version strategy.

3. Roll Forward Recovery

   Entails using a backup copy of the database or tablespace to replace the current data and then applying log records to recover changes made after the backup images was created. This type provides recovery from media, hardware, operational and software problems. The recovery can be to a point in time or the last committed unit of work. Note, that if tablespace level recovery is being completed, the only option is to apply all of the log records. Point in time recovery is not supported for tablespaces. Rollforward recovery using a recent backup will require less log record application than a rollforward using an older backup, so the frequency of backups will still be a consideration for the database administrator.

To use the database roll forward recovery method, you must ensure that log retention is done and that the log files created since the backup are available for the roll forward process. For more information on logging, see Chapter 6, "Logging" on page 131.

7.5.2 Media Failure

In Version 1, a media error would cause the entire database to be brought down. If the database was not able to perform a crash recovery, the entire database would be marked as damaged. The only recovery method available was to restore the entire database.

In Version 2, the media error can now be isolated at the tablespace level. This leaves remaining tablespaces in the database still accessible for use. If an I/O error is encountered while trying to read or write data to disk, the tablespace in which the page read is disabled. That tablespace is placed in a rollforward pending state.

Note:

This is only possible if the database has roll forward recovery enabled; otherwise, the database will be brought down with SQLCODE -902 as it does in Version 1.

By putting the tablespace in rollforward pending state instead of recovery pending, the user is able to recover the tablespace by rolling forward only the database. The tablespace does not have to be restored to do roll forward recovery. Before the tablespace becomes completely disabled, the active transactions accessing the tablespace are allowed to complete. While these existing transactions are completing, the tablespace is placed in an intermediate state called disabled pending, in which access to the tablespace is still allowed with restrictions. This allows the existing transactions to write all log records to disk before access to the tablespace is totally restricted.

If a transaction received an I/O error in the tablespace during the rollback processing, the tablespace will be disabled immediately; the tablespace is placed in rollforward pending

7.6 Tablespace Backup Guidelines

The following lists some guidelines for implementing a backup/restore strategy at the tablespace level:

1. Roll forward must be enabled
2. All log records must be applied
3. Multiple tablespaces in image may be desirable
    - Ease of tablespace recovery strategy
    - Access to related tables managed coherently

4. Long field/LOB data and REORG
5. All tablespaces in image restored
6. Catalog tablespace good candidate
7. Critical application tables tablespaces
8. Online or offline

In order to use tablespace level backup and restore, roll forward recovery must be enabled. Furthermore, the recovery point in such a strategy is to the end of the logs. Point in time recovery is not permitted at the tablespace level.

One of the key reasons to use tablespace recovery is to reduce the time of backup and restore. This is accomplished by reducing the amount of data involved in these processes. However, the database administrator should not strive to simply minimize the amount of data in a backup. Although this could be accomplished by placing a single table in its own tablespace, such implementation would likely lead to a management problem concerning backup and recovery.

If the tablespace associated with closely related tables is contained in a single backup, only the applications targeting the closely related tables are impacted. In many cases, such applications would be impacted even if a single table in the “group” was unavailable. (Assuming one table per tablespace.) This is especially true in the case of referentially constrained tables. You should consider grouping the tablespaces that support referential structures in a single backup image.

The key is to reduce the amount of data involved in the event that recovery is necessary while controlling the impact on management of the backup/restore strategy.

LOB data and long field data can be placed in a tablespace that is separate from “regular” data. This can assist the recovery time since the frequency of backing up LOB data is not as often as the frequency of backing up “regular” data. The nature of LOB data is that is tends not to be updated frequently and it is large. However, if a REORG of such a table is part of the roll forward process, you must have all tablespaces relating to the table in the backup. This would defeat one of your reasons to separate the data. The solution is to establish new backups of the tablespaces associated with such tables after you perform the REORG.

If you are logging large object (LOB) data, you have to consider the impact on performance. If you turn logging on for LOB data, your application′s performance will deteriorate and you may encounter problems related to the increased size of the log file. If you turn the logging off for LOB data, your application′s performance improves, however, its recoverability is sacrificed.

It is not possible to selectively restore a tablespace from a backup that contains several tablespace images. All are restored. Therefore, the choices you make during your backups will impact the granularity of your restore.

7.6.1 Backup/Restore Recommendations

No two installations will have identical recovery requirements, but the following are some considerations:

- Place the log files and database files on separate physical disks.
- Decide what medium (disk, tape or ADSM) can best handle the volume of log data to be generated in your environment.
- An offline backup/restore in Version 2.1.1 will perform better than an online backup/restore.
- If recovery is critical, consider mirroring the log files and making copies of backups. (AIX only)
- Determine how far back recovery should be enabled for your needs.
- Determine when log files and backups may be discarded.
- Consider using a user-exit and modifying it for your environment.

Particular consideration should be made to the use of user exit programs. Although these are supplied by IBM as samples and are therefore not guaranteed regarding functionality, they are a good starting point. It is necessary to test the functionality at your installation. Table 25 summarizes key points regarding backup and logging and their effect on resources. The descriptions of resources and management are relative. For example, some database administrators may consider management of an installation′s roll forward recovery strategy simple, once the planning and implementation process has been completed. However, compared to supporting crash recovery, the cost is high.

Table 25. Logging/Backup Requirements Summary

Consideration	**Crash**	**Version**	**Roll Forward**
Logging	Circular or Archival	Circular or Archival	Archival
Backup	N/A	OFFLINE	OFFLINE or ONLINE
Resources	Low	Medium	High
Management	Low	Medium	High
Currency after Media Failure	Not Provided	Latest Backup	Last Committed Work
Tablespace Recovery	N/A	No	Yes

7.7 Performance Issues

There are a number of items in Version 2 that may affect the performance of a backup/restore. These items are as follows:

- Offline backup versus online backup
- Number of buffers
- Size of buffers

In Version 2.1.1 of DB2, offline backup is faster than online backup. It is linear scalable. The term “linear scalable” in this context refers to the increase in performance when adding more disk to the backup/restore process. The performance of an online backup/restore would also benefit, but is even more noticed when doing an offline backup.

7.7.1.1 Buffers and Performance

The number of buffers and size of buffers can have a direct impact on the performance of your backup/restore. The exact figures will depend on your installation. The number of buffers you allocate should be at least twice as many as the number of I/O devices you are using.

The size of buffers can also affect performance. Choosing the size of the buffer is related to the amount of memory in your system configuration available for the backup/restore process. When possible, the buffer should reside in memory during the backup/restore process and not be swapped out.

Another consideration is the type of tablespace could affect the size of the backup/restore buffer. A large buffer size for DMS tablespaces would improve the backup/restore performance because of the internal format in which the DMS tablespace is backed up or restored.

7.8 Recovery History File

A recovery history file contains historical information for a database. The file is maintained at the database level and resides in the same directory as the database configuration file. If the database is dropped, the history file is lost.

The recovery history file is updated when any one of the following operations is performed:

- A backup of the full database or tablespace(s)
- A restore of the full database or tablespaces(s)
- A load of a table

The recovery history file provides a summary of backup information useful in the event that a database or tablespace must be restored. The backup information includes:

- The part of the database that has been copied by a backup, load or copy operation
- When the database was copied
- The location of the copy
- Time of the last restore

The full format of the file can be found in 7.8.2, “Format of Recovery History File” on page 179. For example, to list all the backups, restores and loads that have been done for a database (here called dss), the command would be:

```
db2 list history all for dss
```

An option on the restore command permits only the history file in a backup image to be restored. This is useful in recovery situations where the history file currently associated with the database is not accessible and information contained in the history file is needed to plan the recovery strategy.

All insertions to the recovery history file are done automatically whenever a backup, restore or load is performed. A warning will be given if the history file cannot be inserted into or updated, due to either a full file system, a damaged history file or an I/O error. If the file system is full, the current entry will be lost, since insertions are made as the backup, restore or load is being done. The user cannot insert records into the history directly. In the case of a full file system, this information would be lost.

7.8.1.1 Managing the Recovery History File

You can manage the recovery history file by using the prune history command.

The prune history command may be used to delete entries from the recovery history file. The syntax for the PRUNE HISTORY file is as follows:

```
prune history timestamp with force option
```

The timestamp is used to identify a range of entries in the recovery history file. A complete timestamp (in the format yyyymmddhhmmss) or an initial prefix (at a minimum, yyyy) may be specified. All entries with timestamps equal to or less than the timestamp provided are deleted from the recovery history file.

The WITH FORCE OPTION specifies that the entries will be pruned according to the timestamp specified, even if some entries from the most recent restore set are deleted from the file.

The following are examples of the prune history command:

```
db2 prune history 19950726100922
db2 prune history 19950726 with force option
```

You must have SYSADM, SYSCTRL, SYSMAINT or DBADM authority to use the prune history command.

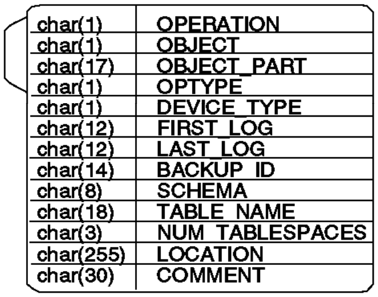
There will a recovery history file for every database. The size of the file depends on the REC_HIS_RETENTN database configuration parameter and the frequency of backups, restores and table loads. REC_HIS_RETNTN is used to set the retention period of the history file; the default is 366 days. If the recovery history file is not needed to keep track of backups, restores or loads, REC_HIS_RETENTN can be set to a small value.

If you want to keep the recovery history file indefinitely, set the REC_HIS_RETNTN value to -1. In this case, the user must explicitly prune the recovery history file. By default, the recovery history file is automatically pruned after recording a full database backup.

No matter how small the retention period, the most recent full database backup plus its restore set will always be kept unless the PRUNE with FORCE option is used.

7.8.2 Format of Recovery History File

Figure 49 shows the format of the recovery history file.



char(1)	OPERATION
char(1)	OBJECT
char(17)	OBJECT_PART
char(1)	OPTYPE
char(1)	DEVICE_TYPE
char(12)	FIRST_LOG
char(12)	LAST_LOG
char(14)	BACKUP_ID
char(8)	SCHEMA
char(18)	TABLE_NAME
char(3)	NUM_TABLESPACES
char(255)	LOCATION
char(30)	COMMENT

Figure 49. Format of the Recovery History File

The following lists the columns contained in the recovery history file and their description:

Table 26 (Page 1 of 2). Format of the Recovery History File

Column Name	Type	Description
OPERATION	Char(1)	Type of operation performed, B=Backup, R=Restore, L=Load
OBJECT	Char(1)	Granularity of operation, D=Full Database, P=Tablespace, T=Table
OBJECT_PART	Char(17)	First 14 characters=timestamp=yyyymmddhhnnss, Next 3 characters=Sequence Number. Backup can save to multiple files/tapes Restore/Load always ′001′
OPTYPE	Char(1)	Additional operation qualification, F=Offline backup, N=Online backup, R=Load Replace, A=Load APpend, C=Load Copy, Blank for other operations
DEVICE_TYPE	Char(1)	D=Disk, K=Diskette, T=Tape, A=ADSM, U=UserExit, O=Other Vendor Device Support
FIRST_LOG	Char(12)	Earliest Log File ID (S0000000 to S9999999) Required for roll forward recovery after full database/tablespace backup
LAST_LOG	Char(12)	Latest Log File ID
BACKUP_ID	Char(14)	Timestamp ′yyyymmddhhnnss′ that references one or more file lines representing backup operation. For full database restore, references full database backup that was restored. For tablespace, references tablespace backup or full database backup used to restore specified tablespaces. Blank for other operations.
SCHEMA	Char(8)	Tablename qualifier for load
TABLE_NAME	Char(18)	Name of Loaded Table
NUM_TABLESPACES	Char(3)	Number of tablespaces involved in backup/restore. If non-zero, next lines in fill contain one line for each tablespace.

Table 26 (Page 2 of 2). Format of the Recovery History File

Column Name	Type	Description
LOCATION	Char(255)	Where data is saved for backups/load copy. For restore/loads where first part of data was saved. Refers to DEVICE_TYPE. If D or K, then fully qualified file name. If T, then tape volume label. If A, then ADSM server name. If U or O, then free form text.
COMMENT	Char(30)	Free form text.

7.8.2.1 Damaged Recovery History 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 image to use to restore the database. This restored history file will contain all entries up to, but not including backup used for restoration.

7.9 Customer Scenario - The Telephone Company

The backup strategy of the phone company is conditioned by the time window available to perform backups. The batch time window for DB2/MVS is from 3 PM to 7 PM. Changes generated by all the batch jobs are then propagated to DB2/AIX. Database and tablespace backups are taken using scripts executed by cron daemons. Time window for backups is from 8 PM to 6 AM. ADSM is not installed. The backup strategy relies on a weekly database backup and selected daily tablespace backups. Archival logging is being used.

Tables were grouped into tablespaces according to the number of modified rows per day:

- Tablespace space01 stores all the tables with a low number of modifications.
- Tablespace space02 is used to store tables with a “medium” number of modifications per day.
- Tablespaces space03 to space11 store frequently modified tables.

The backup policy implemented by the database administrator takes in consideration the volume of modifications undergone by the tablespaces. Taking backups more often of frequently modified tablespaces will reduce the space needed in the log files to store those changes, and will reduce the time to roll-forward the changes after restoring a tablespace. The backup schedule outlined by the DBA is the following:

- Monday: Tablespace backups of the space03 to space11 tablespaces.
- Tuesday: Tablespace backups of the space01 to space07
- Wednesday: Tablespace backups of the space03 to space11 tablespaces.
- Thursday: Tablespace backups of the space01, space02, and space08 to space11 tablespaces.
- Friday: Full database backup.

Before taking backups, the amount of memory given to the utility heap is increased. As in the case of the load utility, the backup utility does not use the buffer pool. The buffer pool is reduced to 100 MB and the UTIL_HEAP_SZ is increased to 20000 pages. The memory allocated for the buffers used by the backup utility comes from the utility heap.

All the backups taken share the following options:

- The 5 GB 8mm tape drive as destination device.
- No backups are taken ONLINE.
- The number of buffers used is 16.
- The size of this buffers is 1024 4KB pages.

Chapter 8. Data Access

There are different methods of accessing DB2 resources, including:

- Command Line Processor (CLP)
- SQL Query Products, such as Visualizer
- Database Director

8.1 Command Line Processor (CLP)

The Command Line Processor (CLP) is provided in all of the DB2 V2 products. The amount of functionality provided by CLP is determined by the product which you are using. For example, The Client Application Enabler (CAE) provides a minimal CLP interface which allows you to `connect` to a remote database and `bind` an application. The Software Developer Kit (SDK) provides an enhanced CLP which allows you to do such things as precompile an application. The Single-User and the Server versions of DB2 provide a full CLP interface which allows the user to execute SQL statements.

To invoke the CLP, simply type `db2` at a command line. You can issue SQL statements or DB2 commands using CLP. If you are issuing a DB2 command and you need to verify the syntax of the command, issue `db2 ? command-name`. The syntax of the particular command will then be displayed. If you are attempting to issue an SQL statement, you can easily verify the syntax of the statement by entering `db2 help sql-statement`. This will display the appropriate section of the SQL Reference manual online.

Many of the examples throughout this book use the Command Line Processor (CLP) as the method of performing DB2 tasks, such as backup and load. The syntax for these utilities can become very long, and errors can easily occur. The first step to making CLP commands more usable is to place the commands into a file. To execute commands from a file, you must use the *-t* of the CLP.

There have been some time-saving changes added to CLP in V2. The following is a list of abbreviations recognized by CLP. This will help save a few keystrokes when using interactive CLP commands.

The word database can be abbreviated as `db`, as in the following commands:

```
db2 list db directory
db2 get db configuration for sample
```

The term “database manager” may be abbreviated to `dbm`.

```
db2 get dbm directory
```

The word “configuration” can be shortened to `cfg` or `config`.

```
db2 get dbm config
```

The CLP interface provides the ability to issue DB2 commands and most SQL statements and therefore is the most flexible interface available to manage a DB2 installation. Some of the limitations of CLP are its speed and inability to integrate with some of the new features of DB2 V2, such as Visual Explain and the Graphical Performance Monitor. There is another variation of CLP called db2batch. It is a tool to be used in benchmarking and tuning a query. The db2batch program is located in the misc directory of the instance.

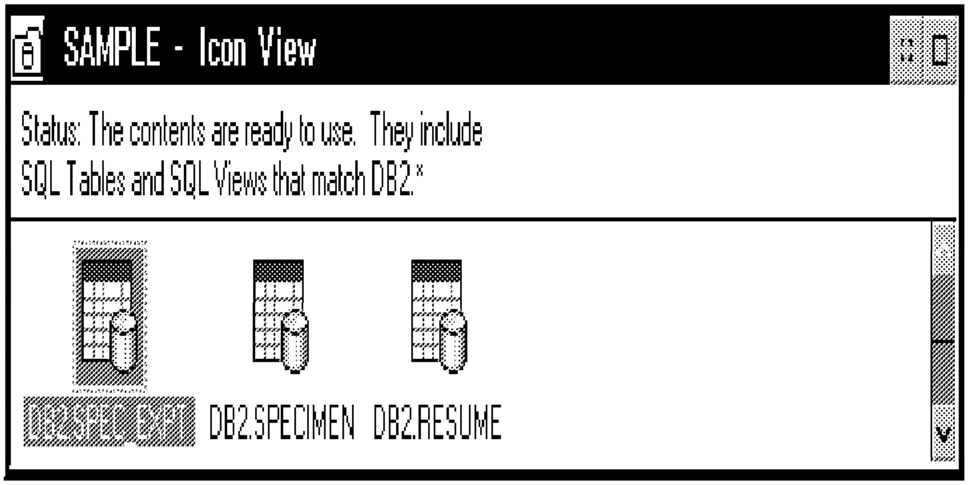
As mentioned previously, the Command Line Processor is the most flexible user interface to DB2. There are some tasks that can only be performed using CLP or the equivalent API (NOT accessible using the Database Director), including:

- Load
- Import
- Export
- Reorg
- Runstats

8.2 SQL Query Products

There are various query products available which use SQL to manipulate DB2 resources. DB2 for OS/2 V1.x provided a SQL query tool, called Query Manager. This product is not shipped with V2.x. Therefore, an alternative product should be used to issue SQL queries. Query Manager may still be used but only as a remote client to a V2.x database. If using Query Manager, some of of the V2.x features cannot be utilitized. such as IBM′s Visualizer Products, Lotus Approach or Microsoft Access are some of the more popular SQL query products used with DB2 today. Figure 50 on page 185 shows an example of the Visualizer SQL query product from IBM. Some of the day-to-day activities of a database administrator can be accomplished using SQL statements. Here are a few SQL statements that would be of interest to a database administrator:

- Alter tablespace - to add a new container to a DMS tablespace
- Create event monitor - to examine SQL activity
- Create index - to increase `select` performance on heavily accessed tables
- Create tablespace - to define a new tablespace within a database
- Grant - to provide database object access to users and/or groups
- Revoke - to prevent unauthorized access to database objects
- Set constraints - to remove the check-pending status of a table
- Set event monitor state - to activate/deactivate a defined monitor



Status: The contents are ready to use. They include
SQL Tables and SQL Views that match DB2.*

Figure 50. Visualizer for OS/2

The list of SQL statements does not include such tasks as making a backup image of a database and restoring the backup image. These tasks cannot be performed using SQL. To perform these tasks, you may decide to use a database management product such as DataHub, or use the Database Director shipped with DB2.

An SQL query product will not allow a database administrator the ability to perform key tasks, such as backup and restore. Therefore, using such an interface is only a partial solution to manage a DB2 database system.

8.3 Database Director

DB2 V1.x included some database tools known as DBAT (Database Tools). DB2 for AIX V1.x included a program that was invoked from an aixterm by the command db2adm. This utility is no longer shipped in V2. DB2 for OS/2 V1.x included a recovery tool, directory tool and configuration tool. These tools are also no longer shipped in V2. A new database-management interface called the Database Director is provided.

8.3.1 Getting Started

The Database Director provides a graphical interface that helps you perform common database administration tasks. It is provided with the SDK, Single-User or Server. The Database Director can be used to create, modify or delete various DB2 objects, such as tables and tablespaces. It can also be used to invoke the Performance Monitor Tool and the Visual Explain utility. The Database Director can be started by issuing the db2dd command from a window in AIX. In OS/2, the Database Director is started by selecting the Database Director icon in the DB2 folder. Figure 51 on page 186 shows the first screen of the Database Director. The interface is almost identical between OS/2 and AIX. (DB2 for Windows NT V2.1 does not have the Database Director available in the initial release.)

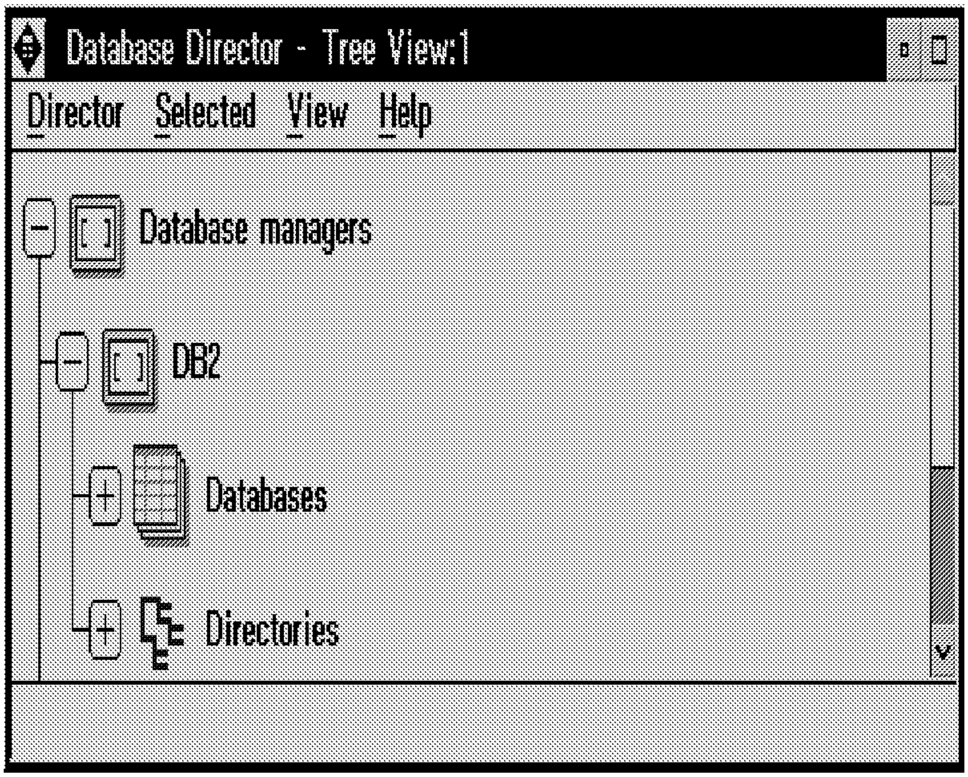


Figure 51. The Database Director Main Screen - Tree View

There are two different methods of viewing the DB2 objects within the Database Director, the tree view and the list view. The default view is the tree view, as seen in Figure 51. In Figure 52 on page 187, the equivalent list view of the DB2 objects can be seen. This view will display objects of the same type. Figure 52 on page 187 shows that there are three instances: DB2, INSTB and SERV1. The title bar for the Database Director will show which view is currently being displayed, followed by a number. Each time a Database Director window is created, a new number is assigned. There is no significance to this number other than to help identify the different windows that are open.

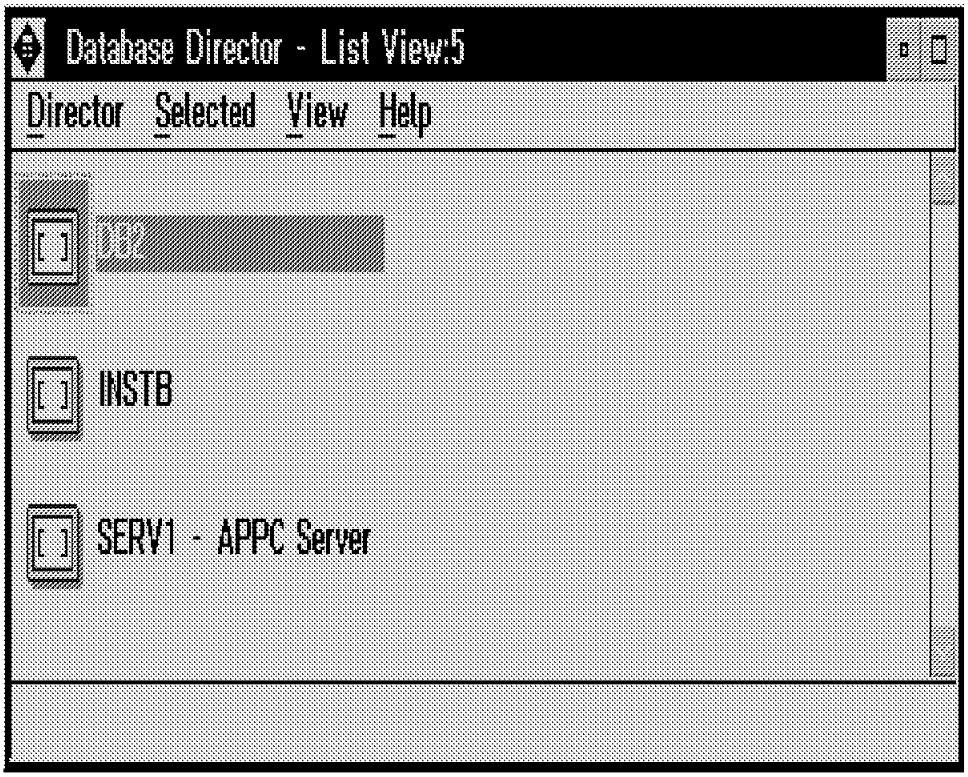


Figure 52. The Database Director Main Screen - List View

There are three main areas that can be accessed using the Database Director:

1. Instances
2. Databases
3. Directories (See Figure 51 on page 186)

The following sections discuss the tasks that are possible using the Database Director.

8.3.2 Configuration

A DB2 Server usually requires some type of tuning to achieve optimal performance. The task of tuning a DB2 Server (instance) and tuning a DB2 database involves the understanding of many configuration parameters. These parameters are stored in two different files: the DBM configuration file (sqlsystm) and the DB configuration file (sqldbcon). There are two ways to modify these configuration parameters:

1. Using the CLP

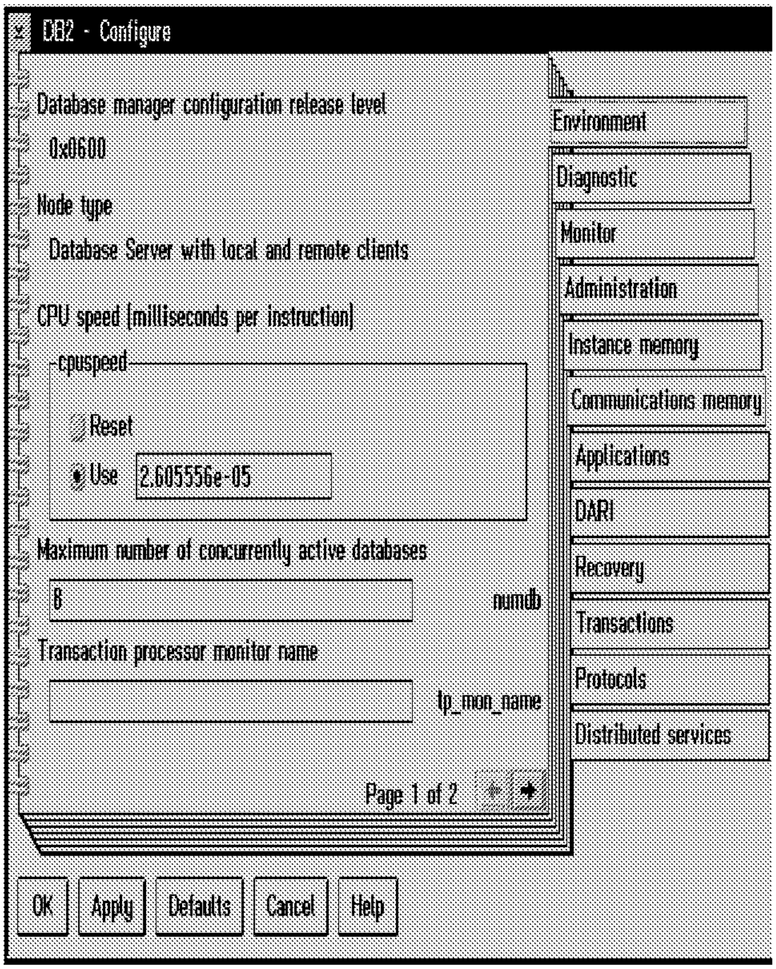
   ```
   db2 update database manager configuration using <keyword> <value>
   ```

2. Using the Database Director

We discuss the use of the Database Director to accomplish this task.

8.3.2.1 Configuring DB2 Instances

DB2 V2 has over 50 Database Manager (DBM) (instance) configuration parameters. Note that these configuration parameters are established for a single instance. In a multiple instance environment, it is important to know which instance you are configuring. Any changes to DBM configuration parameters take effect when all applications have disconnected from databases within the instance, the instance has been stopped and then restarted (db2start). If you have created more than one instance on a DB2 server or you would like to access instances residing on other DB2 servers, you must catalog all of the instances for them to be accessible via the Database Director. The Database Director uses the node directory to locate the instances which are displayed. Therefore, simply creating a new instance using the db2icrt command will not result in an object being displayed via the Database Director. The term used in the Database Director which refers to DB2 instances is Database Managers.



DB2 - Configure

Database manager configuration release level

0x0600

Node type

Database Server with local and remote clients

CPU speed (milliseconds per instruction)

cpuspeed

○ Reset

● Use 2.605556e-05

Maximum number of concurrently active databases

8 numdb

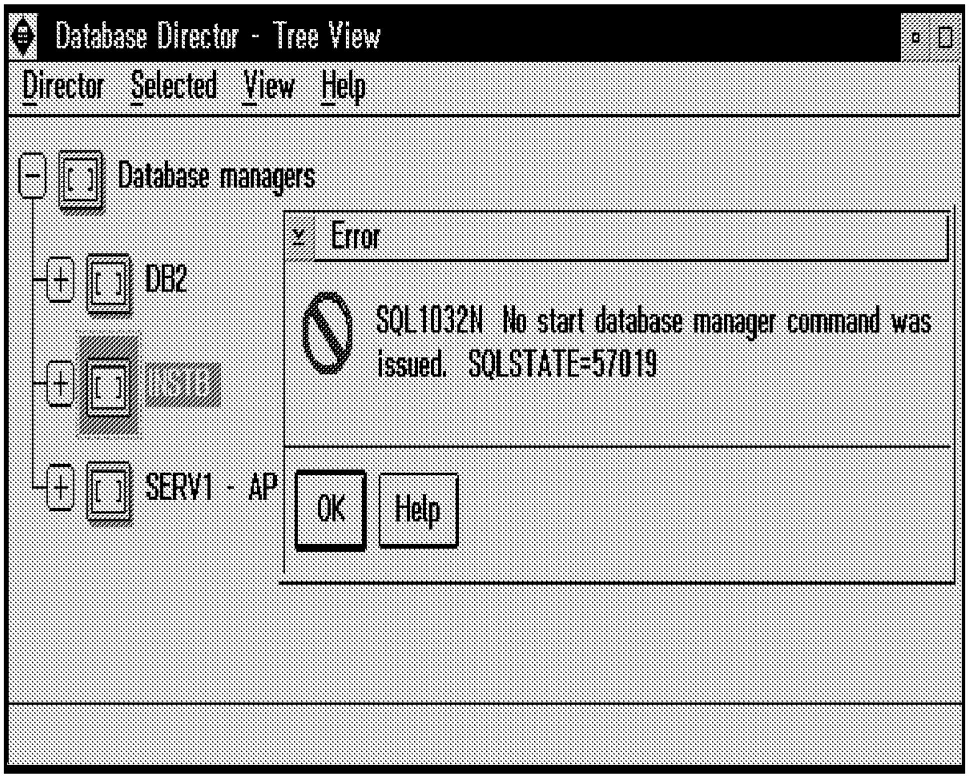
Transaction processor monitor name

tp_mon_name

Page 1 of 2

Figure 53. Notebook Settings for DB2 Instance (DBM) Configuration

The database director allows you to modify any of the DBM configuration parameters by using the pop-up menu for the instance you wish to configure. Figure 53 shows the DBM configuration panel for a DB2 instance. Notice that the parameters are grouped by type; this is useful in understanding how they relate to each other. Also, note that the database manager configuration release level is 0x600; this value is used to determine which release of DB2 Common Server is being configured. It is possible to configure any local or remote instance which has been cataloged, but the instance must have been started. (See Figure 54 on page 189.)

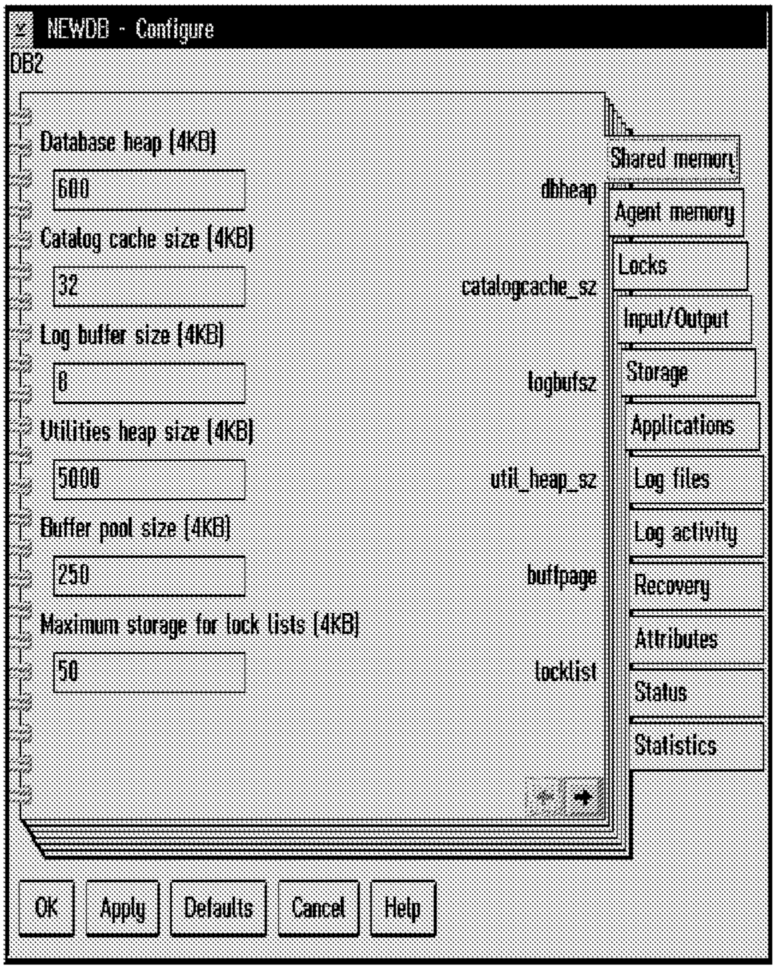


SQL1032N No start database manager command was issued. SQLSTATE=57019

Figure 54. Failed Attempt to Configure a DB2 Instance

8.3.2.2 Configuring DB2 Databases

There are over 50 DB (database) configuration parameters in DB2 V2. These parameters are stored within the database in the sqldbcon file, which is stored with a database backup image. Any changes to the database configuration take effect when all applications have disconnected from the database, and a new database connection is established (`connect to` dbname).


NEWDB - Configure

DB2

Database heap (4KB) 600 dbheap

Catalog cache size (4KB) 32 catalogcache_sz

Log buffer size (4KB) 8 logbufsz

Utilities heap size (4KB) 5000 util_heap_sz

Buffer pool size (4KB) 250 buffpage

Maximum storage for lock lists (4KB) 50 locklist

Figure 55. Notebook Settings for Database (DB) Configuration

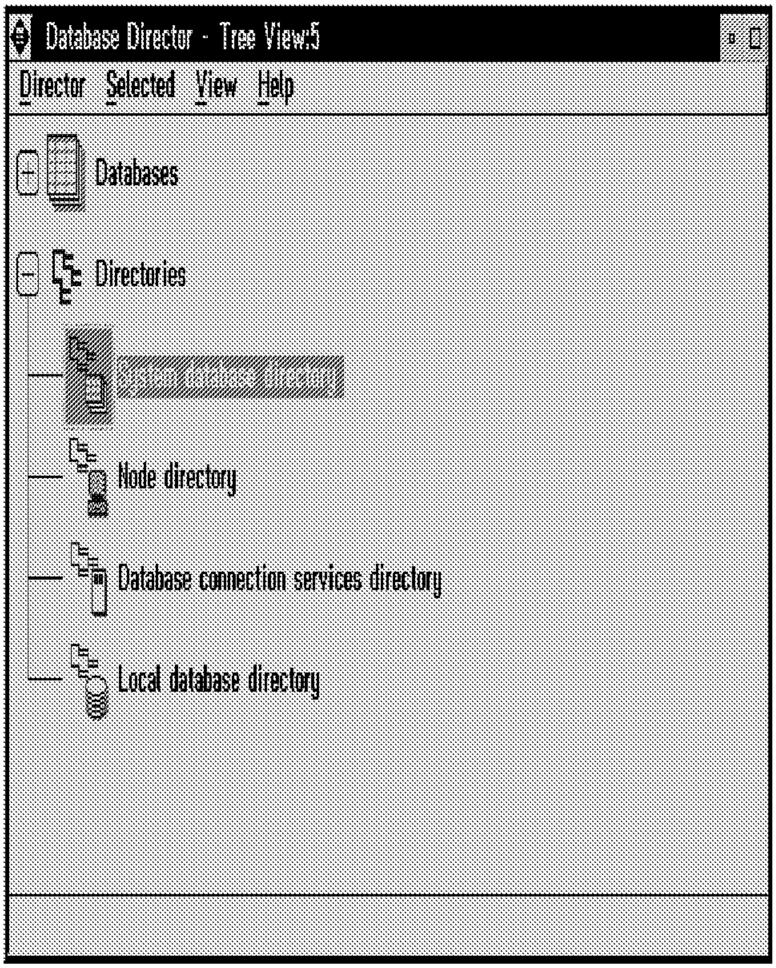
The Database Director allows you to modify the DB configuration via the pop-up menu for the database you wish to configure. As shown in Figure 55, the DB configuration parameters are grouped by type. This is useful in understanding how they relate to each other. Figure 55 shows the notebook settings for a default OS/2 DB2 database. Note the default buffer pool size for OS/2 of 250-4 KB pages, which is approximately 1 MB. You will very likely wish to increase this value if the DB2 server has the physical memory available. The default buffer pool size for an AIX server is 1000-4 KB pages, or approximately 4 MB.

8.3.2.3 Catalog/Uncatalog Databases

The task of cataloging databases and nodes can be confusing when using the Command Line Processor (CLP). Entries in the different directories must match for a successful database connection to occur. The DB2 directories are used to determine where the databases physically reside, either locally or remotely. During each connect statement, DB2 has to find the appropriate information from these files. This action takes extra I/O during each connect statement. To increase performance during a connect statement, these files can now be cached in memory to avoid the extra disk access during all subsequent connections. As shown in Figure 56 on page 191, there are four directories which can be accessed using the Database Director, including:

- System database directory
- Node directory

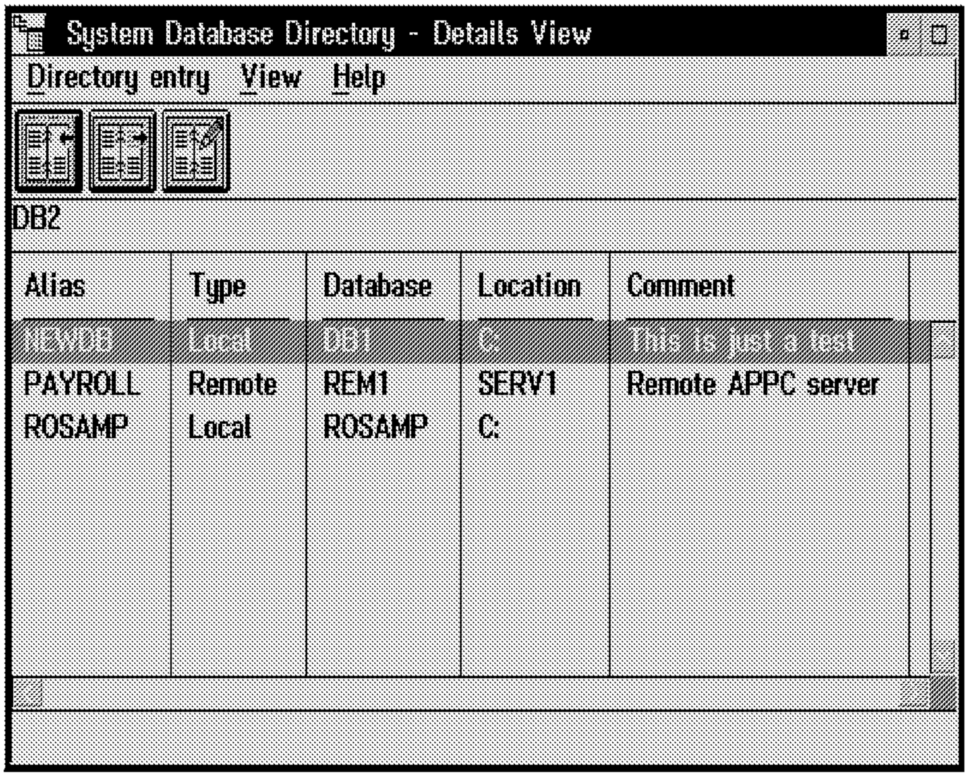
- Database connection services directory
- Local database directory



Database Director - Tree View:5
Director
Selected
View
Help
Databases
Directories
System database directory
Node directory
Database connection services directory
Local database directory

Figure 56. DB2 Directories

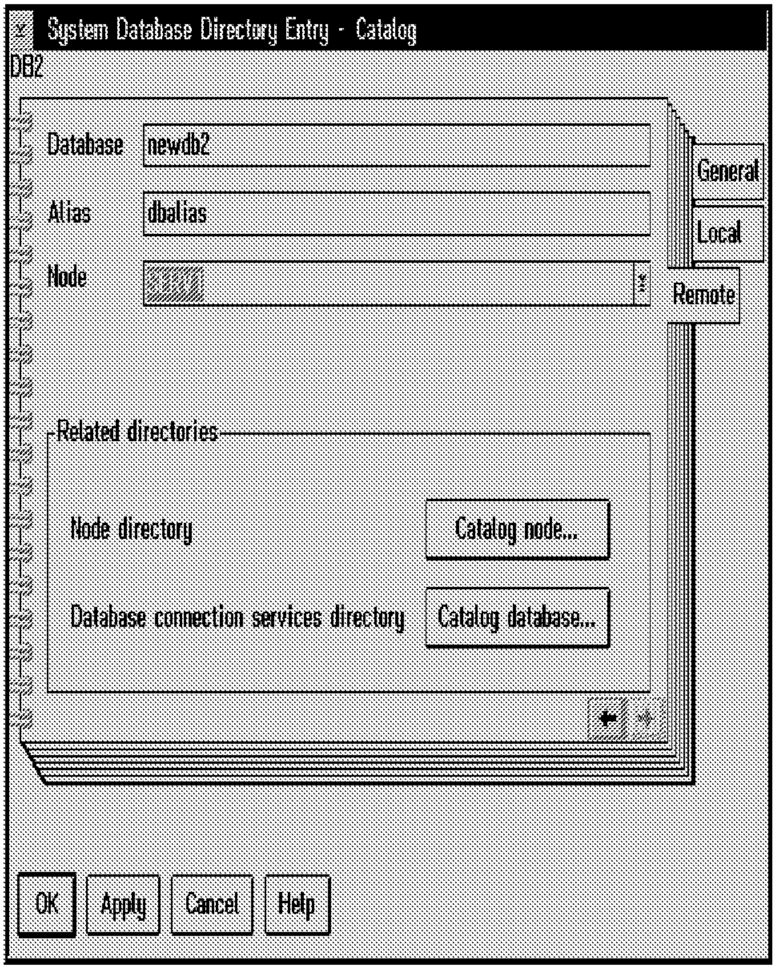
The first DB2 directory that is examined during a `connect to <dbname>` is the system database directory. There is a single system database directory for each DB2 instance. If the database has been cataloged as a local database, the local database directory is then examined to locate the database. If the database has been cataloged as a remote database, the node directory is examined to locate the DB2 server where the database resides. Figure 57 on page 192 shows an example of the system database directory. This example shows two local databases and one remote database.



Alias	Type	Database	Location	Comment
NEWDB	Local	DB1	C:	This is just a test
PAYROLL	Remote	REM1	SERV1	Remote APPC server
ROSAMP	Local	ROSAMP	C:	

Figure 57. System Database Directory

The advantage of using the Database Director to catalog your databases is that the integration of the various DB2 directories makes the task easier. In Figure 58 on page 193, you can see that the corresponding node directory entries can be chosen or defined when the catalog database task is being performed. The CLP interface does not provide an easy method of performing all of the necessary steps to catalog/uncatalog databases.



System Database Directory Entry - Catalog

DB2

Database newdb2

Alias dbalias

Node

Figure 58. System/Node Directory Relationship

8.3.3 Creating/Modifying DB2 Objects

This section shows how to create/modify DB2 objects using the Database Directory. Specifically, the following operations may be performed:

- Creating a Database
- Dropping a Database
- Creating a Tablespace
- Backing up a Database
- Backing up a Tablespace
- Restoring a Database and Changing Container Definitions
- Examining Database Packages
- Using the Performance Monitor

8.3.3.1 Creating a Database

In DB2 Version 2, the task of creating a database has become more flexible. The Database Director makes this task simpler than remembering the complete syntax of the `create database` command. Figure 59 shows how to start the task of creating a new database by using the pop-up menu and selecting the **Databases** object.

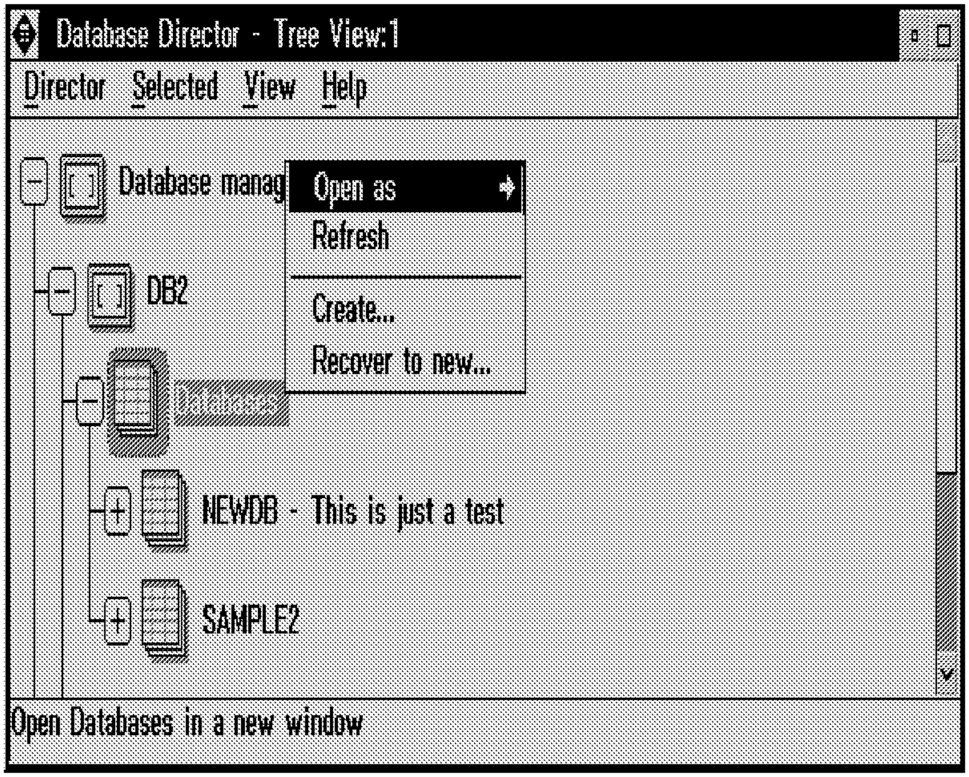
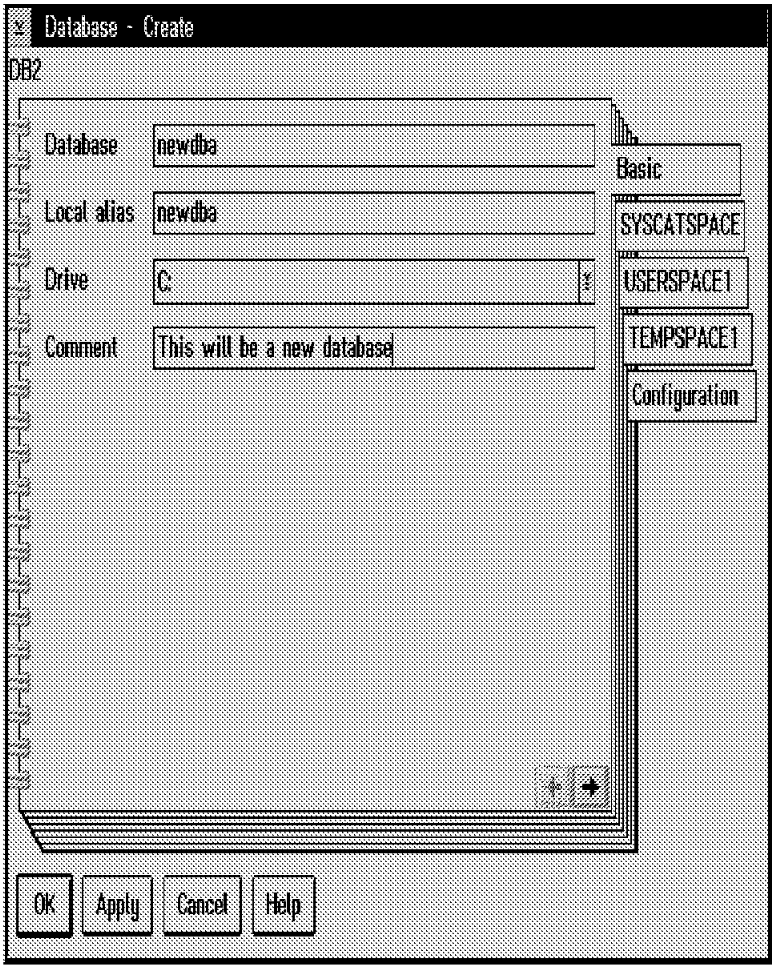


Figure 59. Pop-up Menu for Creating a Database

The notebook settings for creating a database in Figure 60 on page 195 shows that the characteristics of the new database can be easily defined including:

- The location of the three default tablespaces
- The number and location of the containers for the tablespaces
- The collating sequence



Database - Create

DB2

Database: newdba

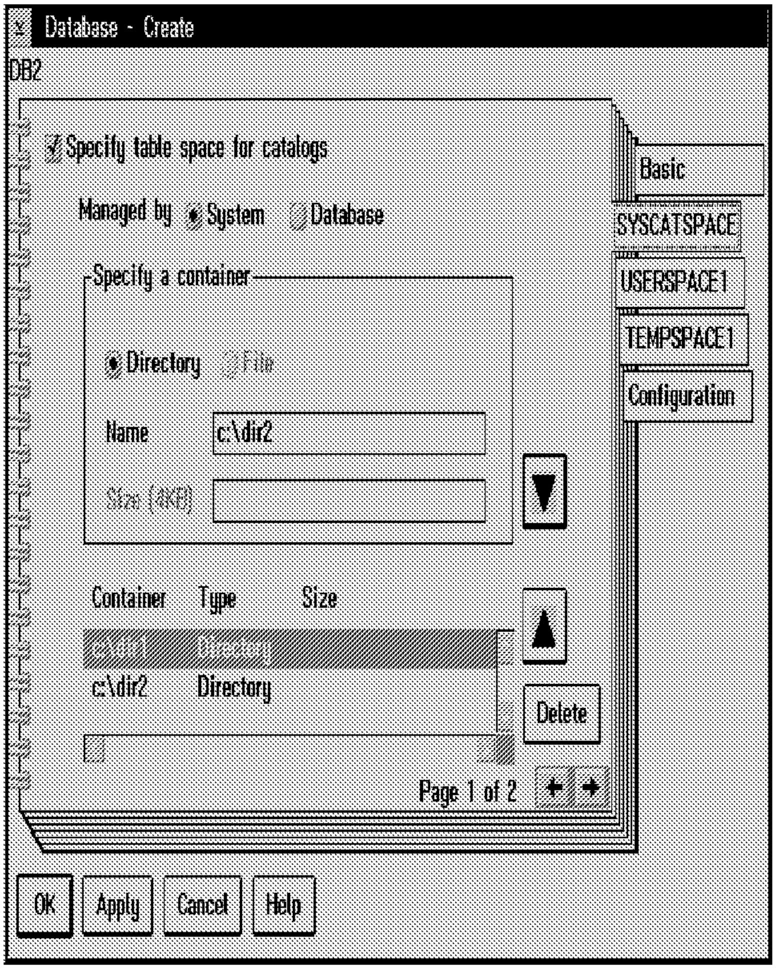
Local alias: newdba

Drive: C:

Comment: This will be a new database

Figure 60. Notebook Settings for Creating a Database

When creating a new database, the default tablespaces must be defined Figure 61 on page 196 shows two containers have been defined for the system catalogs (SYCATSPACE). The Database - Create screen contains two large arrows. One arrow is pointing up and the other down. These arrows are used to add (down arrow) or change (up arrow) the default container definitions for a tablespace. Figure 61 on page 196 shows that there will be two containers defined for the SYSCATSPACE tablespace. Remember that in an SMS (System Managed Tablespace) tablespace, once the containers are defined for the tablespace, no additional containers can be added. However, there is one possiblity that allows you change the container definition when performing a restore operation. This will allow you the opportunity to redefine the containers for a tablespace, whether the tablespace is SMS or DMS. However, the type of tablespace remains the same. For more information on this option, see 8.3.6, “Restoring a Database” on page 200.



Database - Create

DB2

☑ Specify table space for catalogs

Managed by ● System ○ Database

Specify a container

● Directory ○ File

Name c:\dir2

Size (4KB)

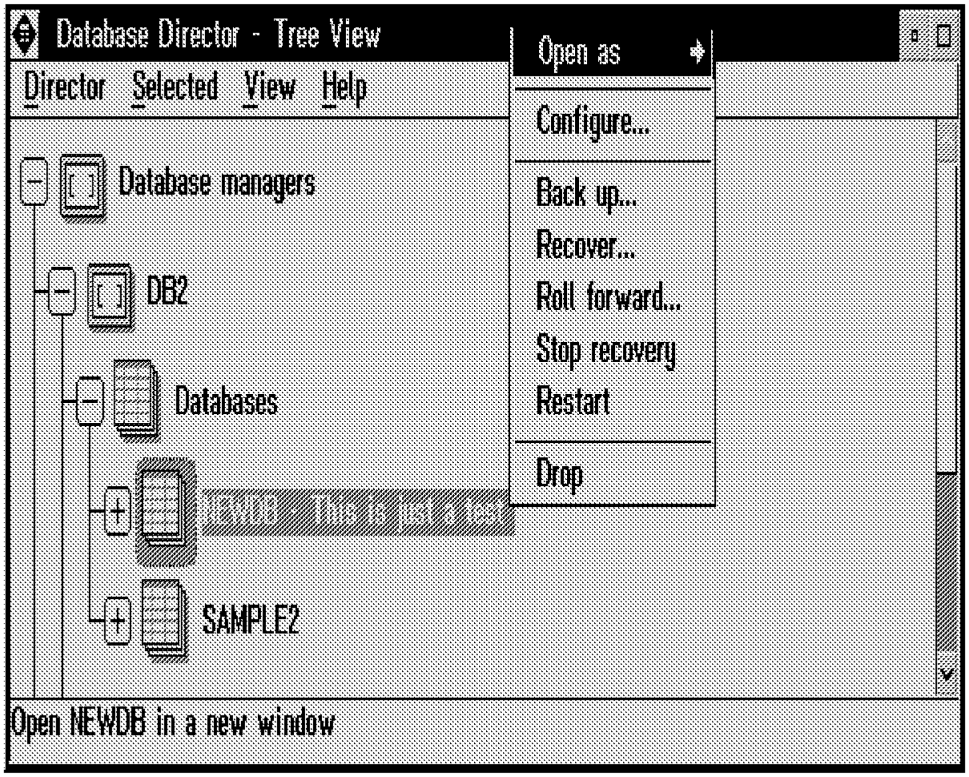
Container	Type	Size
c:\dir1	Directory	
c:\dir2	Directory	

Page 1 of 2

Figure 61. Defining Tablespaces During Database Creation

8.3.3.2 Dropping a Database

The Database Director can be used to drop a database by using the Database Object pop-up Menu. As shown in Figure 62 on page 197, the Database Director will warn you before the database is actually dropped.

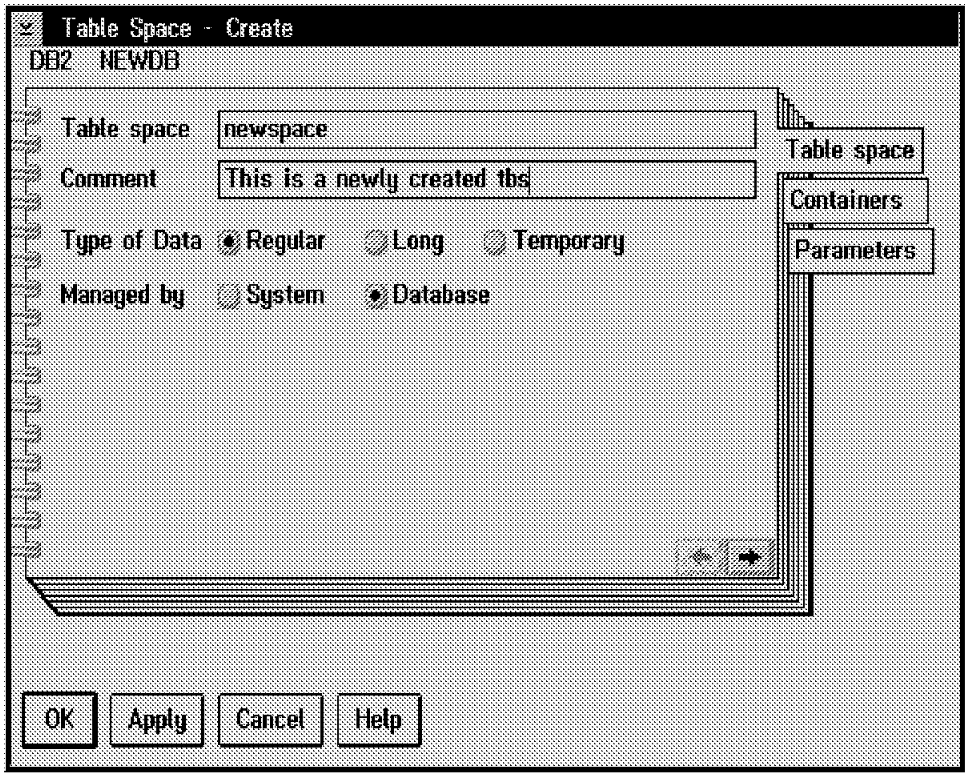


Open NEWDB in a new window

Figure 62. Dropping a Database

8.3.3.3 Creating a Tablespace

The Database Director may be used to create a new tablespace within a database. As seen in Figure 63 on page 198, the type of tablespace (DMS or SMS) can be selected, and the containers for the tablespace can be selected.



DB2 NEWDB

Table space: newspace

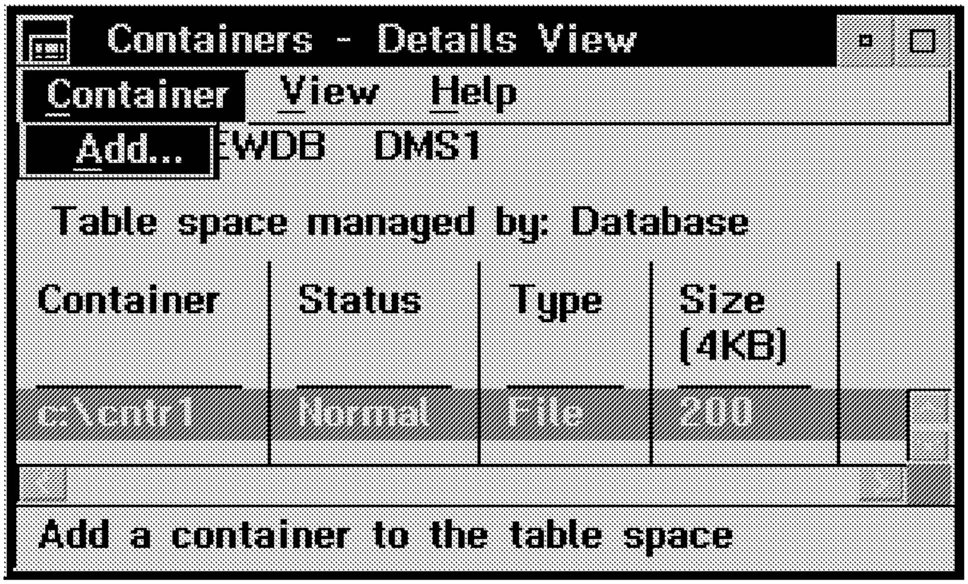
Comment: This is a newly created tbs

Type of Data ● Regular ○ Long ○ Temporary

Managed by ○ System ● Database

Figure 63. Creating Tablespaces

It is possible to add new containers to a DMS tablespace by using the Database Director, as shown in Figure 64. When a new container is added to a DMS tablespace, the data contained within the tablespace is rebalanced by DB2. Alternatively, you can also add containers to a DMS tablespace with the `alter tablespace` command. For more information, see 4.6.3, “Alter Tablespace” on page 80.



EWDB DMS1

Table space managed by: Database

Container	Status	Type	Size (4KB)
c:\cntr1	Normal	File	200

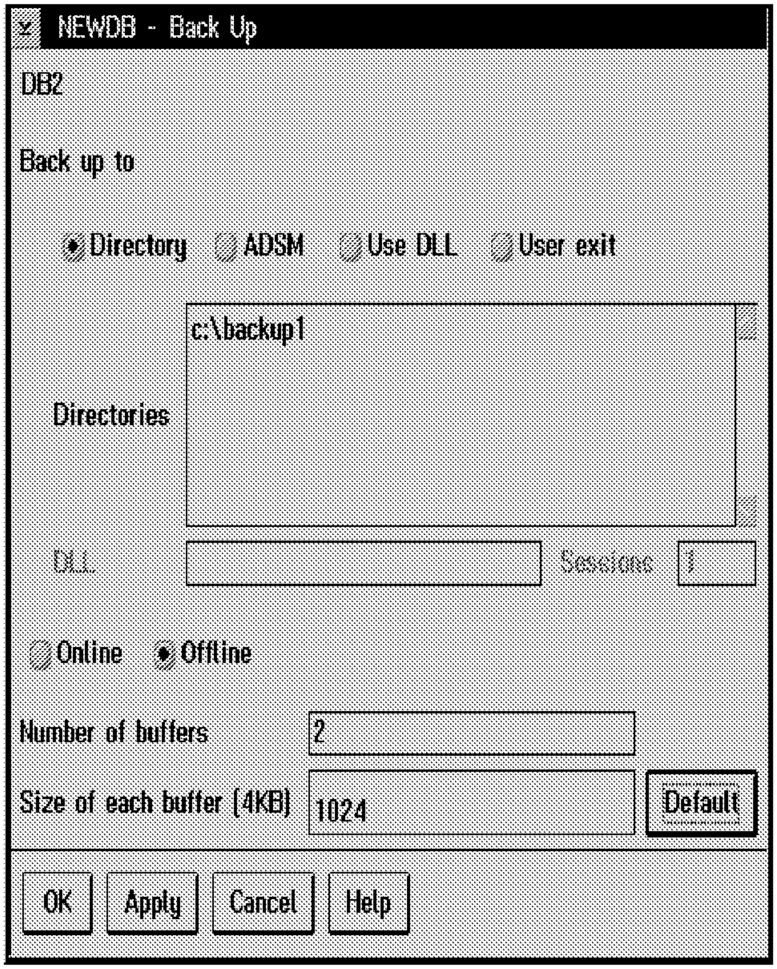
Add a container to the table space

Figure 64. Adding a Container to a DMS Tablespace

8.3.4 Backing up a Database

The Database Director is a an easy-to-use interface for performing backups of a database/tablespace. You can start the backup process and monitor its progress by using the jobs recovery tool. The jobs recovery tool is only accessible via the Database Director in DB2 Version 2.1.1. For DB2 Version 2.1.0, the jobs recovery tool can only be accessed from the command line. Use the db2jobs command in AIX. In OS/2, there is an icon in the DB2 folder for the jobs recovery tool.

Figure 65 shows the Database Director Back Up Database Panel. The same options exist in the Database Director as in the backup database command. The backup image may be placed on disk, sent to tape or sent to an ADSM server.



NEWDB - Back Up

DB2

Back up to

● Directory ○ ADSM ○ Use DLL ○ User exit

Directories: c:\backup1

DLL:

Sessions: 1

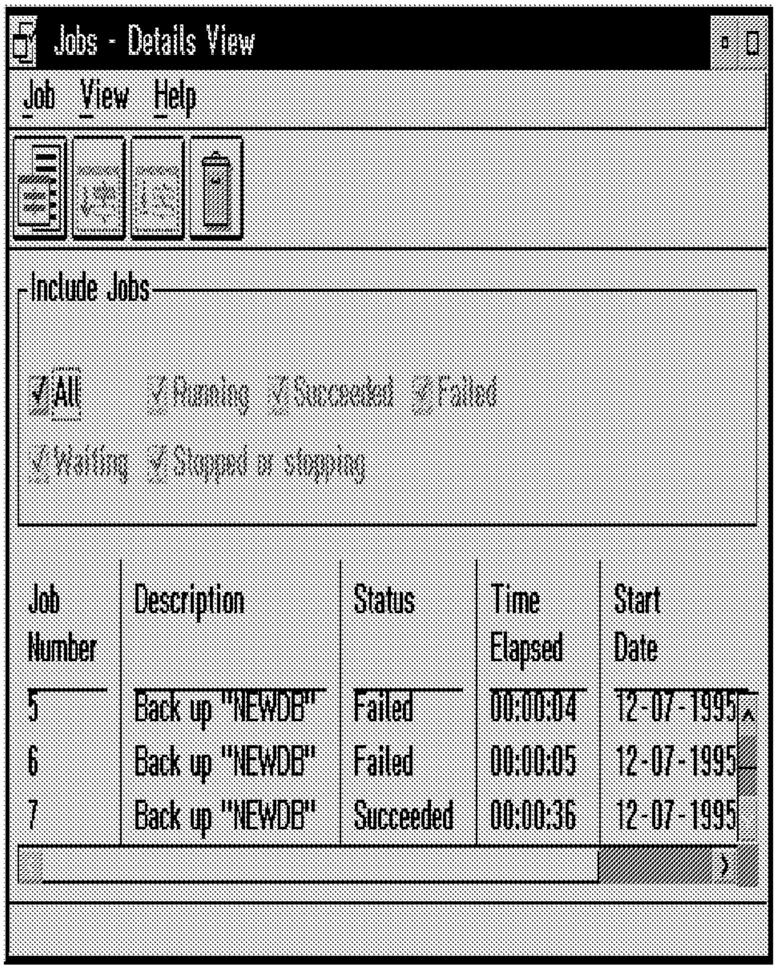
○ Online ● Offline

Number of buffers: 2

Size of each buffer (4KB): 1024

Figure 65. Backing Up a Database

Once the database backup has been initiated, it is assigned a job number. The job number can be used to check the progress of the backup operation by using the jobs recovery tool. Figure 66 on page 200 shows an example of using the jobs recovery tool to check the status of a backup. Note that job number 7 was a successful full database backup.



Include Jobs

☑ All ☑ Running ☑ Succeeded ☑ Failed

☑ Waiting ☑ Stopped or stopping

Job Number	Description	Status	Time Elapsed	Start Date
5	Back up "NEWDB"	Failed	00:00:04	12-07-1995
6	Back up "NEWDB"	Failed	00:00:05	12-07-1995
7	Back up "NEWDB"	Succeeded	00:00:36	12-07-1995

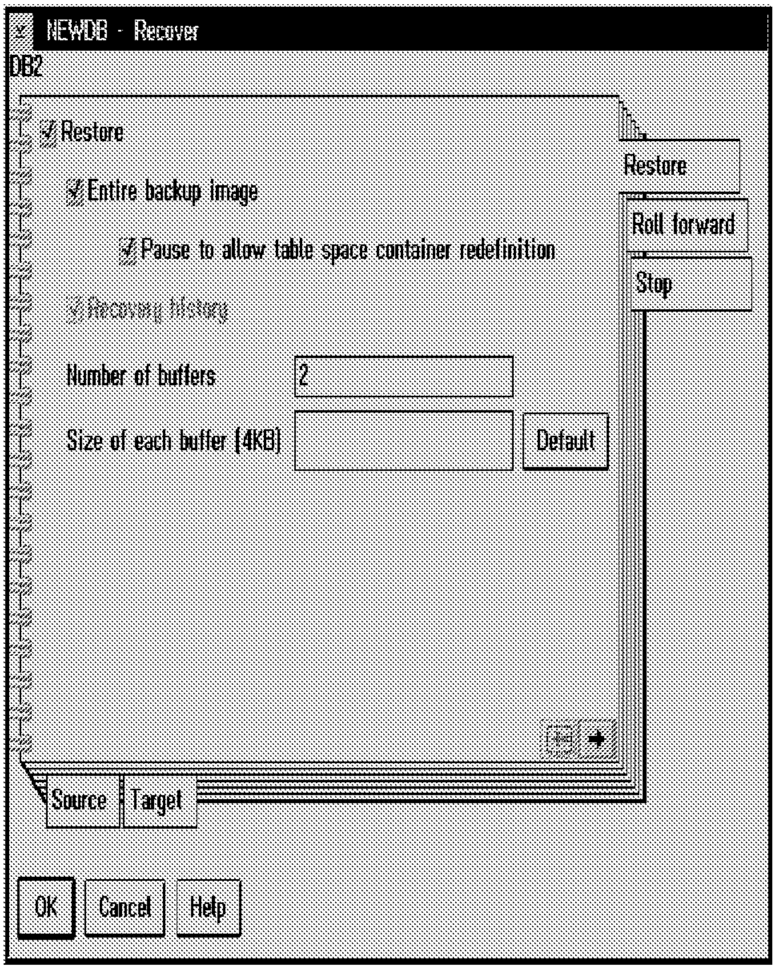
Figure 66. Jobs Recovery Tool

8.3.5 Backing up a Tablespace

The procedure for performing a tablespace-level backup using the Database Directory is very similar to performing a full database backup. To back up a tablespace, use the pop-up menu for the tablespace which is to be backed up. Remember that to perform a table space backup, rollforward recovery needs to be enabled. To examine the progress of a tablespace backup, the jobs recovery tool can be also be used.

8.3.6 Restoring a Database

The task of restoring a previously backed up database image can become complicated, especially if archival logging is being used. The Database Director allows you to perform the restore and as part of the process, perform a rollforward recovery. The Database Director has a recovery database notebook setting which makes the job of restoring a database easier. Figure 67 on page 201 shows the Restore Notebook.



NEWDB - Recover

DB2

☑ Restore

☑ Entire backup image

☑ Pause to allow table space container redefinition

☑ Recovery history

Number of buffers: 2

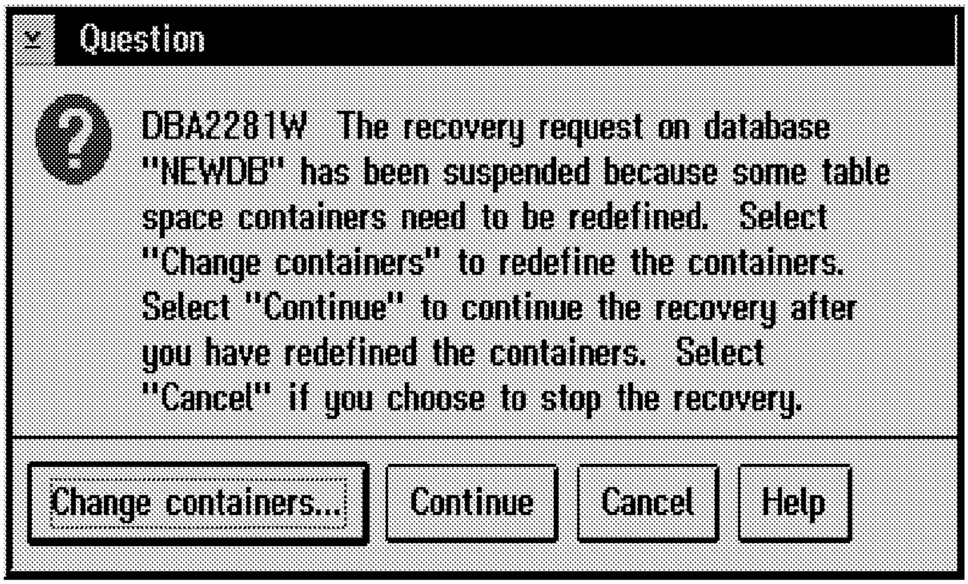
Size of each buffer (4KB):

Figure 67. Changing Container Definition during Restore

Note the ability to **Pause to allow table space container redefinition**. This option allows you the ability to add a new container or change the container locations for both SMS and DMS tablespaces. This is called the Redirected Restore Option.

8.3.6.1 Using the Redirected Restore Option

The redirected restore option is unique to the Database Director. It is not possible when using the CLP or an API.

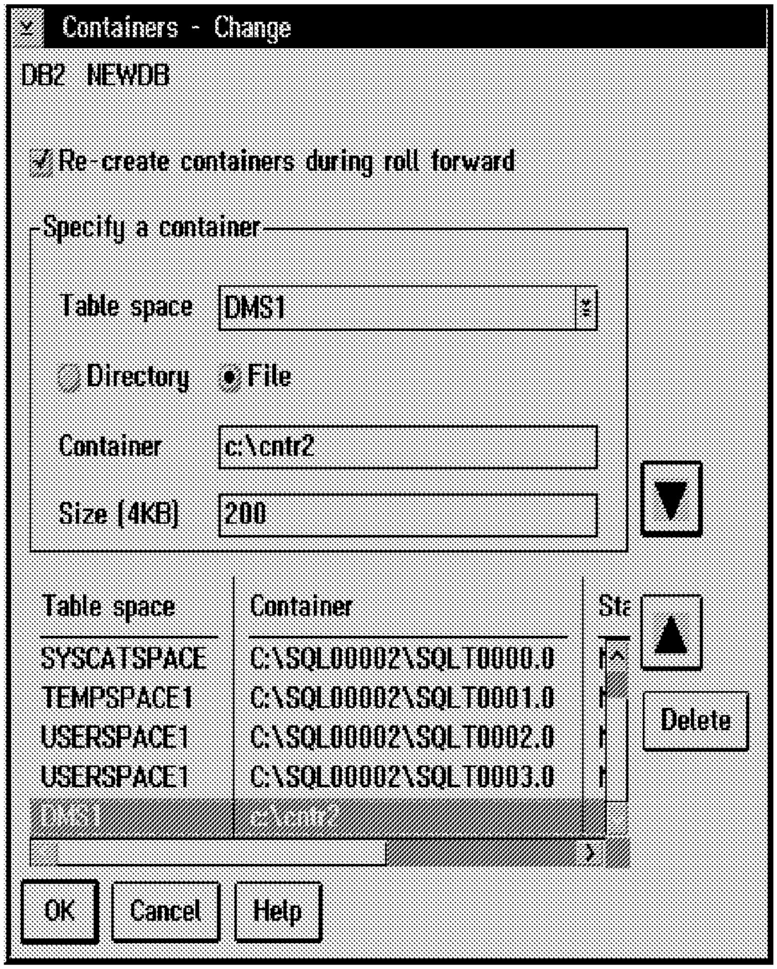


Question

DBA2281W The recovery request on database "NEWDB" has been suspended because some table space containers need to be redefined. Select "Change containers" to redefine the containers. Select "Continue" to continue the recovery after you have redefined the containers. Select "Cancel" if you choose to stop the recovery.

Figure 68. Redirected Restore

The redirected restore will prompt the database administrator to redefine the container definitions before the restore task is started. This is illustrated in Figure 68. Containers can be added or dropped during the restore, as shown in Figure 69 on page 203. Here, a new container has been added to the DMS1 table space. The container location is c:\cntr2.



DB2 NEWDB

☑ Re-create containers during roll forward

Specify a container

Table space: DMS1

○ Directory ● File

Container: c:\cntr2

Size (4KB): 200

Table space	Container	Sta
SYSCATSPACE	C:\SQL00002\SQLT0000.0	
TEMPSPACE1	C:\SQL00002\SQLT0001.0	
USERSPACE1	C:\SQL00002\SQLT0002.0	
USERSPACE1	C:\SQL00002\SQLT0003.0	
DMS1	c:\cntr2	

Figure 69. Redefining Containers

Containers

Directory and file containers are created automatically if they do not exist. Device containers in AIX must exist before attempting the redirected restored. Also, you may have to give ownership/permissions to the containers on the AIX platform.

Following a successful database restore, the Database Director can be used to perform the rollforward phase of database recovery.

After the list of containers has been redefined, a confirmation or revalidation of the list takes place. If there is a problem, you will return to the previous window until the list is correct, or you decide to cancel the operation.

8.3.7 Examining Database Packages

The are two methods of examining Visual Explain output for SQL statements which have had explain snapshots taken.

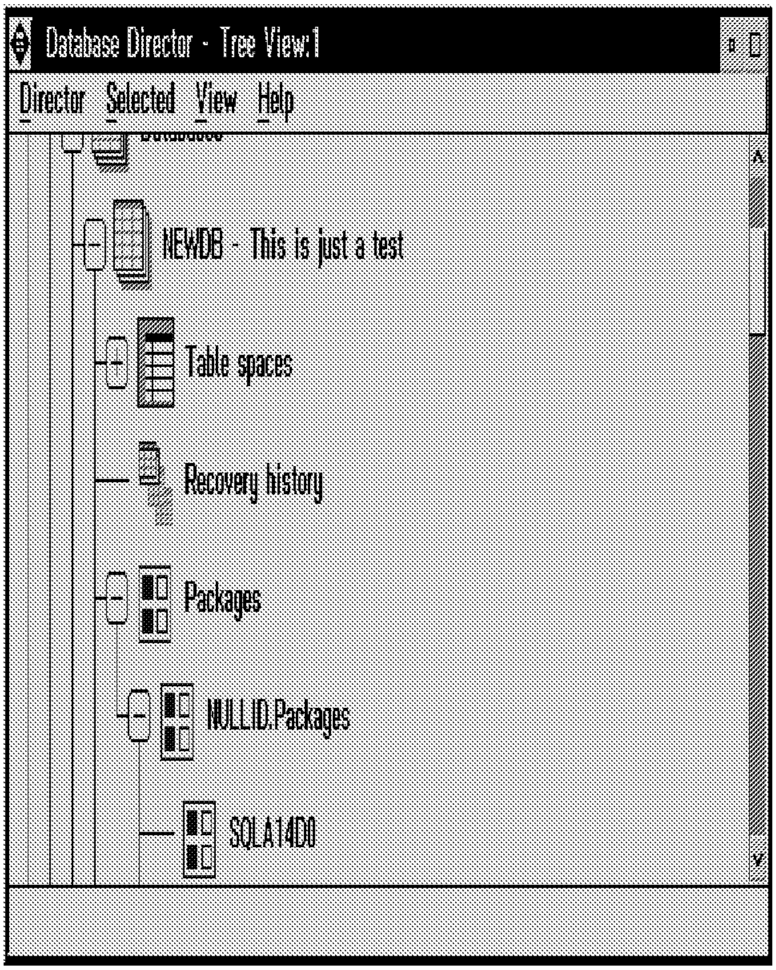
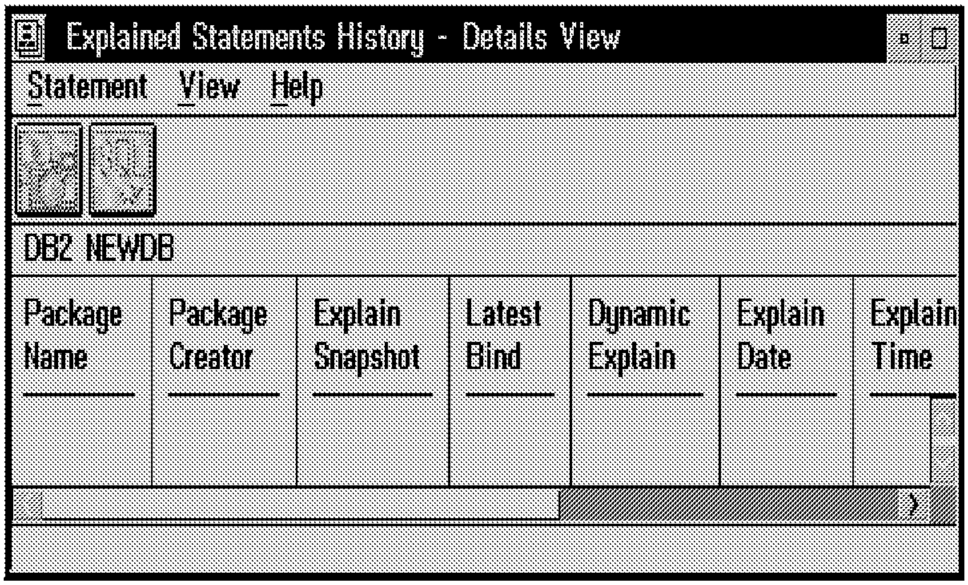

Database Director - Tree View:1
Director Selected View Help
Database
NEWDB - This is just a test
Table spaces
Recovery history
Packages
NULLID.Packages
SQLA14D0

Figure 70. Examining Database Packages

The first method is by using the Database Director and opening the packages for static SQL explain statements, as shown in Figure 70. Another method of examining the Visual Explain output is to open the **Explained statements history** from the Database Director. Figure 71 on page 205 shows the interface for the Visual Explain tool. There will be an entry for each explain snapshot taken. By selecting the statement, a graphical representation of the access plan chosen for the SQL statement will be displayed. The Visual Explain objects and their meanings are documented in the online help facility. To obtain an explain snapshot, the explain tables must be created and the snapshots taken for the SQL statements.



Explained Statements History - Details View

Statement View Help

DB2 NEWDB

Package Name	Package Creator	Explain Snapshot	Latest Bind	Dynamic Explain	Explain Date	Explain Time

Figure 71. Explainable Statements

An example of the output of the Visual Explain tool is shown in Figure 72.

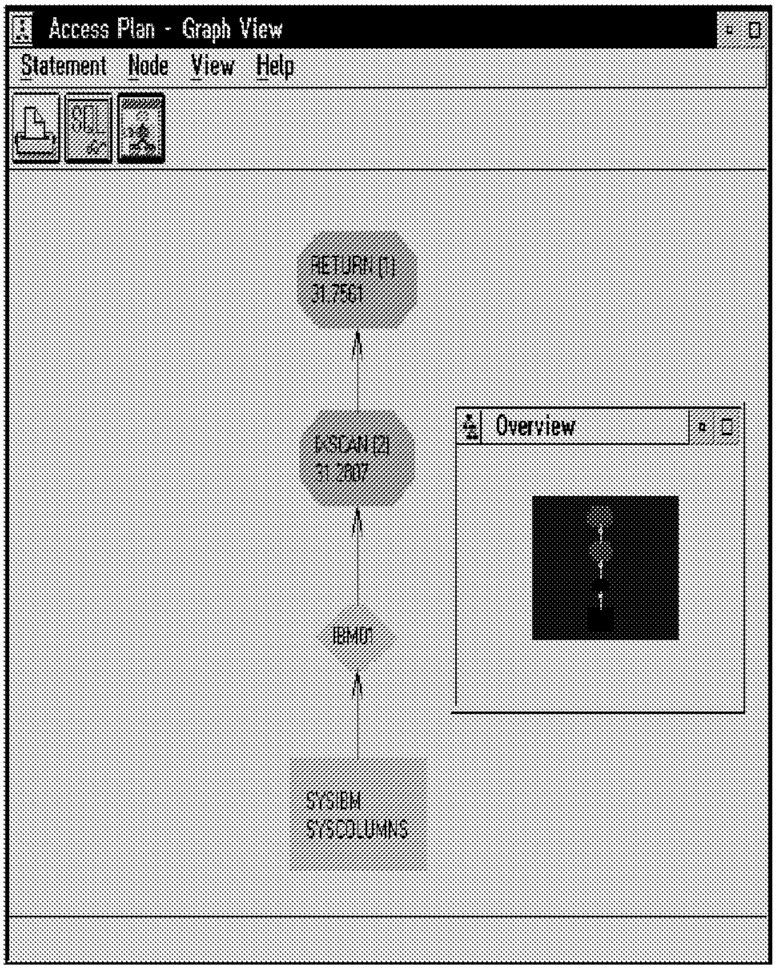


Figure 72. Visual Explain Example

8.3.8 Performance Monitoring

In DB2 Version 2.1.1, there are two new types of Performance Monitor. One is based on the Snapshot Monitor; the other is based on Event Monitor output. The graphical Performance Monitor may be invoked from the Database Director by selecting the database object to be monitored and the appropriate monitor variables.

Appendix A. Database Migration

For those with DB2 Version 1 installed, a successful migration strategy and process are essential. This section reviews the general issues with migration and the specific factors for the OS/2 and AIX platforms. A step-by-step migration process for each environment is described, along with a guide to problem determination. The lessons for our chosen customer scenario are also discussed.

A.1 General Considerations

The general issues to be considered before the migration process is initiated are:

- Version Incompatibilities
- Authentication
- Storage Requirements

A.1.1 Version Incompatibilities

For a comprehensive review of the incompatibilities between Version 1 and Version 2 of DB2 product, consult the *SQL Reference.* However, we draw your attention to two significant incompatibilities which may impact migration, namely:

- View Definitions

  If an existing view involves a SELECT *, the view may be unusable after migration. Such a view must be dropped and re-created to avoid this error.

- Configuration Parameters

  The migration process preserves the database configuration parameters, with the exception of the following where new default parameter values have been assigned:

  - Application Heap Size
  - Package Cache Size (AIX only)
  - Maximum Storage of Lock Lists
  - Statement Heap Size
  - Log Records to Write Before Soft Checkpoint (softmax)

A.1.2 Authentication

Authentication refers to the location of the UserID and password verification. For example, verification can take place on the client workstation, at the database server or at a remote host server. However, a database cannot be cataloged with a mix of authentication types. In addition, all authentication types must match the instance authentication. If mixed types are detected during migration, you can either stop the migration and change the directories or continue with the migration. If you choose to continue, 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. 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 be initiated. Note: All DB2 for OS/2 V1.x databases were server authenticated; in migrating them to Version 2, you will now have a choice of where to perform authentication.

A.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 of the database. As a rule of thumb, you should allow for double the amount of disk space for user and system tables. If there is not enough disk space, migration fails, and all changes are rolled back. Note: Migration is a single transaction (UOW), and V1.x and V2.1 logging is performed.

A.2 Specific Considerations for OS/2

The migration driver shipped with DB2 for OS/2 Version 2 will migrate the following databases.

- Extended Services V1.0 databases
- DB2/2 V1.1 databases
- DB2/2 V1.2 databases

A.2.1 Pre-Migration

Before you install Version 2, you should verify that your databases can be migrated. A command (db2ckmig) is provided to do the verification (on the Version 2 diskette 1 or the CD-ROM). Execute the db2ckmig command using the following syntax:

```
►►──db2ckmig──┬─database─┬──/l filename──┬──────────────────────────┬──────────────────►◄
              └─/e───────┘               └─/u userid──/p password─┘
```

database Specifies an alias name of a database to be scanned.

/e Specifies that all local cataloged databases are to be scanned.

/l Specifies a file to keep a list of errors and warnings generated for the scanned database(s).

/u An optional parameter that specifies the UserID of the system administrator.

/p An optional parameter that specifies the password of the system administrator′s UserID.

An error is logged for each database that is in one of the following states:

- backup pending
- rollforward pending
- database inconsistent

and for each database object that uses either SYSCAT or SYSSTAT as the qualified object name. Here is an example of the output file report from the db2ckmig command:

```
           * * * ERROR * * *

     object name: 'TABLES'
     object type: 'TABLE'
     bad schema name: 'SYSCAT'
```

Correct all the errors that are reported for databases that are in one of the following states:

- Backup pending

  Perform a backup of the database.

- Rollforward pending

  Recover the database as required; perform or resume a ROLLFORWARD DATABASE.

- Database inconsistent

  Restart the database to return it to a consistent state.

If the databases contain one or more objects that use SYSCAT or SYSSTAT as schema names, as in the example, then these objects must be dropped and recreated using a different schema name.

A.2.2 Remote Unattended Migration

The db2cidmg migration program can be useful in the Configuration, Installation, and Distribution (CID) architecture environment and allows for remote, unattended installation and configuration on LAN-based workstations. The db2cidmg program does not accept all the options generally supported by software distribution manager (SDM) products. The syntax for invoking the product is:

```
►►──db2cidmg──┬─database─┬──┬─────┬──┬────┬──────────────────────────────────►◄
              ├─/r───────┤  └─/l1─┘  └─/b─┘
              └─/e───────┘
```

database Specifies an alias name for the database to be migrated. The database must be cataloged on the target workstation, but it can be a local or a remote database.

/r Specifies a response file to be used for CID migration. The response file is an ASCII file containing a list of databases which are to be migrated

/e Indicates that every single database cataloged in the system database directory is to be migrated.

/l1 Specifies the path name of the file to which error log information from remote workstations can be copied after the migration process is completed. If more than one database is specified in the response file, the log information for each database migration is appended to the end of the file. This log file only records errors or warnings that occur during the database migration; any CID interface-related error will not be logged.

/b Indicates that all packages in the database are to be rebound once migration is complete

You must have NetView DM/2 on your LAN to use CID migration.

A.2.3 Parameter Value Changes

Parameter values, previously allocated in units of 64 KB segments, are now multiplied by 16 to allow for allocation in units of 4 KB pages. In addition, the "Log Records to Write Before Soft Checkpoint (softmax)" will be set to the default value since this parameter is now measured as a percentage of the log records rather than as the absolute number of records.

A.2.4 Restoring DB2 Version 1.x Databases

An executable called db2resdb is provided in the \sqllib\misc directory to support the restoring of back-level database backups taken from from Version 1.x and Extended Services databases. It provides an alternative approach to migration, if required. The executable is provided "as is" without warranty of any kind. The syntax is:

```
►►──db2resdb──database-alias──source drive──target drive──────────────────────►◄
```

The restored databases will have SMS tablespaces.

A.3 OS/2 Database Migration Procedure

The migration process comprises two distinct, but interrelated, phases: DB2 product file migration and user database migration. By installing the DB2 V2.1 product, an automatic migration of V1.x files and configuration parameters takes place. Figure 73 shows these migration phases.

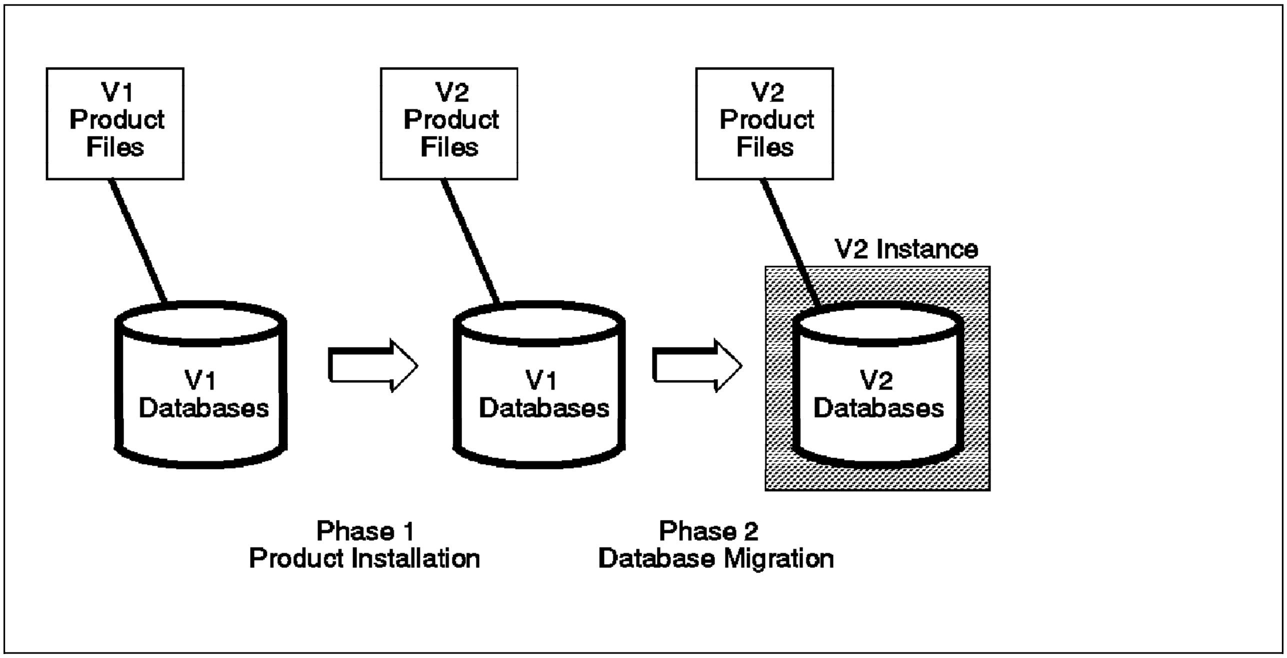


Figure 73. OS/2 Database Migration Phases

The procedure for migration is:

Step 1: Acquire SYSADM authority in order to migrate databases. Back up all the databases. Should the migration fail, this backup can be used to restore the backup to Version 2; the restored database is then implicitly migrated.

Step 2: Ensure that all databases that you wish to migrate are cataloged. End all applications that are using the database manager.

Step 3: Run `db2ckmig` command to verify that databases can be migrated.

Step 4: Use the Command Line Processor to list the contents of the system database directory, node directory and DDCS directory to a file; if you need to move back to Version 1 for any reason, you can reference this file and re-catalog all entries. An alternative method is to copy the entire system database directory and save it on your workstation.

You do not need to save your local database directories (volume directories). All local database directories will be migrated during installation, and a backup copy of the pre-migrated local database directory file in the local database directory will be saved. The backup copy is named SQLDBDIR and has a file extension xxx, where xxx is a number between 1 and 999. If you attempt to install and de-install between Version 2 and the previous version, the backup copy of SQLDBDIR uses the next highest file extension number.

If you need to go back to Version 1 to access the database, you must delete SQLDBDIR and SQLDBBAK from the local database directory, and copy SQLDBDIR.xxx to both SQLDBDIR and SQLDBBAK. You must also de-install Version 2, re-install Version 1, and then re-catalog all entries to the system database directory.

Step 5: Stop the database manager by issuing the `stopdbm` command. Install DB2 Version 2.1. An instance of DB2 is automatically generated. The installation program replaces Version 1 product files with Version 2 product files and migrates Version 1 directories and configuration information by merging relevant values from Version 1 with Version 2 defaults. The following are migrated (if applicable):

- DB2 settings in CONFIG.SYS
- Database manager configuration file
- Database directories

  system
  local

- Node directory
- Gateway directory

Reboot the operating system to implement the new CONFIG.SYS settings. DB2 then requests that you authenticate yourself by using the default UserID and password, or otherwise. Secondly, it requests whether or not you wish to create the sample database; if you did not drop the sample database under Version 1, you will not be able to create the sample database under Version 2. Next, start the database manager using the `db2start` command.

Step 6: Migrate database(s) using the `migrate database` command or the API. The syntax is:

```
►►──migrate database──database-alias──┬──────────────────────────┬──────────────────────►◄
                                      └─/u userid──/p password─┘
```

The migrated databases will remain in SMS format.

Step 7: If you have a local database directory (or its files) backed up before installation and you would like to use it so that the database from this local database directory can be migrated, you must first create directory SQLDBDIR (if it does not already exist) on the drive. Copy the directory files which have been backed up to the directory SQLDBDIR.

Run db2migdr against this local database directory drive to migrate its directory entries to Version 2 format. The syntax is:

```
              ┌─,─────┐
►►──db2migdr───▼─drive─┴──────────────────────────────────────────────────────────────────►◄
```

For example, db2migdr c f migrates the local database directory from drive c and from drive f.

If the system database directory does not have the entries cataloged in the local database directory, you must first catalog them to the system database directory before you can migrate the database.

Step 8 (Optional): Re-bind all database packages. All the packages will have been marked as invalid during the catalog migration. You can now use a command called db2rbind to re-validate the packages, or allow package re-validation to occur implicitly when a package is first used after migration. The db2rbind tool uses the REBIND SQL statement on all packages. The syntax is:

```
►►──db2rbind──database-alias──/l=logfile──┬──────────────────────────┬──────────────────►◄
                                          └─/u userid──/p password─┘
```

You could use the REBIND (or BIND) SQL statement to validate each package in turn, as required, instead of running db2rbind at a global level. Also, there is a special column in the SYSPLAN table that identifies if the bind was explicit or implicit.

Step 9 (Optional): In order to take advantage of Version 2 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; consult the *Administration Guide* for details.

A.4 Specific Considerations for AIX

The migration driver shipped with DB2 for AIX Version 2 will migrate the following databases:

- DB2/6000 V1.1 databases
- DB2/6000 V1.2 databases

A.4.1 User/Group Security

The AIX operating systems allows the same name to be created for a user and a group. This can cause problems for authorization and privilege checking within DB2. There needs to be a way to indicate if the privilege is intended for a user, group or both.

During migration, the authorization catalog tables are checked to determine if existing privileges are for users or groups, and the GRANTEETYPE (′U′ or ′G′) is determined. The following condition-check occurs:

- If GRANTEE in a V1.x table is a USER or undefined, then GRANTEETYPE is set to U.
- If GRANTEE in a V1.x table is a GROUP, then GRANTEETYPE is set to G.
- If GRANTEE in a V1.x table is both a USER and a GROUP, then GRANTEETYPE is set to U.

In DB2 Version 2, changes have been made to the GRANT and REVOKE SQL statements to add an optional parameter in the TO/FROM authorization-name clause which is used to indicate if the privilege is intended for a user or group. As a result, if a user and a group have the same name in the previous database version, the authority and privilege to the group must be explicitly regranted after migration. This condition check may result in a member of a group no longer having authority for database objects that he had in V1.x. If the user within the group still requires the authority, then you must explicitly GRANT the authority at the group level.

A.4.2 Instance Migration

Instances were introduced in DB2 for AIX Version 1 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. In considering migration, you may want to retain these separate instances under Version 2. So, in the AIX environment, the migration process comprises two distinct, but interrelated, phases:

- instance migration
- database migration

Figure 74 on page 214 shows these migration phases.

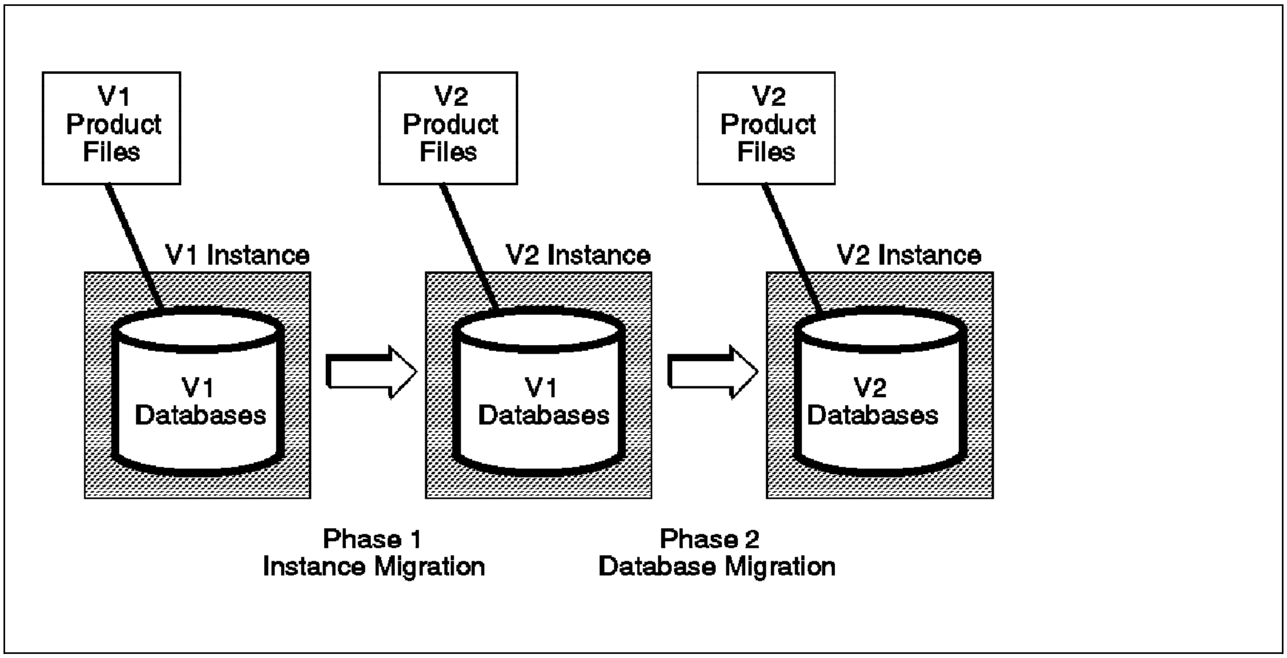


Figure 74. AIX Database Migration Phases

To perform an instance migration:

1. Do not delete Version 1 of DB2 (/usr/lpp/db2_01_0000) until your migration work is completed and tested.
2. Log on as SYSADM for the instance you are migrating. Back up all databases in that instance.
3. Ensure that all local databases that you wish to migrate are cataloged.
4. End all applications that are currently using the instance.
5. Stop the database manager and the Command Line Processor.
6. A program script (db2imigr) is provided to migrate an existing instance to Version 2. The instance migration routine is invoked from the /usr/lpp/db2_02_01/instance directory as follows:

```
►►──db2imigr──instanceName──┬─────────────────────┬──────────────────────────────►◄
                            └─-a──┬─server─┬──┘
                                  ├─client─┤
                                  └─dcs────┘
```

instanceName The name of the existing instance to be migrated.

-a An optional parameter which specifies the authentication type for the new instance. Valid authentication types are server, client or DCS. If the -a parameter is not specified, then the authentication type will default to server.

While this script does not automatically initiate the migration of databases, it detects conditions that would prevent the successful migration of any local databases that are cataloged in the instance. If db2imigr detects any of these conditions, it aborts the instance migration and generates a report that lists the conditions that were detected.

7. Correct all the errors that were reported. For example, if a backup-pending state was detected, perform a full database backup to clear this error condition, and run `db2imigr` again. When the `db2imigr` verification finds no errors, the instance migration will be initiated.

8. Within the automatic instance migration process, the following actions are performed:

   - The Version 1 instance is backed up.
   - A Version 2 instance is created.
   - The Version 1 system database directory is copied over to Version 2 instance (no changes), along with the node directory (add file server and object name) and the DCS directory (no changes).
   - The Version 2 instance database manager configuration file is merged with the Version 1 database manager configuration file.
   - The list of instances is updated with this new Version 2 instance.
   - The db2profile and db2cshrc files in Version 1 are copied to db2profile.v1 and db2cshrc.v1 files under the Version 2 instance.
   - The contents of the Version 1 function directory are copied to the Version 2 instance.

9. Instance migration is then complete, and database migration can begin.

Note: Once an instance has been migrated to Version 2, it is unusable in Version 1. You must perform a `db2imigrev` instance_name to return the instance to Version 1 format.

You should consult the *DB2 for AIX Installation and Operations Guide* for a full discussion of instance migration, if required.

A.5 AIX Database Migration Procedure

The following steps are recommended for the database migration process.

Step 1: Acquire SYSADM authority in order to migrate databases. Back up all the databases. Should the migration fail, this backup can be used to restore the backup to Version 2; the restored database is then implicitly migrated.

Step 2: Ensure that all databases that you wish to migrate are cataloged. End all applications that are using the database manager and the instance. Stop the database manager and the command line processor by issuing the `db2stop` and `db2 terminate` commands.

Step 3: Perform instance migration, and this will ensure that databases can be migrated and that there are no outstanding, pending situations.

Step 4: Install DB2 Version 2.1 product files. Create an instance of DB2 and start the database manager by using the `db2start` command.

Step 5: Migrate database(s) using the `migrate database` command or the API. The syntax is:

```
►►──migrate database──database-alias──┬───────────────────────────┬───────────────────►◄
                                      └─/u userid──/p password─┘
```

The migrated databases will remain in SMS format.

Step 6 (Optional): Re-bind all database packages. All the packages will have been marked as invalid during the catalog migration. You can now use a command called db2rbind to re-validate the packages, or allow package re-validation to occur implicitly when a package is first used after migration. The db2rbind tool uses the REBIND SQL statement on all packages. The syntax is:

```
►►──db2rbind──database-alias──/l logfile──┬──────────────────────────┬──────────────►◄
                                           └─/u userid──/p password─┘
```

You could use the REBIND (or BIND) SQL statement to validate each package in turn, as required, instead of running the db2rbind at a global level. Also, there is a special column in the SYSPLAN table that identifies if the bind was explicit or implicit.

Step 7 (Optional): In order to take advantage of Version 2 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; consult the *Administration Guide* for details.

A.6 Problem Solving

During the migration step, error code SQL1704N may be returned with one of five reason codes:

Reason Code 1 An invalid schema was found.

Reason Code 2 Database is abled be migrated because database is in one of the following states: backup pending, rollforward pending, or database inconsistent.

Reason Code 3 Database logs are full.

Reason Code 4 Insufficient disk space available.

Reason Code 5 The database configuration file cannot be updated.

The resolution of these problems is as follows:

1. The reserved schema names are: SYSIBM, SYSCAT and SYSSTAT. Ensure that all the database objects that use one or more of these schema names are dropped and re-create the objects using a different schema name. This correction must be made in the Version 1 database manager. Resubmit the database configuration command under Version 2.
2. Correct the database state by going back to Version 1 of the database manager. See the *Administration Guide* for instructions on the actions that are required to restore a consistent database state. Resubmit the database migration command under Version 2.
3. Increase the database configuration parameters, LOGFILSIZ or LOGPRIMARY, to a larger volume. Resubmit the database migration command.
4. Increase the disk space available and resubmit the database migration command.

5. Ensure that the database configuration file is not being held exclusively by any users and can be updated. Resubmit the database migration command.

A.7 Customer Scenario

The migration procedure for both the OS/2 and AIX environments should be easy to follow, but a number of issues of a general nature should be highlighted again for the customer scenarios, namely:

- Back up all V1.x databases before starting a migration procedure.
- Do not delete any V1.x product files unless automatically deleted by the V2.1 install process or unless the migration has completed successfully.
- Re-bind all databases packages after migration, either explicitly or implicitly.
- Tune your application and database environment to take advantage of V2.1 enhancements by reviewing configuration parameters and application design.

A.7.1 Telephone Company

- A parallel cut-over environment is possible where DB2 V1 could exist in one instance until DB2 V2.1 is installed and productive in a second instance.
- Care is required in reviewing the DB2 V1 authorities and privileges before moving to V2.1 to avoid privileges being inadvertently removed from an individual user.
- In their complex network environment, care is required in checking the system and node directory entries for the migrated database.

List of Abbreviations

ADSM	Adstar Distributed Storage Manager
AIX	Advanced Interactive Executive
ASC	Non-delimited ASCII
ASCII	American National Standard Code for Information Interchange
BLOB	Binary Large Object
CLOB	Character Large Binary Object
CAE	Client Application Enabler
CLP	Command Line Processor
DB	Database
DB2	DATABASE 2
DBADM	Database Administration
DBLOB	Double Byte Binary Large Object
DBM	Database Manager Configuration
DDCS	Distributed Database Connection Services
DEL	Delimited ASCII
DPropR	Data Propagator Relational
DRDA-AR	Distributed Relational Database Architecture Application Requestor
DRDA-AS	Distributed Relational Database Architecture Application Server
DMS	Database Managed Space (Storage)
IBM	International Business Machines Corporation
ITSO	International Technical Support Organization
IXF	Integrated Exchange Format
PC/IXF	Personal Computer/Integrated Exchange Format
SDK	Software Developer′s Kit
SMS	System Managed Storage (Space)
SQL	Structure Query Language
SYSADM	System Administrator
SYSCTRL	System Control
SYSMAINT	System Maintenance
TCP/IP	Transmission Control Protocol/Internet Protocol
WAL	Write-Ahead-Logging
WSF	Work-Sheet Format

Index

ITSO Technical Bulletin Evaluation RED000

International Technical Support Organization
DB2 Version 2 Planning Guide
for Database Administrators
January 1996

Publication No. SG24-2523-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____

Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

____

____

What other topics would you like to see in this Bulletin?

____

____

What other Technical Bulletins would you like to see published?

____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

______________________________ ______________________________
Name Address

______________________________ ______________________________
Company or Organization

______________________________ ______________________________
Phone No.

ITSO Technical Bulletin Evaluation **RED000**
SG24-2523-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Department JN9B, Building 045
Internal Zip 2834
11400 BURNET ROAD
AUSTIN TX
USA 78758-3493

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.

SG24-2523-00